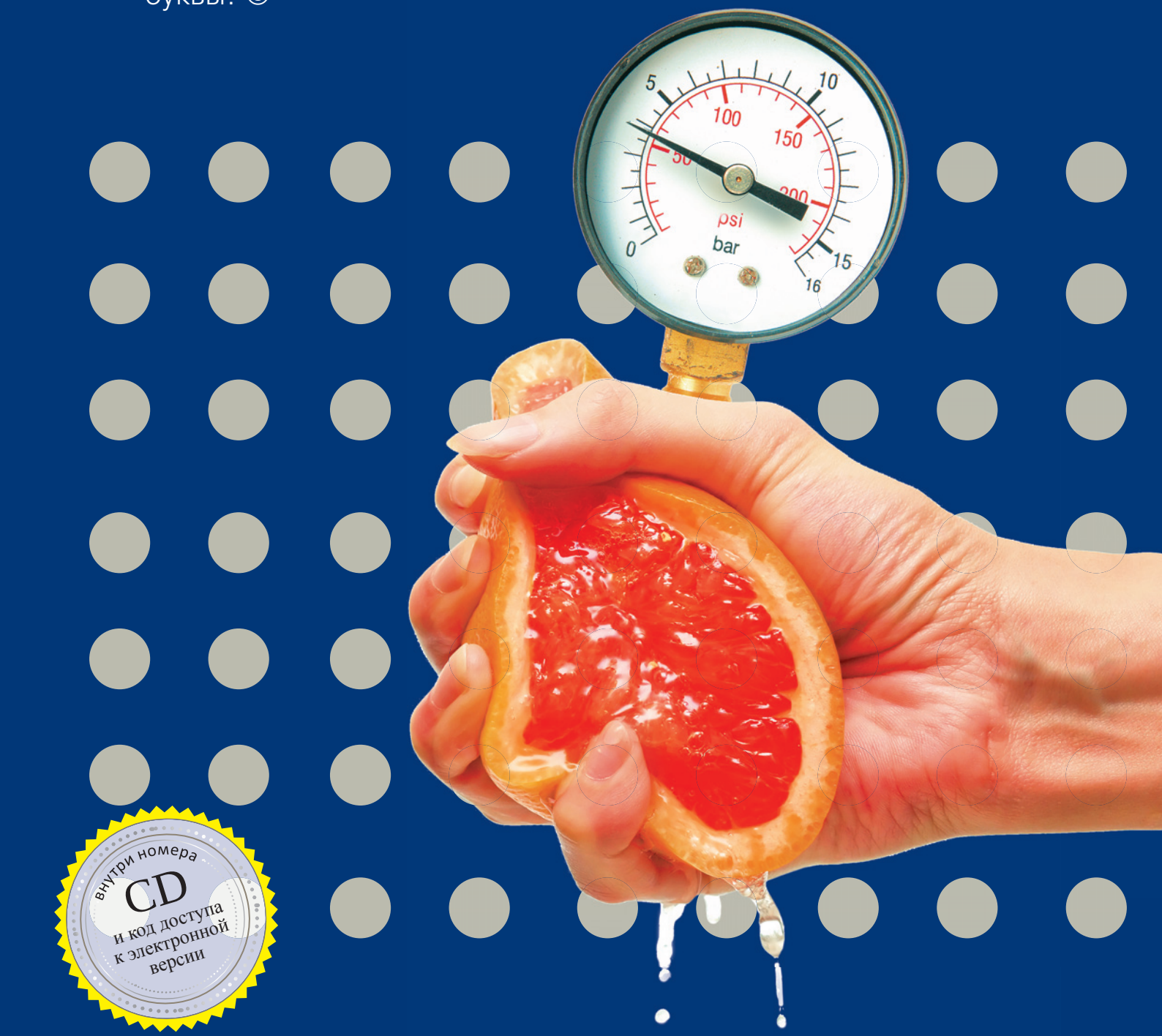
№ 11 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА

4
Влга впдт в Кспске мре
Если и так понятно, зачем "лишние" буквы? ☺

16
Мама мыла ламу
А дети в это время писали ЕГЭ

22
$(2 \cdot 10_8)^{2010} - 4^{2011} + 2^{2012}$
B7 — убит!
Задача ЕГЭ



внутри номера CD и код доступа к электронной версии

издательский дом 1september.ru
Первое сентября | декабрь 2012
ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

Интересный вопрос: алгоритмы какого типа чаще всего используются в процессе реальной работы обычного “пользовательского” компьютера? Ну, арифметику не берем, понятно, что все в конце концов к ней сводится. Нас интересуют классы алгоритмов уровнем повыше. Думаете, алгоритмы сортировки? Поиска? Похоже, что на самом деле самыми используемыми являются алгоритмы компрессии-декомпрессии. И речь не только о “ручном” вызове архиваторов — это даже не в счет. Файлы сжимаются, картинки сжимаются, данные при передаче по каналам связи сжимаются. А ведь еще и разжимать надо ☺.

В НОМЕРЕ

3 ПАРА СЛОВ
- Удивительное рядом ☺

4 ПРОФИЛЬ
- Сжатие данных: теория и практикум

16 ЕГЭ
- Еще раз про однозначное декодирование
- Сколько единиц и сколько нулей?
- Решение систем однородных логических уравнений

44 ВНЕКЛАССНАЯ РАБОТА
- Вопросы по информатике для проведения школьных конкурсов “Что? Где? Когда?” и “Брейн-ринг”

47 ИНТЕРНЕТ
- В Интернет — на “машине времени” ☺

48 ЗАНИМАТЕЛЬНЫЕ МАТЕРИАЛЫ ДЛЯ ПЫТЛИВЫХ УЧЕНИКОВ И ИХ ТАЛАНТЛИВЫХ УЧИТЕЛЕЙ
- “В мир информатики” № 182

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Файлы к практикуму “Сжатие данных”
- Презентации и дополнительные материалы к статьям номера

ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ: по каталогу “Роспечати”: 32291 (бумажная версия), 19179 (электронная версия); “Почта России”: 79066 (бумажная версия), 12684 (электронная версия)

http://inf.1september.ru

Учебно-методический журнал для учителей информатики
Основан в 1995 г.
Выходит один раз в месяц

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 18 424 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ “ПЕРВОЕ СЕНТЯБРЯ”
Главный редактор: Артем Соловейчик (генеральный директор)
Коммерческая деятельность: Константин Шмарковский (финансовый директор)
Развитие, IT и координация проектов: Сергей Островский (исполнительный директор)
Реклама, конференции и техническое обеспечение Издательского дома: Павел Кузнецов
Производство: Станислав Савельев
Административно-хозяйственное обеспечение: Андрей Ушков
Главный художник: Иван Лукьянов
Педагогический университет: Валерия Арсланьян (ректор)

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

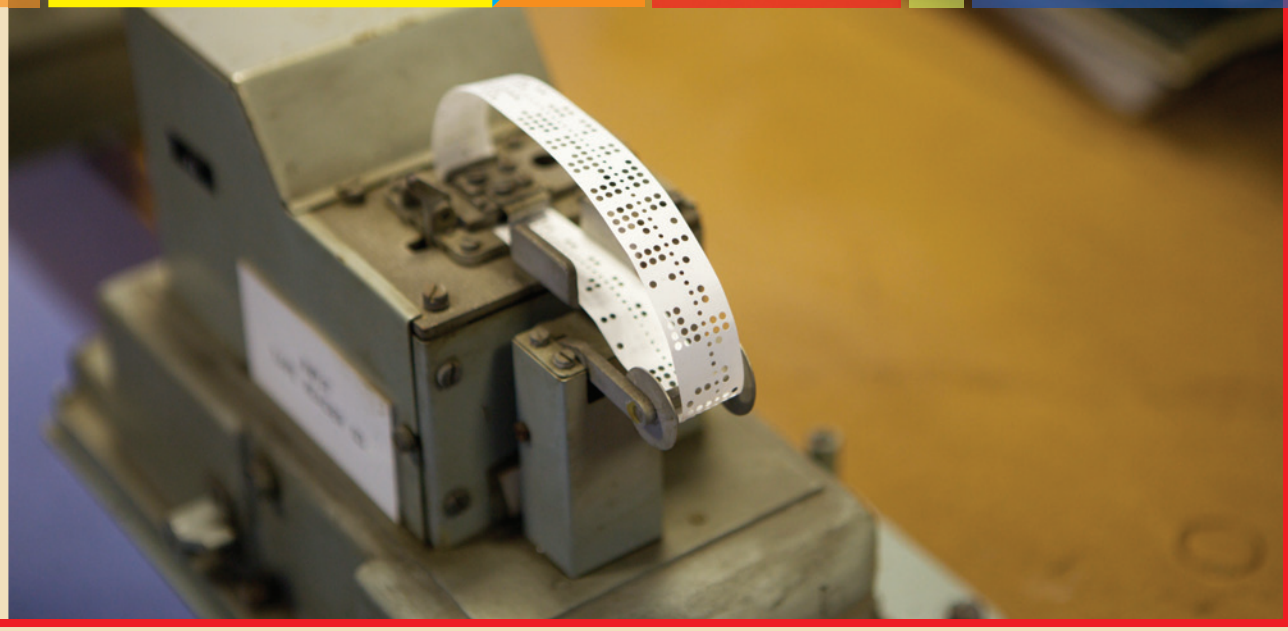
ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА
Английский язык – А.Громушкина
Библиотека в школе – О.Громова
Биология – Н.Иванова
География – О.Коротова
Дошкольное образование – Д.Тюттерин
Здоровье детей – Н.Сёмина
Информатика – С.Островский
Искусство – М.Сартан
История – А.Савельев
Классное руководство и воспитание школьников – М.Битянова
Литература – С.Волков
Математика – Л.Рослова
Начальная школа – М.Соловейчик
Немецкий язык – М.Бузоева
Русский язык – Л.Гончар
Спорт в школе – О.Леонтьева
Управление школой – Е.Рачевский
Физика – Н.Козлова
Французский язык – Г.Чесновицкая
Химия – О.Блохина
Школьный психолог – И.Вачков

УЧРЕДИТЕЛЬ: ООО “ЧИСТЫЕ ПРУДЫ”

Зарегистрировано ПИ № ФС77-44341 от 22.03.2011 в Министерстве РФ по делам печати
Подписано в печать: по графику 12.11.2012, фактически 12.11.2012
Заказ №
Отпечатано в ОАО “Первая Образцовая типография” Филиал “Чеховский Печатный Двор” ул. Полиграфистов, д. 1, Московская область, г. Чехов, 142300
Сайт www.chpk.ru,
E-mail: salas@chpk.ru,
факс 8 (496) 726-54-10,
8 (496) 988-63-87

АДРЕС ИЗДАТЕЛЯ: ул. Киевская, д. 24, Москва, 121165
Тел./факс: (499) 249-31-38
Отдел рекламы: (499) 249-98-70
http://1september.ru

ИЗДАТЕЛЬСКАЯ ПОДПИСКА:
Телефон: (499) 249-47-58
E-mail: podpiska@1september.ru



Удивительное рядом ☺

В конце года мне довелось оказаться на заседании школьного научного общества. Узнал много интересного. При случае обязательно постараюсь сходить еще. Некоторые факты поразительно освежают мозги

Мы привыкли, что компьютеры считают быстро. И многочисленные тексты про то, что первые компьютеры создавались как вычислительные и именно для быстрых вычислений, проглядываются по диагонали — это же всем известно, ничего нового. Оказывается, не все так просто.

В конце ноября в музее вычислительной техники в британском Блетчли-Парк запустили компьютер, возраст которого составляет более 60 лет. Это такая махина весом в две тонны, с релейными механизмами и памятью на газоразрядных лампах. Самое удивительное то, что ряд операций Harwell Dekatron — так называется компьютер — выполнял медленнее человека. Так на умножение двух чисел он мог затратить до 10 секунд. Человек — даже на листочке, а при помощи механических устройств и подавно — мог считать быстрее. Преимущество компьютера было в том, что он уставал меньше человека (когда уставал — просто ломался ☺) и был способен отработать многочасовую смену без перерывов на чай (английский). О самом компьютере я узнал еще много интересного, но по-настоящему поразил меня именно тот факт, что банальное утверждение про то, что компьютеры, даже первые, считали быстрее людей, оказывается, ложно.

Услышал я еще несколько очень любопытных мини-исследований ребят. Запомнилось, например, такое. Ребята проводили среди учеников старших классов в школе следующий опрос. Спрашивали, например: ты ведь знаешь, что такое контрастная фотография? Чаще всего получали ответ — да, конечно! А скажи — это был следующий вопрос — в чем именно на физическом/математическом/формальном уровне заключается эта самая контрастность? Интересно было, конечно, не то, как именно описываются эти свойства, — это можно за секунды найти в Интернете. Удивительно (и более всего это удивляло самих респондентов), что ребята в подавляющем большинстве случаев на самом деле не знали и не понимали, о чем идет речь, но уверенно отвечали “конечно, знаю!”.

Еще один замечательный доклад был посвящен технологиям проведения презентации. Ребята рассказали, что разным людям при подготовке к презентации удобно бывает следовать традициям разных школ актерского мастерства. Сильно обобщая и упрощая, школа Станиславского применительно к технологии проведения презентаций предполагает, что человек, “проигрывая” презентацию, видит себя “изнутри” — именно глазами выступающего. Тогда как школа Чехова предлагает посмотреть на себя со стороны — фактически из зала. И разным людям в процессе подготовки к презентации (да и к уроку, пожалуй) удобно пользоваться разными приемами (кто-то может и двумя, но большинство предпочитает какой-то один).

Я еще много всего интересного узнал. Но колонка закончилась ☺. До встречи!

Сергей Островский
(*so@1september.ru*), главный редактор

На фото: Элемент компьютера Harwell Dekatron

Сжатие данных: теория и практикум

К.Ю. Поляков,
д. т. н., Санкт-Петербург

Введение

Человеку всегда хочется, чтобы данные передавались мгновенно. Несмотря на развитие широкополосных линий связи, скорости все равно не хватает. Это чувствуется, например, при просмотре видеороликов высокого качества из Интернета — нередко сервера не справляются с нагрузкой, и поток данных прерывается или его скорость становится слишком низкой.

Для того чтобы ускорить передачу данных по существующей линии, нужно уменьшить длину сообщения, то есть количество передаваемых битов. Эта операция называется *сжатием* (или компрессией, упаковкой) данных. Сжатие — это один из способов кодирования, поэтому иногда используют выражение “сжимающее кодирование”.

Сжатие полезно еще и для экономии места на внешних носителях (винчестерах, флэш-дисках и т.п.), однако эта проблема постепенно отходит на второй план, поскольку цены на устройства внешней памяти постоянно снижаются.

Итак, при сжатии уменьшается длина сообщения. Существует два типа сжатия:

- *сжатие без потерь*, при котором исходные данные могут быть в точности восстановлены из сжатого сообщения;
- *сжатие с потерями*, при котором восстановленные данные могут отличаться от оригинала (данных, которые использовались при упаковке).

Неминуемо возникает два вопроса:

1) если при сжатии (без потерь) мы уменьшили длину битовой цепочки, как же мы можем восстановить потерянное;

2) зачем нужно сжатие с потерями, если оно приводит к потере информации?

Ответ на первый вопрос довольно прост: если после сжатия сообщение удается восстановить в исходном виде, значит, в нем была *избыточность*. Например, если из текста выкинуть все гласные (кроме начальной и конечной букв каждого слова), его, как правило, можно легко понять: “Влга впдт в Кспске мре”. Большой избыточностью обладает язык авиадиспетчеров, которые несколько раз повторяют каждую фразу и запрашивают подтверждение (“Как поняли?”) для того, чтобы надежно передать информацию при помехах.

Если избыточности нет (нет никаких закономерностей, нет “лишнего”), то и сжать данные без потерь не удается. Классический пример таких “несжимаемых” данных — последовательность случайных чисел.

Сжатие без потерь возможно тогда и только тогда, когда в данных есть **избыточность** (например, повторение одинаковых кодовых последовательностей).

Второй вопрос несколько сложнее. Действительно, если сжимать с потерями бухгалтерские документы, то ситуация может стать угрожающей :-). Однако в некоторых случаях мы все-таки можем пожертвовать частью информации ради уменьшения объема данных и увеличения скорости передачи. Например, хорошо известно, что средний человек слышит звуки с частотой от 20 Гц до 20 кГц, поэтому всю информацию, относящуюся к частотам вне этого диапазона, можно безболезненно “вырезать”, все равно мы ее не услышим. Такой подход используется для сжимающего кодирования звука во всех современных форматах звуковых файлов (в том числе в MP3, Ogg Vorbis, WMA, AAC). При этом удается уменьшить объем файла более чем в 10 раз. Аналогичная ситуация с фотографиями (формат JPEG) и видеофайлами (все современные форматы).

В этой статье мы рассмотрим подробно сжатие без потерь в объеме профильного курса информатики. Кроме теории, предлагаются практические работы, использующие оригинальное программное обеспечение — тренажеры для изучения различных алгоритмов сжатия.

Уменьшение длины равномерного кода

Пример 1. Пусть текстовый файл объемом 10 Кбайт содержит всего четыре различных символа: латинские буквы A, B, C и пробел. Для кодирования одного из четырех возможных вариантов достаточно двух битов, поэтому использовать для его передачи обычное 8-битное кодирование символов невыгодно. Можно присвоить каждому из четырех символов двухбитные коды, например, так:

A — 00, B — 01, C — 10, пробел — 11.

Тогда последовательность “ABA CABABA”, занимающая 10 байт в однобайтной кодировке, может быть представлена как цепочка из 20 бит:

00010011100001000100

Таким образом, нам удалось уменьшить информационный объем текста в 4 раза, и передаваться оно будет в 4 раза быстрее. Однако непонятно, как декодировать это сообщение, ведь получатель не знает, какой код был использован. Выход состоит в том, чтобы включить в сообщение служебную информацию — заголовок, в котором каждому коду будет сопоставлен код символа в кодовой таблице. Условимся, что первый байт заголовка — это количество используемых символов N, а следующие N байт — это ASCII-коды этих символов. В данном случае заголовок занимает 5 байт и выглядит так:

00000100_2	01000001_2	01000010_2	01000011_2	00100000_2
4 символа	A (код 65)	B (код 66)	C (код 67)	пробел (код 32)

Файл, занимающий 10 Кбайт в 8-битной кодировке, содержит 10 240 символов. В сжатом виде каждый символ кодируется двумя битами, кроме того, есть 5-байтный заголовок. Поэтому сжатый файл будет иметь объем

$$5 + 1024 \cdot 2/8 \text{ байт} = 2565 \text{ байт}.$$

В данном случае коэффициент сжатия (отношение объемов исходного и сжатого файлов) равен

$$k = 10\,240/2565 \approx 4,$$

то есть удалось сжать файл почти в 4 раза.

Если принимающая сторона “знает” формат файла (заголовок известной структуры + закодированные данные), она сможет восстановить в точности его исходный вид.

За счет чего удалось сжать файл? Только за счет того, что в файле использовались только 4 символа вместо полного набора, поэтому 8-битная кодировка была избыточна.

Алгоритм RLE

При сжатии данных, в которых есть цепочки одинаковых кодов, можно применять еще один простой алгоритм, который называется *кодированием цепочек одинаковых символов* (англ. *RLE = **R**un **L**ength **E**ncoding*).

Пример 2. Рассмотрим файл, в котором записаны сначала 100 русских букв А, а потом — 100 букв Б:

1		100	101		200
АА	.....	АА	ББ	.....	ББ

При сжатии с помощью алгоритма RLE сначала записывается количество повторений первого символа, затем сам первый символ, затем количество повторений второго символа и т.д. В данном случае весь закодированный файл занимает 4 байта:

01100100_2	11000000_2	01100100_2	11000001_2
100	А (код 192)	100	Б (код 193)

Таким образом, мы сжали файл в 50 раз за счет того, что в нем снова была *избыточность* — цепочки одинаковых символов. Это сжатие без потерь, потому что, зная алгоритм упаковки, можно восстановить исходные данные из кода.

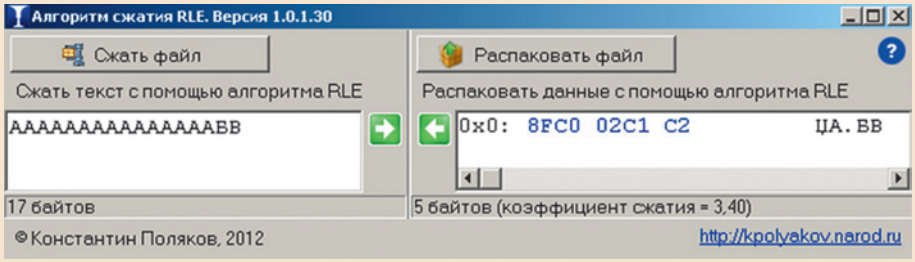
Очевидно, что такой подход будет приводить к *увеличению* (в 2 раза) объема данных в том случае, когда в файле нет соседних одинаковых символов. Чтобы улучшить результаты RLE-кодирования даже в этом наихудшем случае, алгоритм модифицировали следующим образом. Упакованная последовательность содержит управляющие байты, за каждым управляющим байтом следует один или несколько байтов данных. Если старший бит управляющего байта равен 1, то следующий за управляющим байт данных при распаковке нужно повторить столько раз, сколько записано в оставшихся 7 битах управляющего байта. Если же старший бит управляющего байта равен 0, то надо взять несколько следующих байтов данных без изменения. Сколько именно — записано в оставшихся 7 битах управляющего байта. Например, управляющий байт 10000111_2 говорит о том, что следующий за ним байт надо повторить 7 раз, а управляющий байт 00000100_2 — о том, что следующие за ним 4 байта надо взять без изменений. Так последовательность

10001111_2	11000000_2	00000010_2	11000001_2	11000010_2
повтор 15	А (код 192)	2	Б (код 193)	В (код 194)

распаковывается в 17 символов: АААААААААААААААББ.

Алгоритм RLE успешно применялся для сжатия рисунков, в которых большие области закрашены одним цветом, и некоторых звуковых данных. Сейчас вместо него применяют более совершенные, но более сложные методы. Один из них (кодирование Хаффмана) мы рассмотрим ниже. Алгоритм RLE используется, например, на одном из этапов кодирования рисунков в формате JPEG. Возможность RLE-сжатия есть также в формате BMP (для рисунков с палитрой 16 или 256 цветов).

Лучший способ понять принцип работы алгоритма — потренироваться в его использовании. С сайта автора (*http://kpolyakov.narod.ru/prog/compress.htm*) можно загрузить бесплатную программу-тренажер, которая предназначена для изучения алгоритма RLE:



В левой части окна программы находится текстовый редактор. При нажатии на кнопку  введенный текст сжимается с помощью алгоритма RLE, сжатые данные отображаются в виде шестнадцатеричных кодов в правой части окна. Окно в правой части — это тоже редактор, поэтому коды можно изменить и выполнить обратную операцию (распаковку, декомпрессию), нажав на кнопку  . Кнопки в верхней части окна позволяют сжимать и восстанавливать файлы на диске. Нужно только учитывать, что программа использует свой собственный формат хранения данных.

Контрольные вопросы

1) Оцените максимально достижимый коэффициент сжатия с помощью рассмотренного варианта RLE-алгоритма. В каком случае его удастся достичь?

2) Оцените коэффициент сжатия с помощью RLE-алгоритма в худшем случае. Опишите этот худший случай.

3) Придумайте три последовательности, которые невозможно сжать с помощью алгоритма RLE.

4) Постройте последовательности, которые сжимаются алгоритмом RLE ровно в 2 раза, в 4 раза, в 5 раз.

Практикум

1) Используя алгоритм RLE, закодируйте последовательность символов
ВВВВВВАССССАВВВВВВВ

Запишите результат в виде шестнадцатеричных кодов (каждый символ кодируется в виде байта, который представлен двумя шестнадцатеричными цифрами). Проверьте полученный результат с помощью программы RLE.

2) Декодируйте последовательность, упакованную с помощью алгоритма RLE (приводятся шестнадцатеричные коды): 01 4D 8E 41 01 4D 8E 41 16. Для определения символов по их шестнадцатеричным кодам используйте таблицу ASCII. Определите количество байтов в исходной и распакованной последовательности и вычислите коэффициент сжатия. Проверьте результат с помощью программы RLE. Предложите два способа проверки.

3) Используя программу RLE, примените RLE-сжатие к следующим файлам[1] и найдите для каждого из них коэффициент сжатия.

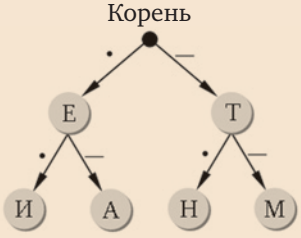
grad_vert.bmp *grad_horz.bmp* *grad_diag.jpg*

Объясните полученные результаты:

- почему для двух рисунков в формате BMP одинакового размера коэффициенты сжатия по алгоритму RLE так сильно отличаются;
- почему не удается сжать рисунки, сохраненные в формате JPEG?

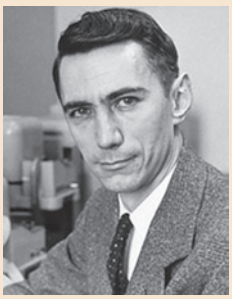
Префиксные коды

Вспомните азбуку Морзе, в которой для уменьшения длины сообщения используется неравномерный код — часто встречающиеся буквы (А, Е, М, Н, Т) кодируются короткими последовательностями, а редко встречающиеся — более длинными. Такой код можно представить в виде структуры, которая называется *деревом*:



Здесь показано неполное дерево кода Морзе, построенное только для символов, коды которых состоят из одного и двух знаков (точек и тире). Дерево состоит из узлов (черная точка и кружки с символами алфавита) и соединяющих их направленных ребер, стрелки указывают направление движения. Верхний узел (в который не входит ни одна стрелка) называется “корнем” дерева. Из корня и из всех промежуточных узлов (кроме конечных узлов — “листьев”) выходят две стрелки, левая помечена точкой, а правая — знаком “тире”. Чтобы найти код символа, нужно пройти по стрелкам от “корня” дерева к нужному “листу”, выписывая метки стрелок, по которым мы переходим. В дереве нет циклов (замкнутых путей), поэтому код каждого символа определяется единственным образом. По этому дереву можно построить такие коды:

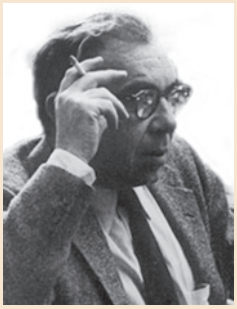
Е • И •• А • – Т – Н – • М – –



Клод Шеннон (1916–2001)

Это *неравномерный* код, в нем символы имеют коды разной длины. При этом всегда возникает проблема разделения последовательности на отдельные кодовые слова. В коде Морзе она решена с помощью символа-разделителя — паузы. Однако можно не вводить дополнительный символ, если выполняется *условие Фано*: ни одно из кодовых слов не является началом другого кодового слова. Это позволяет однозначно декодировать сообщение в реальном времени, по мере получения очередных символов.

Префиксный код — это код, в котором ни одно кодовое слово не является началом другого кодового слова (условие Фано).



Роберт Фано (род. 1917) (nytimes.com)

Для использования этой идеи в компьютерной обработке данных нужно было разработать алгоритм построения префиксного кода. Впервые эту задачу решили, независимо друг от друга, американские математики и инженеры Клод Шеннон (в 1948 году) и Роберт Фано (в 1949 году) [1–2]. Они использовали *избыточность* сообщений, состоящую в том, что символы в тексте имеют разные частоты встречаемости. В этом случае нужно читать данные исходного файла два раза: на первом проходе определяется частота встречаемости каждого символа, затем строится код с учетом этих данных, и на втором проходе символы текста заменяются на их коды. Алгоритм кодирования, предложенный Шенноном и Фано, получил название *кода Шеннона — Фано*.

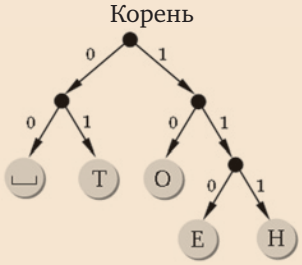
Пример 3. Пусть текст состоит только из букв О, Е, Н, Т и пробела. Известно, сколько раз они встретились в тексте: пробел — 179, О — 89, Е — 72, Н — 53 и Т — 50 раз.

Следуя методу Шеннона — Фано, делим символы на две группы так, чтобы общее количество найденных в тексте символов первой группы было примерно равно общему количеству символов второй группы. В нашем случае лучший вариант — это объединить пробел и букву Т в первую группу (сумма 179+50 = 229), а остальные символы — во вторую (сумма 89+72+53 = 214).

Символы первой группы будут иметь коды, начинающиеся с 0, а остальные — с 1. В первой группе всего два символа, у одного из них, например у пробела, вторая цифра кода будет 0 (и полный код 00), а у второго — 1 (код буквы Т — 01) .

[1] Эти и другие файлы, использующиеся в заданиях практикума, находятся на диске-приложении к этому номеру журнала.

Во второй группе три символа, поэтому продолжаем деление на две группы, примерно равные по количеству символов в тексте. В первую выделяем одну букву, которая чаще всего встречается, — это буква О (ее код будет 10), а во вторую — буквы Е и Н (они получают коды 110 и 111). Код Шеннона — Фано, построенный для этого случая, можно нарисовать в виде дерева:



Символ "␣" на этой схеме обозначает пробел. В данном случае получаются такие коды символов:

␣ — 00, T — 01, O — 10, E — 110, H — 111.

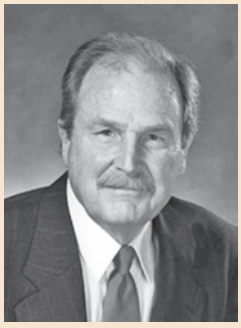
Легко проверить, что для этого набора кодов выполняется условие Фано. Это можно сразу определить по построенному дереву. В нем все символы располагаются в листьях, а не в промежуточных узлах. Это значит, что "по пути" от корня дерева до любого символа никаких других символов в промежуточных узлах не встречается (сравните с деревом кода Морзе).

Для декодирования очередного символа последовательности мы просто спускаемся от корня дерева, выбирая левую ветку, если очередной бит — 0, и правую, если этот бит равен 1. Дойдя до листа дерева, мы определяем символ, а затем снова начинаем с корня дерева, чтобы декодировать следующий символ, и т.д. Например, пусть получена последовательность

01100110001101111001.

Результат ее декодирования — "TOTO EHOT".

Алгоритм Хаффмана



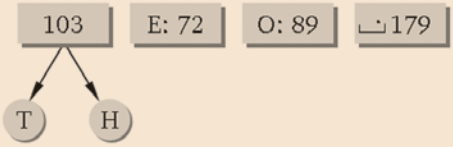
Дэвид Хаффман (1925–1999)

Было доказано, что в некоторых случаях кодирование Шеннона — Фано дает неоптимальное решение, и можно построить код, который еще больше уменьшит длину кодовой последовательности. Например, в предыдущем примере (код Шеннона — Фано) пробел и буква Т имеют коды одинаковой длины, хотя буква Т встречается в 3,5 раза реже пробела.

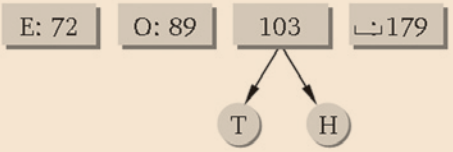
В 1952 году Дэвид Хаффман, ученик Фано, разработал новый алгоритм сжимающего кодирования и доказал его оптимальность среди алфавитных кодов [3–4].

Пример 3. Построим код Хаффмана по исходным данным, которые использовались в примере 3. Сначала отсортируем буквы по увеличению частоты встречаемости:

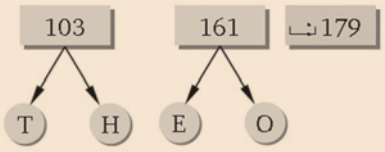
Затем берем две самых первых буквы, они становятся листьями дерева, а в узел, с которым они связаны, записываем сумму их частот:



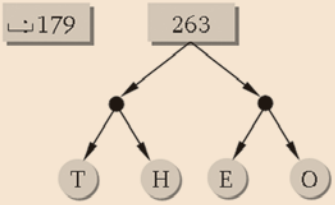
Таким образом, буквы, которые встречаются реже всего, получили самый длинный код. Дальше сортируем по возрастанию частоты в верхней строчке, причем для букв Т и Н используется их суммарная частота:



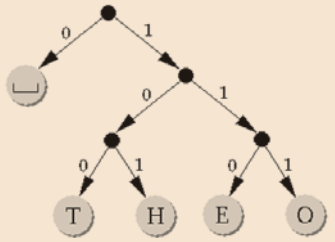
Повторяем ту же процедуру для букв Е и О, частоты которых оказались минимальны, и сортируем по возрастанию частот:



Теперь объединяем уже не отдельные буквы, а пары и снова сортируем:



На последнем шаге остается объединить символ "пробел" с деревом, которое построено для остальных символов. У каждой стрелки, идущей влево от какого-то узла, ставим код 0, а у каждой стрелки, идущей вправо, — код 1:



По этому дереву, спускаясь от корня к листьям-символам, получаем коды Хаффмана, обеспечивающие оптимальное сжатие с учетом частоты встречаемости символов:

␣ — 0, T — 100, O — 111, E — 110, H — 101.

Легко проверить, что этот код также удовлетворяет условию Фано.

Сравним эффективность рассмотренных выше методов. Для алфавита из пяти символов при равномерном кодировании нужно использовать 3 бита на каждый символ, так что общее число бит в сообщении равно (179 + 89 + 72 + 53 + 50)·3 = = 1329 бит. При кодировании методом Шеннона — Фано получаем 179·2 + 89·2 + 72·3 + 53·3 + + 50·2 = 1011 бит, коэффициент сжатия составляет 1329/1011 ≈ 1,31. Использование кода Хаффмана дает последовательность длиной 179·1 + 89·3 + + 72·3 + 53·3 + 50·3 = 971 бит и коэффициент сжатия 1,37. В сравнении со случаем, когда для передачи используется однобайтный код (8 бит на символ), выигрыш получается еще более весомым: кодирование Хаффмана сжимает данные примерно в 3,65 раза. Обратим внимание, что сжать данные удалось за счет *избыточности*: мы использовали тот факт, что некоторые символы встречаются чаще, а некоторые — реже.

Отметим, что, кроме сжатых данных, приёмнику нужно передать еще и дерево, использованное при кодировании.

Алгоритм кодирования Хаффмана и сейчас широко применяется благодаря своей простоте, высокой скорости кодирования и отсутствию патентных ограничений (он может быть использован свободно). Например, он применяется на некоторых этапах при сжатии рисунков (в формате JPEG) и звуковой информации (в формате MP3).

Тем не менее кодирование Хаффмана не лишено недостатков. Во-первых, нужно заранее знать вероятности появления всех символов, что на практике чаще всего невыполнимо. Во-вторых, код Хаффмана чувствителен к ошибкам распознавания в случае помех. Это значит, что один неверно распознанный бит приведет к тому, что вся оставшаяся часть сообщения будет декодирована неверно.

Теоретический предел сжатия

Префиксные коды Шеннона — Фано и Хаффмана используют посимвольное (алфавитное) кодирование. Это означает, что каждому символу исходного сообщения сопоставляется некоторый код, который зависит только от распределения вероятностей появления символов в сообщении. Каков же теоретический предел сжатия для этой группы методов и как определить его для некоторого потока данных?

На этот вопрос ответил Клод Шеннон. Он ввел в теорию передачи информации понятие *энтропии источника* данных, которая характеризует неопределенность наших знаний о нем [1,4]. Пусть алфавит сообщения содержит N различных символов и известны вероятности их появления p_i ($i = 1, \ldots, N$). Тогда энтропия источника вычисляется по формуле

$$H = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} = p_1 \log_2 \frac{1}{p_1} + \ldots + p_N \log_2 \frac{1}{p_N}$$

В работе [2] Шеннон показал, что

1) оптимальная длина кода для символа, имеющего вероятность появления p_i, равна $\log_2 \frac{1}{p_i}$;
2) для любого префиксного кода средняя длина кодовых слов не может быть меньше, чем энтропия H;
3) существует код, в котором средняя длина кодового слова отличается от энтропии не более, чем на 1 (этим свойством обладают коды Шеннона — Фано и Хаффмана).

Кодирование, основанное на использовании информации о вероятностях появления символов, называют *энтропийным кодированием*.

Пример 4. Предположим, что в сообщении встречаются только символы A, B и C, причем заранее известно, что букв A — 100, букв B — 50 и букв C — тоже 50. Нужно найти предельный коэффициент сжатия для этого файла.

Вычислим вероятности появления каждого символа:

$$p_A = \frac{100}{100+50+50} = 0{,}5;\ p_B = p_C = \frac{50}{200} = 0{,}25.$$

При этих данных найдем энтропию источника:

$$H = p_A \log_2 \frac{1}{p_A} + p_B \log_2 \frac{1}{p_B} + p_C \log_2 \frac{1}{p_C} =$$

$$= 0{,}5 \cdot 1 + 0{,}25 \cdot 2 + 0{,}25 \cdot 2 = 1{,}5 \text{ бита.}$$

Это означает, что средняя длина кодового слова не может быть меньше, чем 1,5 бита.

Длины оптимальных кодовых слов равны целым числам:

для буквы A: $L_A = \log_2 \frac{1}{p_A} = \log_2 \frac{1}{0{,}5} = 1$ бит

для букв B и C: $L_B = L_C = \log_2 \frac{1}{p_B} = \log_2 \frac{1}{0{,}25} = 2$ бита

При этом средняя длина кодового слова равна

$$L = \frac{100 \cdot 1 + 50 \cdot 2 + 50 \cdot 2}{200} = 1{,}5 \text{ бита,}$$

то есть мы получили оптимальный код, который обеспечивает теоретически предельное сжатие для алфавитных кодов.

Пример 5. Предположим, что в сообщении встречаются только символы A, B и C, причем заранее известно, что букв A — 120, букв B — 40 и букв C — тоже 40. Нужно найти предельный коэффициент сжатия для этого файла.

Вычислим вероятности появления каждого символа:

$$p_A = \frac{120}{120+40+40} = 0{,}6;\ p_B = p_C = \frac{40}{200} = 0{,}2$$

и соответствующие длины оптимальных кодовых слов:

для буквы A: $L_A = \log_2 \frac{1}{p_A} = \log_2 \frac{1}{0{,}6} \approx 0{,}74$ бита

для букв B и C: $L_B = L_C = \log_2 \frac{1}{p_B} = \log_2 \frac{1}{0{,}2} \approx 2{,}32$ бита

При алфавитном кодировании на каждый символ выделяется целое число бит. Оптимальный код дает алгоритм Хаффмана:

A — 0, B — 10, C — 11.

При этом средняя длина кодового слова равна

$$L = \frac{120 \cdot 1 + 40 \cdot 2 + 40 \cdot 2}{200} = 1{,}4 \text{ бита.}$$

Вычислим энтропию источника, которая определяет теоретический предел сжатия:

$$H = p_A \log_2 \frac{1}{p_A} + p_B \log_2 \frac{1}{p_B} + p_C \log_2 \frac{1}{p_C} \approx$$

$$\approx 0{,}6 \cdot 0{,}74 + 0{,}2 \cdot 2{,}32 + 0{,}2 \cdot 2{,}32 \approx 1{,}37 \text{ бита.}$$

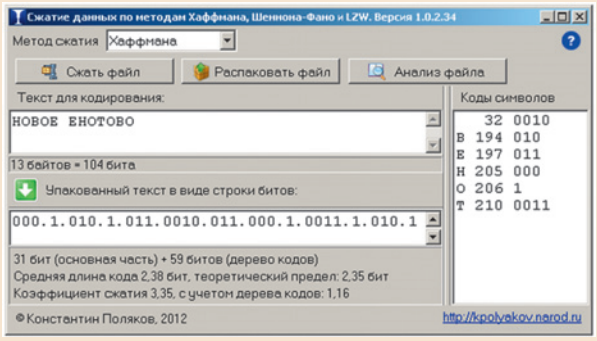
В данном случае предельного значения достигнуть не удалось.

Когда достижим теоретический предел? Очевидно, что тогда, когда для каждого символа оптимальная длина кодового слова, рассчитанная по формуле $L_i = \log_2 \frac{1}{p_i}$, — целое число.

Для этого необходимо, чтобы все вероятности p_i (i = 1, ..., N) были равны отрицательным степеням числа 2

$$(0{,}5 = \frac{1}{2};\ 0{,}25 = \frac{1}{2^2};\ 0{,}125 = \frac{1}{2^3};\ \ldots).$$

Для практического освоения сжатия данных с помощью префиксных кодов предназначена бесплатная программа-тренажер **Huffman**, которую можно скачать с сайта автора (*http://kpolyakov.narod.ru/prog/compress.htm*).



Программа позволяет сжимать текст, введенный во встроенном редакторе, а также файлы с помощью алгоритмов Шеннона — Фано, Хаффмана и LZW[2]. После щелчка по кнопке (клавиша F4) текст, введенный в текстовом редакторе, сжимается. Сжатые данные показываются в нижней части окна программы в виде строки, содержащей двоичный код. Для удобства отдельные кодовые слова в сообщении разделяются точками. В левой части выводится таблица кодов символов, использованная при сжатии.

Кнопки в верхней части окна предназначены для сжатия и распаковки файлов. Обратите внимание, что распаковываются только те файлы, которые сделаны в этой программе, поскольку она использует собственный формат хранения данных.

Кнопка "*Анализ файла*" позволяет определить для любого файла предельный теоретически достижимый коэффициент сжатия k_{max} при алфавитном кодировании (на основе вычисления энтропии).

Контрольные вопросы

1) Оцените максимальный коэффициент сжатия, который можно получить с помощью метода Хаффмана. Приведите пример файла, для которого достижим этот результат.

Практикум

1) С помощью программы **Huffman** закодируйте строку "ЕНОТ НЕ ТОНЕТ", используя методы Шеннона — Фано и Хаффмана. Запишите результаты в таблицу:

	Шеннон и Фано	Хаффман
Длина основного кода		
Длина кодовой таблицы (дерева)		
Коэффициент сжатия (по основным кодам)		
Коэффициент сжатия (с учетом дерева кодов)		

2) Как, по вашему мнению, будет изменяться коэффициент сжатия при увеличении длины текста, при условии, что набор символов и частота их встречаемости останутся неизменными? Проверьте ваш вывод с помощью программы (например, можно несколько раз скопировать одну и ту же фразу).

3) Повторите эксперимент с фразой "НОВОЕ ЕНОТОВО". Сделайте выводы.

4) Нарисуйте в тетради кодовые деревья, которые были построены программой при использовании обоих методов.

5) Используя кнопку "*Анализ файла*" в программе **Huffman**, определите предельный теоретический коэффициент сжатия для файла **a.txt**[3] при алфавитном кодировании.

6) С помощью программ **RLE** и **Huffman** выполните сжатие файла **a.txt** разными способами. Запишите результаты в таблицу:

	RLE	Шеннон и Фано	Хаффман
Размер сжатого файла			
Коэффициент сжатия			

Объясните результаты.

7) Используя кнопку "*Анализ файла*" в программе **Huffman**, определите предельный теоретический ко-

[2] Последний из этих алгоритмов мы рассмотрим чуть позже.

[3] Этот файл имеет объем 1 Мбайт и состоит из одних символов "А".

эффициент сжатия k_{max} для файла **a.txt.huf** (так будет называться результат сжатия файла **a.txt** по методу Хаффмана) при побайтном кодировании. Объясните результат.

8) Примените несколько раз повторное сжатие этого файла с помощью алгоритма Хаффмана (новые файлы получат имена **a.txt.huf2**, **a.txt.huf3** и т.д.) и заполните таблицу, каждый раз выполняя анализ полученного файла.

	Размер файла	k_{max}
a.txt		
a.txt.huf		
a.txt.huf2		
a.txt.huf3		
a.txt.huf4		
a.txt.huf5		

Объясните, почему с некоторого момента при очередном применении алгоритма сжатия размер файла увеличивается.

9) Выполните те же действия, используя метод Шеннона — Фано.

	Размер файла	k_{max}
a.txt		
a.txt.shf		
a.txt.shf2		
a.txt.shf3		
a.txt.shf4		
a.txt.shf5		

10) Сравните результаты сжатия этого файла с помощью алгоритма RLE, лучшие результаты, полученные методами Шеннона — Фано и Хаффмана, а также результат сжатия этого файла каким-нибудь архиватором. Объясните результаты и сделайте выводы.

Метод	Размер файла	k_{max}
RLE		
Хаффман		
Шеннон и Фано		
ZIP		
RAR		
7Z		

11) *Напишите программу, которая вычисляет предельный коэффициент сжатия для файла, имя которого вводится с клавиатуры. Сравните результаты работы вашей программы и программы **Huffman**.

12) *Предложите какой-либо вариант хранения в памяти (или в файле) дерева, использованного при кодировании Шеннона — Фано или Хаффмана.

Алгоритм LZW

Алфавитное кодирование, при котором каждый символ кодируется отдельно, учитывает только вероятности появления отдельных символов, но не их комбинаций. Поэтому иногда применение алгоритма RLE (учитывающего последовательности!) обеспечивает более высокую степень сжатия, чем алгоритм Хаффмана, оптимальный среди алфавитных кодов. Кроме того, на практике вероятности появления символов, как правило, неизвестны, поэтому применить кодирование Хаффмана для потока данных, передаваемого в реальном времени, нельзя. Таким образом, перед учеными встала задача создания универсального алгоритма кодирования, который

- не требует знания вероятностей появления отдельных символов;
- позволяет декодировать сообщение, не “заглядывая вперед”, то есть по мере получения данных в реальном времени;
- учитывает повторяющиеся последовательности символов;
- обладает помехоустойчивостью.

Такой алгоритм был разработан в 1977 году израильскими математиками А.Лемпелом и Я.Зивом [5] и в 1984 году усовершенствован Т.Велчем [6]. Он получил название LZW-алгоритма[4] по первым буквам фамилий авторов.

Идея кодирования состоит в том, что последовательностям символов присваиваются числовые коды, которые в конечном счете кодируются в виде битовых цепочек. Как только в сообщении встретилась цепочка, которой еще не было, ей присваивается очередной код, и пара “цепочка — код” заносится в словарь. Таким образом, словарь составляется прямо во время кодирования. В отличие от алгоритмов Хаффмана и Шеннона — Фано, для работы алгоритма LZW не требуется никаких данных о частоте встречаемости символов, поэтому кодирование выполняется за один проход.

Для кодирования используются две переменные, которые мы назовем **СТРОКА** и **СИМВОЛ**. В переменной **СТРОКА** хранится текущая последовательность, а в переменной **СИМВОЛ** — последний символ, полученный от источника. Получая символы из входного потока, мы добавляем их к текущей строке, пока не получим строку, которой еще нет в словаре. Тогда эта строка добавляется в словарь, а накопление продолжается с последнего введенного символа. На алгоритмическом языке алгоритм LZW-сжатия можно сформулировать так:

```
ввод СТРОКА
нц пока не конец данных
  ввод СИМВОЛ
  если Есть_в_словаре(СТРОКА + СИМВОЛ) то
    СТРОКА := СТРОКА + СИМВОЛ
  иначе
    вывод Код(СТРОКА)
    Добавить_в_словарь(СТРОКА + СИМВОЛ)
    СТРОКА := СИМВОЛ
  всё
кц
вывод Код(СТРОКА)
```

[4] Кроме LZW, существует еще несколько алгоритмов, основанных на идеях работы А.Лемпела и Я.Зива 1977 года. Наиболее известный из них — LZMA, который используется в архиваторе 7zip.

Строка	Символ	Строка+символ	Есть в словаре?	В словарь	Вывод
Т					
Т О	О	ТО	Нет	4 = ТО	Код(Т) = 0
О Т	Т	ОТ	Нет	5 = ОТ	Код(О) = 1
Т	О	ТО	Да		
ТО Е	Е	ТОЕ	Нет	6 = ТОЕ	Код(ТО) = 4
Е Н	Н	ЕН	Нет	7 = ЕН	Код(Е) = 2
Н О	О	НО	Нет	8 = НО	Код(Н) = 3
О	Т	ОТ	Да		
ОТ	Конец				Код(ОТ) = 5

Здесь используются три подпрограммы:

• функция **Код** — возвращает целое число — код строки в словаре;

• функция **Есть_в_словаре** — возвращает логическое значение “истина”, если строка есть в словаре, и “ложь”, если ее там нет;

• процедура **Добавить_в_словарь** — добавляет строку в словарь и присваивает ей первый свободный код.

Для примера закодируем с помощью алгоритма LZW сообщение

ТОТОЕНОТ

Здесь всего 4 разных символа, поэтому сначала заносим в словарь четыре пары “код – строка”: 0 = Т, 1 = О, 2 = Е, 3 = Н. Дальнейшее выполнение приведенного алгоритма показано в таблице (см. выше).

Итак, закодированное сообщение состоит из шести чисел: “0 1 4 2 3 5”. Можно было бы использовать равномерный трехбитный код, но в алгоритме LZW длина кодового слова определяется размером словаря, даже если какие-то записи в нем не используются. Поскольку в словаре 9 записей (с 0 по 8), код будет четырехбитным[5]:

0000 0001 0100 0010 0011 0101

общая длина закодированного сообщения 24 бита. При 8-битном кодировании сообщение из восьми букв имеет длину 64 бита, поэтому коэффициент сжатия равен

$$k = \frac{64}{24} \approx 2{,}67.$$

Заметим, что если использовать двухбитное равномерное кодирование (Т — 00, О — 01, Е — 10, Н — 11), длина сообщения будет равна 16 битам; в сравнении с этим вариантом алгоритм LZW не дал выигрыша. Однако ситуация будет значительно улучшаться по мере увеличения длины сообщения и заполнения словаря.

Теперь посмотрим, как декодировать сообщения, сжатые с помощью LZW. Серьезное достоинство алгоритма состоит в том, что не нужно хранить полный словарь: он строится в процессе декодирования. Нужны только коды отдельных букв.

```
ввод СТАРЫЙ_КОД
вывод Получить_строку(СТАРЫЙ_КОД)
нц пока не конец данных
  ввод НОВЫЙ_КОД
  СТРОКА := Получить_строку(НОВЫЙ_КОД)
  вывод СТРОКА
  СИМВОЛ := СТРОКА[1]
  СТРОКА := Получить_строку(СТАРЫЙ_КОД)
  Добавить_в_словарь(СТРОКА + СИМВОЛ)
  СТАРЫЙ_КОД := НОВЫЙ_КОД
кц
```

Здесь используются три подпрограммы:

• функция **Получить_строку** — возвращает символьную строку, соответствующую заданному коду;

• функция **Есть_в_словаре** — возвращает логическое значение “истина”, если заданный код есть в словаре, и “ложь”, если его там нет;

• процедура **Добавить_в_словарь** — добавляет строку в словарь и присваивает ей первый свободный код.

Итак, у нас есть коды букв (0 = Т, 1 = О, 2 = Е, 3 = Н) и числовая последовательность “0 1 4 2 3 5”. Сразу заносим в словарь коды букв, таким образом, в словаре уже есть четыре записи.

Старый код	Новый код	Строка	Символ	В словарь	Вывод
0					Т
0	1	О Т	О	4 = ТО	О
1	4	ТО О	Т	5 = ОТ	ТО
4	2	Е ТО	Е	6 = ТОЕ	Е
2	3	Н Е	Н	7 = ЕН	Н
3	5	ОТ Н	О	8 = НО	ОТ

[5] Часто используют код переменной длины — длина кодовых слов увеличивается по мере роста размера словаря.

Строка	Символ	Строка+символ	Есть в словаре?	В словарь	Вывод
A					
A A	A	AA	Нет	1 = AA	Код(A) = 0
A	A	AA	Да		
AA A	A	AAA	Нет	2 = AAA	Код(AA) = 1
A	A	AA	Да		
AA	A	AAA	Да		
AAA	Конец				Код(AAA) = 2

Таким образом, на выходе получили “ТОТОЕНОТ”.

Но в приведенном алгоритме декодирования есть одна проблема, которая проявляется, например, при кодировании цепочки “АААААА”. Используя алгоритм LZW-кодирования, получаем последовательность кодов “0 1 2” со словарем

0 = A, 1 = AA, 2 = AAA.

Выше представлена таблица, описывающая процесс кодирования.

Можно заметить, что сразу после занесения в словарь цепочки АА (код 1) она снова встретилась в тексте. То же самое относится и к цепочке ААА.

Попробуем декодировать сообщение “0 1 2” по приведенному выше алгоритму. Сразу после стартового кода 0 мы получаем код 1, но записи с кодом 1 еще нет в словаре! Это значит, что слово встретилось сразу же после того, как было занесено в словарь. Однако из алгоритма кодирования следует, что в этом случае занесенное в словарь слово должно заканчиваться на ту же букву, на которую начинается! Поскольку предыдущая буква исходного сообщения — это А (код 0), получаем, что слово с кодом 1 — это АА. При использовании такого подхода алгоритм немного усложняется, но зато теперь работает в любых ситуациях:

```
ввод СТАРЫЙ_КОД
вывод Получить_строку(СТАРЫЙ_КОД)
СИМВОЛ := Получить_строку(СТАРЫЙ_КОД)
нц пока не конец данных
  ввод НОВЫЙ_КОД
  если Есть_в_словаре(НОВЫЙ_КОД) то
    СТРОКА := Получить_строку(НОВЫЙ_КОД)
  иначе
   СТРОКА := Получить_строку(СТАРЫЙ_КОД)
   СТРОКА := СТРОКА + СИМВОЛ
  все
  вывод СТРОКА
  СИМВОЛ := СТРОКА[1]
  СТРОКА := Получить_строку(СТАРЫЙ_КОД)
  Добавить_в_словарь(СТРОКА + СИМВОЛ)
  СТАРЫЙ_КОД := НОВЫЙ_КОД
кц
```

Наверное, читатель уже понял, что при LZW-кодировании нужно как-то передавать приёмнику начальный словарь, содержащий коды отдельных символов. Однако можно этого и не делать, если договориться, что первые 256 элементов словаря — это байты с кодами 0–255.

В практических реализациях часто используют 12-битные коды (как в оригинальной статье Т.Велча [6]), что позволяет строить словарь из $2^{12} = 4096$ записей. Когда словарь переполняется, можно, например:

- очистить словарь (сохранив только коды символов) и начать строить его заново;
- при добавлении нового слова удалять последовательность, которая никогда не использовалась или использовалась реже всего.

Алгоритм LZW применяют для сжатия рисунков в форматах GIF и TIFF. Простейшая реализация на языке Си приведена, например, в [7]. Соответствующие демонстрационные программы на школьном алгоритмическом языке можно найти на диске в приложении к этому номеру.

Практикум

1) Примените алгоритм LZW к сообщению “ПАНАМАМАМА”. Считая, что каждый код записывается с помощью одинакового минимально возможного числа бит, определите коэффициент сжатия данных в сравнении с 8-битной кодировкой. Как изменится результат, если учесть начальный словарь, который нужно передать приёмнику?

2) Декодируйте сообщение “0 1 0 2 5 3”. Известно, что начальный словарь, содержащий коды всех используемых символов, выглядит так:

0 — Т, 1 — А, 2 — И.

Считая, что каждый код записывается с помощью одинакового минимально возможного числа бит, определите коэффициент сжатия данных в сравнении с 8-битной кодировкой.

3) Рассмотрим сообщения, состоящие из *N* идущих подряд последовательностей АВ. Используя программу **Huffman**, определите для различных *N* коэффициенты сжатия при использовании алгоритмов Хаффмана и LZW:

N	Хаффман	LZW
1		
5		
10		
50		
100		

Объясните результаты и сделайте выводы.

4) С помощью программ **RLE** и **Huffman** определите для каждого из перечисленных в таблице файлов:

• предельный коэффициент сжатия k_{max} при алфавитном кодировании;

• коэффициенты сжатия при использовании методов RLE, Хаффмана и LZW.

Как вы можете объяснить полученные результаты, учитывая содержание этих файлов? Почему в ряде случаев алгоритмы RLE и LZW обеспечивают более высокий коэффициент сжатия, чем можно достичь при алфавитном кодировании?

	k_{max}	Хаффман	RLE	LZW
`a.txt`				
`ab.txt`[6]				
`grad_vert.bmp`				
`grad_horz.bmp`				
`grad_diag.jpg`				

5) *Напишите программу, которая выполняет LZW-кодирование и декодирование символьных строк.

6) *Напишите программу, которая выполняет LZW-кодирование и декодирование произвольных двоичных файлов.

Заключение

В этой статье рассмотрены вопросы сжимающего кодирования данных без потерь в объеме профильного курса информатики в средней школе. Кроме теоретических сведений, предлагаются практические работы, в которых используется разработанное автором бесплатное программное обеспечение.

Нужно заметить, что на практике не всегда имеет смысл добиваться предельной степени сжатия и полностью устранять “избыточность” в передаваемых данных. В реальных каналах связи всегда действуют помехи, поэтому приходится специально вводить избыточность для того, чтобы обнаружить и исправить ошибки в процессе передачи. Но это уже “совсем другая история” ☺.

[6] Этот файл имеет объем 1 Мбайт и состоит из цепочек “AB”.

Литература

1. *Шеннон К.* Работы по теории информации и кибернетике. М.: Издательство иностранной литературы, 1963.

2. *Яглом А.М., Яглом И.М.* Вероятность и информация. М.: Комкнига, 2007.

3. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ. М.: Вильямс, 2006.

4. *Вернер М.* Основы кодирования. М.: Техносфера, 2004.

5. *Ziv J., Lempel A.* Compression of Individual Sequences Via Variable-Rate Coding // IEEE Transactions on Information Theory, September 1978, p. 530–536 (*http://www.cs.duke.edu/courses/spring03/cps296.5/papers/ziv_lempel_1978_variable-rate.pdf*).

6. *Welch T.* A Technique for High-Performance Data Compression // IEEE Computer, June 1984, p. 8–19 (*http://www.informatik.uni-freiburg.de/~ipr/teaching/ss_05/lzw.pdf*).

7. Метод LZW-сжатия данных. Электронный ресурс: *http://algolist.manual.ru/compress/standard/lzw.php*.

Автор благодарит д. ф.-м. н. М.А. Ройтберга за полезные замечания по содержанию статьи.

В Интернет — на “машине времени”

Читайте на с. 47

MimioClassroom

Интегрированная система интерактивного обучения

Доступно, просто в использовании, интересно для учеников

MimioBoard
Новая стационарная интерактивная доска

MimioTeach
Интерактивная приставка к классной доске

MimioVote
Создание и проведение тестов и контрольных

MimioView
Документ–камера с переходником для микроскопа

MimioPad
Ведение урока из любой точки класса

MimioStudio
Универсальное ПО для управления всем оборудовнием

Интегрированная система интерактивного обучения MimioClassroom позволяет принципиально изменить методологию преподавания. Педагоги получают возможность обеспечить активное и заинтересованное участие каждого ученика во всем, что происходит на уроке. Дидактический материал становится ярче и нагляднее. Учителя значительно экономят свое время и силы как при подготовке уроков, так и при обработке проверочных работ и тестов. Результат внедрения Mimio-Classroom — существенное повышение эффективности работы педагогов, уровня и качества знаний учеников. При этом оборудование и программное обеспечение Mimio зарекомендовало себя простым и удобным в эксплуатации, не требующим больших затрат времени на освоение.

Продажа оборудования, консультации и обучение:

http://www.mimioclass.ru

8 (800) 5555-33-0

Звонок по России бесплатный

ООО «Рене» — генеральный дистрибьютор Mimio в России

mimio
a better way to learn

Еще раз про однозначное декодирование

К.Ю. Поляков,
д. т. н., Санкт-Петербург

Введение

В последние годы в заданиях КИМ ЕГЭ по информатике, как в демоверсиях, так и в реальных вариантах, неизменно присутствует задача на кодирование данных следующего типа [1, задание А9]:

Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, используется неравномерный двоичный код, позволяющий однозначно декодировать полученную двоичную последовательность. Вот этот код: А — 00, Б — 01, В — 100, Г — 101, Д — 110. Можно ли сократить для одной из букв длину кодового слова так, чтобы код по-прежнему можно было декодировать однозначно? Коды остальных букв меняться не должны. Выберите правильный вариант ответа:

1) для буквы Д — 11;
2) это невозможно;
3) для буквы Г — 10;
4) для буквы Д — 10.

Как показывает практика, эта задача вызывает серьезные трудности не только у многих учеников, но даже у учителей информатики.

Нужно сказать, что этот материал практически не рассматривается в существующих школьных учебниках информатики, поэтому все (как ученики, так и учителя) вынуждены разбираться самостоятельно. В то же время вузовские учебники [2–5], где соответствующая теория изложена строго и научно, достаточно сложны для понимания. Попробуем разобраться в сути кодирования и декодирования на школьном уровне, то есть так, как можно объяснить ученикам 8–11-х классов.

В чем проблема?

Предположим, нам нужно передать сообщение по цифровым каналам связи. Для этого его необходимо закодировать, то есть сопоставить каждому символу исходного сообщения некоторый код (кодовое слово). Для определенности будем использовать двоичные коды, то есть последовательности нулей и единиц.

Пример 1. Пусть для кодирования фразы “МАМА МЫЛА ЛАМУ” выбран такой код:

М	А	Ы	Л	У	пробел
00	1	01	0	10	11

(1)

Коды букв “сцепляются” в одну битовую строку и передаются, например, по сети:

МАМА МЫЛА ЛАМУ → 00100111000101110100 10

Эта цепочка битов приходит в пункт назначения, и тут возникает проблема — как восстановить исходное сообщение (конечно, при условии, что мы знаем код, то есть знаем все пары “символ — кодовое слово”, которые использовались при кодировании).

Итак, мы получили 00100111000101110100 10. Легко понять, что при использовании кода (1) раскодировать такое сообщение можно самыми разными способами. Например, можно предположить, что оно составлено только из букв А (код 1) и Л (код 0). Тогда получаем ЛЛАЛЛАААЛЛЛАЛАААЛАЛЛАЛ. В общем, ни мамы, ни ламы.

Определение. Код называется ***однозначно декодируемым****, если любое кодовое сообщение можно расшифровать единственным способом (однозначно).*

Сказанное выше означает, что код (1) НЕ является однозначно декодируемым. Как же определить, является ли заданный код однозначно декодируемым? Этим вопросом мы и займемся.

Равномерные коды

Проблема состоит в том, чтобы правильно разбить полученную битовую цепочку на отдельные кодовые слова. Для того чтобы ее решить, можно, например, использовать *равномерный код*, то есть код, в котором все кодовые слова имеют одинаковую длину. Например, в нашей фразе 6 символов, поэтому можно использовать 3-битный код (который позволяет закодировать $8 = 2^3$ различных символов).

Пример 2. Закодируем фразу из примера 1, используя код:

М	А	Ы	Л	У	пробел
000	001	010	011	100	101

(2)

Получаем закодированное сообщение

МАМА МЫЛА ЛАМУ →

000001000001101000010011001101011001000100

Длина этого сообщения — 42 бита вместо 22 в предыдущем варианте, зато его легко разбить на отдельные кодовые слова и раскодировать (␣ обозначает пробел):

000	001	000	001	101	000	010	011	001	101	011	001	000	100
М	А	М	А	␣	М	Ы	Л	А	␣	Л	А	М	У

Видим, что равномерные коды неэкономичны (закодированное сообщение в примере 2 почти в два раза длиннее, чем в примере 1), но зато раскодируются однозначно.

Неравномерные коды

Для того чтобы сократить длину сообщения, можно попробовать применить *неравномерный код*, то есть код, в котором кодовые слова, соответствующие разным символам исходного алфавита, могут иметь разную длину.

Пример 3. Используем для кодирования фразы из примера 1 следующий код:

М	А	Ы	Л	У	пробел
01	00	1011	100	1010	11

(3)

Получаем

МАМА МЫЛА ЛАМУ →

0100010011011011100001110000011010

Здесь 34 бита. Это, конечно, не 22, но и не 42.

Несложно показать, что эта битовая цепочка декодируется однозначно. Действительно, первая буква — М (код 01), потому что ни одно другое кодовое слово не начинается с 01. Аналогично определяем, что вторая буква — А. Действительно, за 01 следует 00 (код буквы А), и никакое другое кодовое слово не начинается с 00. Это же свойство, которое называется условием Фано, выполняется не только для кодовых слов 01 и 00, но и кодовых слов всех других букв (проверьте это самостоятельно).

Условие Фано. *Никакое кодовое слово не совпадает с началом другого кодового слова.*

Коды, для которых выполняется условие Фано, называют *префиксными*[1]. Все сообщения, закодированные с помощью префиксных кодов, декодируются однозначно.

Префиксные коды имеют важное практическое значение — они позволяют декодировать символы полученного сообщения по мере его получения, не дожидаясь, пока все сообщение будет доставлено получателю.

Упражнение. Расшифруйте сообщение, закодированное кодом (3). При расшифровке кода очередной буквы не заглядывайте вперед!

10000011010110100010011011011100 00

Термины “условие Фано” и “префиксный код” не используются в заданиях ЕГЭ и ГИА, однако для решения этих задач важно, чтобы ученики понимали *содержание* условия Фано.

Пример 4. Рассмотрим еще один код

М	А	Ы	Л	У	пробел
10	00	1101	001	0101	11

(4)

Ясно, что он не является префиксным: код буквы А (00) совпадает с началом кода буквы Л (001) и код пробела (11) совпадает с началом кода буквы Ы (11).

[1] Префикс слова — это его начальный фрагмент. Условие Фано можно переформулировать так: “Никакое кодовое слово не совпадает с префиксом другого кодового слова”.

Закодированное сообщение

МАМА МЫЛА ЛАМУ →

10001000111011010010011001001001 01

также имеет длину 34 бита, как и при использовании кода (3). Начнем раскодировать с начала. Ясно, что первой стоит буква М, потому что ни один другой код не начинается с 10. Затем — комбинация 001, которая может быть кодом буквы Л или кодом буквы А (00), за которым следует код буквы Ы или пробела. Получается, что для декодирования сообщения нам нужно “заглядывать вперед”, что очень неудобно.

Попробуем раскодировать с конца битовой строки. Последние биты 0101 могут представлять только букву У, следующие 10 — только букву М и т.д. Можно проверить, что теперь сообщение однозначно декодируется *с конца*! Это происходит потому, что выполняется условие, которое можно назвать *“обратным” условием Фано*: никакое кодовое слово не совпадает с окончанием другого кодового слова. Коды, для которых выполняется обратное условие Фано, называют *постфиксными*[2]. В этом случае тоже обеспечивается однозначное декодирование. Таким образом,

Сообщение декодируется однозначно, если для используемого кода выполняется прямое или обратное условие Фано.

Обратим внимание на то, что условие Фано (и обратное условие Фано) — это *достаточное, но не необходимое* условие однозначной декодируемости. Это значит, что

• для однозначной декодируемости достаточно выполнения хотя бы одного из двух условий, или прямого, или обратного;

• могут существовать коды, для которых не выполняется ни прямое, ни обратное условие Фано, но они тем не менее обеспечивают однозначное декодирование.

Наиболее интересен второй вывод, который сразу вызывает два вопроса: 1) что это за коды? и 2) как в общем случае установить, что код декодируется однозначно? Обсудим это в следующем разделе.

Однозначно декодируемые коды[3]

Пример 5. Рассмотрим код, предназначенный для кодирования сообщений, состоящих только из букв А, Б и В:

А	Б	В
0	11	010

(5)

Так как код буквы А (0) совпадает как с началом, так и с концом кода буквы В (010), для этого кода не выполняются ни прямое, ни обратное условия Фано. Поэтому пока мы не можем с уверенностью сказать, декодируется ли он однозначно.

Закодируем сообщение

АББААВВА → 01111000100100

и попытаемся раскодировать эту строку, используя код (5). В первую очередь замечаем, что две соседние единицы могут появиться только при использовании буквы Б (код 11), поэтому сразу выделяем две таких группы:

0ББ000100100

Здесь серым фоном выделена уже декодированная часть сообщения. В оставшейся части единица может появиться только в коде буквы В (010), в битовой строке находим две такие группы:

0ББ00ВВ0

Оставшиеся нули — это коды букв А. Анализ алгоритма показывает, что такой код всегда однозначно декодируется.

Полный ответ на вопрос об однозначной декодируемости получил в начале 1960-х годов советский математик Ал.А. Марков (*http://www.vmk.unn.ru/?id=1266*), предложивший решение с помощью графов [2]. Продемонстрируем его метод на примере.

Пример 6. Рассмотрим код

А	Б	В	Г	Д
01	010	011	11	101

(6)

Здесь не выполняется ни “прямое”, ни “обратное” условие Фано, поэтому возможно, что декодировать сообщение однозначно не удастся. Но утверждать это заранее нельзя.

Следуя методу Ал.А. Маркова, построим граф следующим образом:

1) Определяем все последовательности (строки), которые

а) совпадают с началом какого-то кодового слова и одновременно с концом какого-то кодового слова и

б) сами не являются кодовыми словами.

В данном случае это две последовательности:

0 (начало кода буквы А и конец кода буквы Б) и

1 (начало кода буквы Г и конец кода буквы Д);

последовательности 01 и 11 не учитываем, потому что они совпадают с кодами букв А и Г;

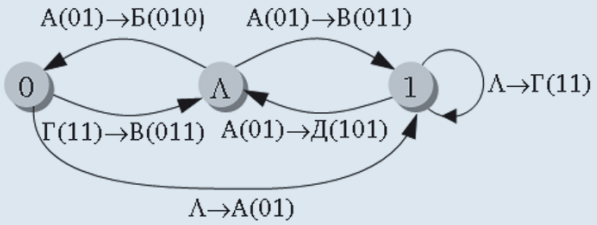
2) Добавляем к этому множеству {0, 1} пустую строку, которую обычно обозначают греческой буквой Λ; элементы полученного множества {Λ, 0, 1} будут вершинами графа:

3) Соединяем вершины дугами (направленными ребрами) по такому правилу: две вершины X и Y соединяются дугой, если последовательная запись кода вершины X, кода некоторой буквы (или нескольких букв) и кода вершины Y дает код еще одной буквы (см. *рис.* на с. 19).

[2] Постфикс (или *суффикс*) слова — это его конечный фрагмент.

[3] Материал этого раздела не требуется для сдачи ЕГЭ и ГИА. Но, по мнению автора, он может быть интересен учителям и заинтересованным ученикам.

Надпись W→Z на дуге, ведущей из X в Y, означает, что если между словами X и Y вставить последовательность букв W (она может состоять из нескольких символов), то полученная цепочка кодов совпадет с кодом буквы Z. Например, последовательная запись пустой строки (Λ), кода буквы А (01) и цепочки 0 дает цепочку 010 (код буквы Б); поэтому рисуем дугу из вершины "Λ" в вершину "0" и у этой дуги пишем "А→Б", и т.д. Поскольку код буквы Г можно записать как 11 = 1Λ1, у вершины "1" появляется петля "Λ→Г".



Результат Ал.А. Маркова состоит в следующем:

Код является однозначно декодируемым тогда и только тогда, когда в построенном таким образом графе нет ориентированных циклов, включающих вершину "Λ".

Иначе говоря,

1) если есть сообщение, которое декодируется неоднозначно, то в графе есть цикл, проходящий через вершину Λ;

2) если в графе есть цикл, проходящий через вершину Λ, то можно построить сообщение, которое декодируется неоднозначно; для этого нужно, проходя по циклу, последовательно выписывать коды всех встречающихся вершин и дуг.

В приложении к этому номеру приводится программа на языке Python 3, которая строит граф Ал.А. Маркова, обнаруживает в нем циклы, проходящие через вершину Λ, и определяет соответствующие им кодовые последовательности, которые декодируются неоднозначно.

В нашем графе есть три таких цикла:

- цикл "Λ0Λ", соответствующий сообщению ΛА0ГΛ = 01011; это сообщение может быть расшифровано как АВ и как БГ;
- цикл "Λ1Λ", соответствующий сообщению ΛА1АΛ = 01101; это сообщение может быть расшифровано как АД и как ВА;
- цикл "Λ01Λ", соответствующий сообщению ΛА01АΛ = 010101; это сообщение может быть расшифровано как ААА и как БД.

Кроме того, у вершины "1" есть петля, которая дает более длинные циклы "из Λ в Λ". Действительно, проходя по стрелкам из вершины Λ, мы приходим в вершину 1, там "вертимся" какое-то количество раз и возвращаемся обратно в вершину Λ, замыкая цикл. Поэтому неоднозначно декодируется любая последовательность вида 01…101, где многоточие обозначает любое количество единиц. Например, сообщение 0111101 может быть декодировано как АГД или ВГА.

Таким образом, код (6) не обладает свойством однозначной декодируемости.

Проверим таким же способом код (5), который, как мы уже выяснили, не является ни префиксным, ни постфиксным. Множество последовательностей, которые совпадают с началом и концом кодовых слов, состоит из пустой строки и единицы: {Λ, 1}. Граф, построенный с помощью приведенного выше алгоритма, содержит два узла и одну петлю:

В этом графе нет цикла, содержащего вершину "Λ", поэтому любое сообщение, записанное с помощью такого кода, декодируется однозначно. Выше мы показали это с помощью простых рассуждений.

Нужно отметить, что на практике применяются главным образом префиксные коды, поскольку они позволяют декодировать сообщение по мере его получения, не дожидаясь окончания приема данных.

Еще примеры

Пример 7. *Рассмотрим задачу А9 из демоварианта КИМ ЕГЭ-2013* [1]*, которая сформулирована в начале статьи. Нужно оптимизировать код*

А — 00, Б — 01, В — 100, Г — 101, Д — 110,

выбрав один из вариантов:

1) для буквы Д — 11; 3) для буквы Г — 10;

2) это невозможно; 4) для буквы Д — 10.

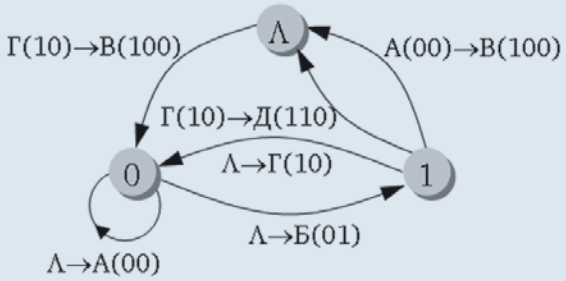
Решение. Сначала давайте посмотрим на исходный код, приведенный в условии. Можно заметить, что он префиксный — для него выполняется условие Фано: ни один из трехбитных кодов не начинается ни с 00 (код А), ни с 01 (код Б). Поэтому сообщения, закодированные с помощью такого кода, декодируются однозначно.

Заметим, что "обратное" условие Фано не выполняется: код буквы А (00) совпадает с окончанием кода буквы В (100), а код буквы Б (01) совпадает с окончанием кода буквы Г (101).

Теперь проверим, что получится, если сократить код буквы Д до 11 (вариант 1). Свойство однозначной декодируемости может быть потеряно только тогда, когда в результате такого сокращения нарушится условие Фано, то есть код буквы Д совпадет с началом какого-то другого кодового слова. Видим, что этого не произошло — нет других кодовых слов, которые начинаются с 11, поэтому вариант 1 — это и есть верное решение.

Остается убедиться, что варианты 3 и 4 не подходят. Если мы сократим код буквы Г до 10 (вариант 3), условие Фано оказывается нарушенным, так как теперь код буквы Г (10) совпал с началом кода буквы В (100). Одновременно нарушено и "обратное" условие Фано: код буквы А (00) совпадает с окончанием кода буквы В (100). Но, как мы знаем, при этом код может все-таки быть однозначно декодируемым.

Конечно, можно построить граф, как было сделано выше, и проверить, есть ли в нем циклы, включающие вершину Л. В данном случае граф выглядит так:



Здесь два цикла, проходящих через вершину Л (не считая петли у вершины 0):

• цикл, соответствующий сообщению ЛГ0Л1АЛ = = 100100; это сообщение может быть расшифровано как ГБА или как ВВ;

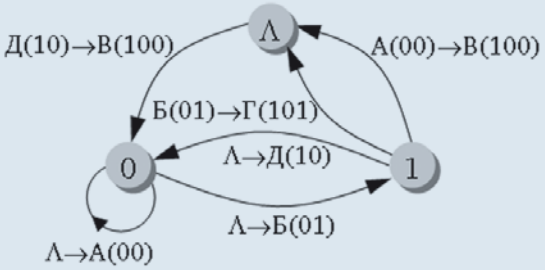
• цикл, соответствующий сообщению ЛГ0Л1ГЛ = = 100110; это сообщение может быть расшифровано как ГБГ или как ВД.

Таким образом, вариант 3 не обеспечивает однозначную декодируемость сообщений.

Построение и анализ графа — дело достаточно трудоемкое и требующее аккуратности. Обычно в таких случаях значительно легче просто подобрать последовательность, которая может быть декодирована двумя разными способами.

Наконец, нужно убедиться, что вариант 4 не удовлетворяет условию. Если мы сократим код буквы Д до 10, условие Фано оказывается нарушенным, так как теперь код буквы Д (10) совпал с началом кода буквы В (100). Как и раньше, нарушено “обратное” условие Фано: код буквы А (00) совпадает с окончанием кода буквы В (100) и код буквы Б (01) совпадает с окончанием кода буквы Г (101).

Построим граф по методу Ал.А. Маркова:



Здесь два цикла, проходящих через вершину Л (не считая петли у вершины 0):

• цикл, соответствующий сообщению ЛД0Л1АЛ = = 100100; это сообщение может быть расшифровано как ДБА или как ВВ;

• цикл, соответствующий сообщению ЛД0Л1БЛ = = 100101; это сообщение может быть расшифровано как ДББ или как ВГ.

Таким образом, вариант 4 также не соответствует условию. Поэтому правильный ответ — 1.

Пример 8. *Оптимизируйте код*

А — 11, Б — 10, В — 011, Г — 000, Д — 001, сохранив свойство однозначной декодируемости сообщений. Выберите один из вариантов:

1) для буквы Г — 00; 3) для буквы В — 01;

2) это невозможно; 4) для буквы Б — 1.

Решение. Определим, за счет чего обеспечивается однозначная декодируемость исходного кода. Легко видеть, что код префиксный — для него выполняется условие Фано: ни одно из трехбитовых кодовых слов не начинается ни с 11 (код А), ни с 10 (код Б). В то же время, обратное условие Фано не выполняется, потому что код буквы А (11) совпадает с окончанием кода буквы В (011).

Проверим вариант 1 — сократим код буквы Г до 00. При этом нарушилось условие Фано, которое обеспечивало однозначную декодируемость исходного варианта: теперь код буквы Г (00) совпадает с началом кода буквы Д (001). Но и обратное условие Фано тоже не выполняется для пары букв А–В. Поэтому можно предположить, что такой код не обладает свойством однозначной декодируемости. И действительно, легко находится цепочка 001011, которую можно раскодировать как ГБА (00 10 11) или ДВ (001 011).

Рассмотрим вариант 3 — сократим код буквы В до 01. При этом условие Фано выполняется, поскольку ни одно из кодовых слов не начинается с 01, то есть код является префиксным и однозначно раскодируется. Это и есть правильный ответ.

На всякий случай проверяем вариант 4 — сокращаем код буквы Б до 1. При этом код перестает быть префиксным, и обратное условие Фано также не выполнено (код буквы Б совпадает с началом и концом кода буквы А). Сразу понятно, что последовательность 11 можно раскодировать как А или как ББ, поэтому этот вариант неверный.

Выводы

В заметке выполнен подробный анализ задачи на кодирование, которая предлагается на ЕГЭ в последние несколько лет. Нужно заметить, что в нем затрагивается вузовский курс дискретной математики. Понятно, что нельзя требовать от школьников знания теорем Ал.А. Маркова об однозначном декодировании, но учителю полезно более глубоко представлять себе эти вопросы, которые можно разбирать на факультативах. В качестве дополнительной литературы по этой теме можно рекомендовать [3–5].

С точки зрения практического подхода для решения всех известных автору реальных задач подобного типа достаточно найти вариант, при котором выполняется условие Фано или обратное условие Фано (одно из двух!).

Литература

1. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2013 года по информатике и ИКТ. *http://www.fipi.ru/binaries/1384/inf11.zip*.

2. *Марков Ал.А.* Введение в теорию кодирования. М.: Наука, 1982.

3. *Яблонский С.В.* Введение в дискретную математику. М.: Наука, 2000.

4. *Жильцова Л.П.* Современные проблемы теории кодирования. Учебно-методические материалы по программе повышения квалификации “Информационные технологии и компьютерное моделирование в прикладной математике”. Нижний Новгород: 2007. *http://www.unn.ru/pages/e-library/aids/2007/6.pdf*.

5. *Жильцова Л.П., Смирнова Т.Г.* Основы теории графов и теории кодирования в примерах и задачах: Учебное пособие. Нижний Новгород: Издательство Нижегородского госуниверситета, 2008. *http://window.edu.ru/resource/606/73606/files/mliva01.pdf*.

Автор благодарит М.А. Ройтберга за полезные замечания по материалу статьи и Л.Н. Евич за обсуждение вопросов однозначного декодирования на форуме http://egekp.unoforum.ru/.

Педагогический университет
«Первое сентября»
предлагает для учителя информатики

Лицензия Департамента образования г. Москвы 77 № 000349, рег. № 027477 от 15.09.2010

Дистанционные курсы повышения квалификации

вне зависимости от места проживания
(обучение с 1 января по 30 сентября 2013 года)

Имеются два варианта учебных материалов дистанционных курсов: брошюры и брошюры+DVD.
Курсы, включающие видеолекции (DVD), помечены значком
Нормативный срок освоения каждого курса – 72 часа.
Дополнительная информация – на сайте edu.1september.ru
Окончившие дистанционные курсы получают удостоверение установленного образца.
Базовая стоимость курса (без учета скидок) составляет
2190 руб. – для курсов без видеоподдержки,
2390 руб. – для курсов с видеоподдержкой.

Профильные курсы

код	
07-001	*И.Г. Семакин*. Информационные системы в базовом и профильном курсах информатики
07-008	*А.Г. Гейн*. Математические основы информатики
07-009	*С.Л. Островский*. Основы web-программирования для школьного «сайтостроительства»
07-010	*А.Г. Кушниренко, А.Г. Леонов*. Методика преподавания основ алгоритмизации на базе системы «Кумир»

Общепедагогические курсы

код	
21-001	*С.С. Степанов*. Теория и практика педагогического общения
21-002	*Н.У. Заиченко*. Методы профилактики и разрешения конфликтных ситуаций в образовательной среде
21-003	*С.Н. Чистякова, Н.Ф. Родичев*. Образовательно-профессиональное самоопределение школьников в предпрофильной подготовке и профильном обучении
21-004	*М.Ю. Чибисова*. Психолого-педагогическая подготовка школьников к сдаче выпускных экзаменов в традиционной форме и в форме ЕГЭ
21-005	*М.А. Ступницкая*. Новые педагогические технологии: организация и содержание проектной деятельности учащихся
21-007	*А.Г. Гейн*. Информационно-методическое обеспечение профессиональной деятельности педагога, педагога-психолога, работника школьной библиотеки
21-008	*А.Н. Майоров*. Основы теории и практики разработки тестов для оценки знаний школьников
21-009	*В.Д. Шадриков, И.В. Кузнецова, М.Д. Кузнецова*. Формирование и оценка профессиональных качеств современного педагога

Очные курсы повышения квалификации

для жителей Москвы и Московской области
(обучение с 11 февраля по 30 апреля 2013 года)

Нормативный срок освоения каждого курса – 72 часа.
Дополнительная информация – на сайте edu.1september.ru
и по телефону (499) 240-02-24 (звонки принимаются с 15.00 до 19.00).
Окончившие очные курсы получают удостоверение государственного образца.
Базовая стоимость курса (без учета скидки) – 5900 руб.

Я.Н. Зайдельман. Алгоритмизация и программирование: от первых шагов до подготовки к ЕГЭ
И.Б. Полякова. Современные фото- и видеотехнологии в деятельности учителя

Электронную заявку можно в режиме on-line подать на сайте
edu.1september.ru

Сколько единиц и сколько нулей?

Д.М. Златопольский,
Москва

В демонстрационном варианте контрольных измерительных материалов апробации в 2012 году единого государственного экзамена по информатике и ИКТ в компьютерной форме приведена следующая задача B7:

Сколько единиц содержится в двоичной записи результата выражения:

$$(2 \cdot 10_8)^{2010} - 4^{2011} + 2^{2012}?$$

Как показало обсуждение задачи в тематической секции "Информатика и ИКТ в компьютерной форме" на сайте ФИПИ (*http://www.fipi.ru/CEGE/rt1.php?ts=1*), она вызвала большой интерес, при этом у ряда учителей возникли вопросы по методике решения этой задачи. Обсудим такую методику.

Но сначала решим две вспомогательные задачи.

Вспомогательная задача 1. Найдите количество нулей n и количество единиц e в двоичной записи результата выражения: $2^5 - 2^2$.

Решение

Здесь можно найти результат непосредственными вычислениями:

$$\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ - & & & 1 & 0 & 0 \\ \hline & 1 & 1 & 1 & 0 & 0 \end{array}$$

Ответ: $n = 2$, $e = 3$.

Вспомогательная задача 2. Найдите количество нулей n и количество единиц e в двоичной записи результата выражения: $2^{100} - 2^{25}$.

Решение

В данном случае проводить непосредственные вычисления с двоичными числами нецелесообразно. Попробуем найти формулы для нахождения искомых величин для общего случая — применительно к разности $2^a - 2^b$ (при $a > b$).

В двоичном виде уменьшаемое является $(a + 1)$-значным числом, а разность будет a-значной. Нетрудно увидеть (в том числе и из решения вспомогательной задачи 1), что в конце этой разности будут записаны b нулей. А это означает, что количество единиц e равно $a - b$.

Ответ: $e = 75$, $n = 25$.

Примечание. Задачу можно решить проще, если выражение в условии записать следующим образом: $(2^{100} - 1) - (2^{25} - 1)$. Учитывая, что $(2^k - 1)$ — это $(k + 1)$-значное двоичное число, состоящее из одних единиц, то для разности двух выражений в скобках ответ получить нетрудно: $n = 25$, $e = 75$.

Вернемся к "основной" задаче.

Представим первые два члена заданного в условии выражения в виде степеней двойки:

$$10_8 = 8_{10} = 2^3,$$

поэтому $(2 \cdot 10_8)^{2010} = (2^4)^{2010} = 2^{8040}$.

$$4^{2011} = 2^{4022}.$$

Тогда заданное выражение примет вид:

$$2^{8040} - 2^{4022} + 2^{2012}.$$

В двоичной записи разности будет 4018 единиц и 4022 нуля, причем именно в таком порядке (см. решение вспомогательных задач). Добавление числа 2^{2012} (представляющего из себя единицу и 2012 нулей — см. таблицу ниже) увеличит количество единиц на 1. Итак, ответ: 4019.

k	Десятичное число $n = 2^k$	Двоичная запись числа n	Количество единиц	Количество нулей	Общее количество цифр
0	1	1	1	0	1
1	2	10	1	1	2
2	4	100	1	2	3
3	8	1000	1	3	4
4	16	10000	1	4	5
…	…	…	…	…	…

Общероссийский проект

Школа цифрового века

Интернет-сопровождение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

2012/13
учебный год

Предметно-методические материалы

Дистанционные модульные курсы

Бесплатно, адресно
каждому учителю!

Участие образовательного учреждения в проекте «Школа цифрового века» в 2012/13 учебном году позволяет каждому педагогическому работнику получать в свой Личный кабинет на сайте www.1september.ru предметно-методические журналы Издательского дома «Первое сентября» и проходить дистанционные модульные курсы по Программе развития профессионально-личностных компетенций педагога.

Заявки принимаются от образовательных учреждений.

Оргвзнос за участие в проекте в течение всего 2012/13 учебного года – 4 тысячи рублей.

Величина оргвзноса не зависит от количества педагогических работников в образовательном учреждении.

Педагогическим работникам образовательного учреждения предоставляются документы, подтверждающие участие в проекте.

Прием заявок от образовательных учреждений

на сайте

digital.1september.ru

до 31 декабря 2012 года

Общероссийский проект «Школа цифрового века» по комплексному обеспечению образовательных учреждений методической интернет-поддержкой разработан в соответствии с Федеральной целевой программой развития образования на 2011–2015 годы и направлен на развитие инновационного потенциала образовательных учреждений: вовлечение педагогических работников в цифровое образовательное пространство, повышение эффективности использования современных образовательных технологий (в том числе информационно-коммуникационных технологий) в профессиональной деятельности

Решение систем однородных логических уравнений

С.М. Авдошин, руководитель отделения программной инженерии,

Р.З. Ахметсафина, доцент, Национальный исследовательский университет "Высшая школа экономики"

Введение

Рассмотрим решение заданий на определение количества решений логического уравнения от N переменных вида

$$F(X_1, X_2, X_3, \ldots, X_N) = 1$$

или системы K логических уравнений от N переменных вида

$$F_1(X_1, X_2, X_3, \ldots, X_N) = 1$$
$$F_2(X_1, X_2, X_3, \ldots, X_N) = 1$$
$$\ldots$$
$$F_K(X_1, X_2, X_3, \ldots, X_N) = 1$$

Заметим, что система логических уравнений такого вида может быть сведена к логическому уравнению

$$F_1(X_1, X_2, X_3, \ldots, X_N) \wedge$$
$$\wedge F_2(X_1, X_2, X_3, \ldots, X_N) \wedge \ldots \wedge$$
$$\wedge F_K(X_1, X_2, X_3, \ldots, X_N) = 1$$

конъюнкцией всех левых частей логических уравнений системы.

Обратное также возможно — логическое уравнение, содержащее в левой части логическое выражение, состоящее из конъюнкций, может быть записано как система логических уравнений.

Система логических уравнений может быть записана в эквивалентной форме, содержащей другое количество логических уравнений.

Например, заданная система уравнений

$$(X_1 \to X_2) \wedge (X_2 \to X_3) \wedge (X_1 \to X_3) = 1$$
$$(X_2 \to X_3) \wedge (X_3 \to X_4) \wedge (X_2 \to X_4) = 1$$
$$(X_3 \to X_4) \wedge (X_4 \to X_5) \wedge (X_3 \to X_5) = 1$$

эквивалентна уравнению

$$(X_1 \to X_2) \wedge (X_2 \to X_3) \wedge (X_1 \to X_3) \wedge$$
$$\wedge (X_3 \to X_4) \wedge (X_2 \to X_4) \wedge (X_4 \to X_5) \wedge$$
$$\wedge (X_3 \to X_5) = 1$$

или системе двух уравнений

$$(X_1 \to X_2) \wedge (X_2 \to X_3) \wedge (X_3 \to X_4) \wedge$$
$$\wedge (X_4 \to X_5) = 1$$
$$(X_1 \to X_3) \wedge (X_2 \to X_4) \wedge (X_3 \to X_5) = 1$$

Подобные задания можно решать разными методами, например:

1) Методом рассуждений. Применяется для решения несложных уравнений [7, 8].

2) Построением таблицы истинности логического выражения — применяется при небольшом количестве переменных, так как количество строк таблицы истинности для N переменных равно 2^N [1, 4–8].

3) Построением дерева решений. Этот метод удобно применять, когда уравнение можно решать, последовательно добавляя логические переменные [7, 8]. Каждый уровень дерева соответствует одной логической переменной. Вершины помечаются логическими значениями 0 или 1. Пусть левая вершина соответствует значению 0, правая — значению 1. Для каждой вершины определяем, имеет ли смысл продолжать из нее построение. Если нет, вершина становится терминальной, помечаем ее цветом и перечеркиваем. Количество листьев дерева соответствует количеству решений, наборы значений переменных определяются последовательными значениями вершин на пути от корня дерева к листу. Как правило, листья находятся на последнем уровне дерева.

4) Построением СДНФ заданного выражения. Количество полных элементарных конъюнкций соответствует количеству решений [1, 4].

5) В некоторых частных случаях можно использовать особенности заданных выражений и избежать построения полной таблицы истинности, дерева или СДНФ. При решении используются метод включения — исключения и методы определения количества путей в графах [8].

В ряде случаев целесообразно ввести замену переменных, решить задание, а затем вернуться к исходным переменным [4–8].

Все перечисленные методы решения позволяют определить не только количество решений, но и наборы логических переменных, на которых уравнение (система уравнений) имеет решение.

Для решения логических уравнений вида

$$F(X_1, X_2, X_3, \ldots, X_N) = 0$$

можно использовать три подхода:

1. Построить СКНФ, количество сомножителей — элементарных полных дизъюнкций — равно количеству нулевых значений функции в таблице истинности, т.е. количеству решений уравнения [1, 4].

2. Решить уравнение вида $F(X_1, X_2, X_3, \ldots, X_N) = 1$. Количество решений исходного уравнения равно общему количеству наборов 2^N минус количество наборов, на которых функция равна единице.

3. Добавить отрицание к обеим частям уравнения, получим

$$\neg F(X_1, X_2, X_3, \ldots, X_N) = 1.$$

Для решения таких уравнений используются описанные выше методы.

Следует заметить, что при решении систем логических уравнений вида

$$F_1(X_1, X_2, X_3, \ldots, X_N) = 0$$
$$F_2(X_1, X_2, X_3, \ldots, X_N) = 1$$
$$\ldots$$
$$F_K(X_1, X_2, X_3, \ldots, X_N) = 0$$

нужно заменить уравнение

$$F_i(X_1, X_2, X_3, \ldots, X_N) = 0$$

на эквивалентное уравнение

$$\neg F_i(X_1, X_2, X_3, \ldots, X_N) = 1$$

и решать задачу одним из пяти способов.

Большое количество примеров решения логических уравнений и систем уравнений, иллюстрирующих методы 1–5, приводится в [3–8]. В данной работе описан метод решения систем однородных логических уравнений [8].

Метод решения систем однородных логических уравнений

Существуют системы логических уравнений, которые невозможно или сложно решить предложенными в [3–7] методами. Большое количество подобных заданий приводится в [2].

Предлагаемый метод можно применить в случае, если уравнения, входящие в систему, имеют одинаковую структуру и различаются лишь индексами переменных. Такие уравнения называются **однородными**, их можно записать в общем виде. В пары соседних уравнений входит постоянное число одинаковых переменных.

Для использования предлагаемого метода необходимо знать некоторые сведения о матрицах и операциях умножения матриц и векторов, рекуррентных формулах, способах описания графов, методах определения количества путей в графах, которые приведены в Приложениях 1–3.

Кроме того, при решении систем уравнений используются двудольные и многодольные графы, которые рассмотрены ниже.

Двудольные графы

Двудольный граф — это граф, множество вершин которого можно разбить на две части (доли) таким образом, что каждое ребро графа соединяет какую-то вершину из одной части с какой-то вершиной другой части, то есть не существует ребра, соединяющего две вершины из одной и той же части.

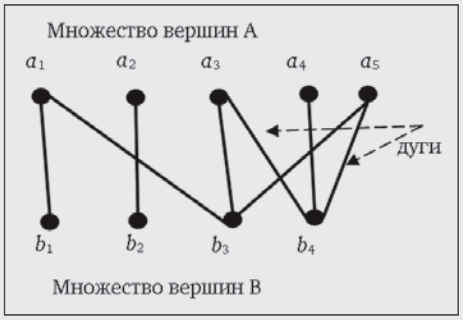


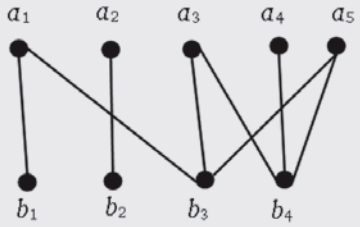
Рис. 1. Пример двудольного графа

Для описания двудольного графа используют матрицу смежности двудольного графа, которую далее для краткости будем называть **матрицей двудольного графа**.

В матрице двудольного графа количество строк равно количеству вершин в одной доле, а количество столбцов — количеству вершин в другой доле графа. Таким образом, размерность матрицы двудольного графа меньше, чем размерность матрицы смежности. На пересечении строк и столбцов мат-

рицы двудольного графа записывается единица, если соответствующая пара вершин связана ребром, и ноль, если вершины не связаны. Если две вершины связаны m ребрами, в матрице может быть записано число m.

Графическая форма



Матрица двудольного графа

M	b_1	b_2	b_3	b_4
a_1	1	0	1	0
a_2	0	1	0	0
a_3	0	0	1	1
a_4	0	0	0	1
a_5	0	0	1	1

Рис. 2. Формы представления двудольного графа

Множество вершин $\{a_i\}$ графа на *рис.* 2 может обозначать сотрудников, множество вершин $\{b_i\}$ — работы или проекты. Граф показывает, какие работы выполняет каждый сотрудник.

Общее количество ребер между долями графа определяется суммой элементов матрицы двудольного графа. Для графа, представленного на *рис.* 2, количество ребер равно восьми.

Для определения суммы элементов матрицы этого графа надо

1) умножить единичную вектор-строку из пяти элементов (по количеству вершин a) на матрицу двудольного графа

$$(11111)\times\begin{pmatrix}1010\\0100\\0011\\0001\\0011\end{pmatrix}=(1133)$$

2) полученную вектор-строку умножить на единичный вектор-столбец из четырех элементов (по количеству вершин b)

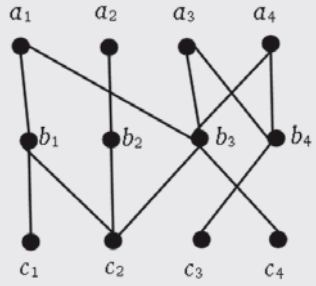
$$(1133)\times\begin{pmatrix}1\\1\\1\\1\end{pmatrix}=1\cdot1+1\cdot1+3\cdot1+3\cdot1=8$$

Полученное значение равно количеству ребер двудольного графа.

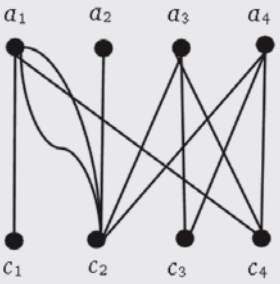
Заметим, что результатом умножения вектора-строки на единичный вектор-столбец является сумма элементов вектора-строки.

Многодольные графы

Рассмотрим граф, состоящий из последовательности двудольных графов. Такой граф называют многодольным графом порядка t, где t — количество долей. На *рис.* 3*а* показан трехдольный граф ($t = 3$), который состоит из последовательности двух двудольных графов.



а)



б)

Рис. 3. Многодольный граф

Матрица первого двудольного графа имеет вид

$$M1=\begin{pmatrix}1&0&1&0\\0&1&0&0\\0&0&1&1\\0&0&1&1\end{pmatrix}$$

Матрица второго двудольного графа:

$$M2=\begin{pmatrix}1&1&0&0\\0&1&0&0\\0&1&0&1\\0&0&1&0\end{pmatrix}$$

Определим количество путей из вершин первой доли (множества $\{a_i\}$) в вершины третьей доли (множества $\{c_i\}$) для этого графа.

1) Для определения **количества путей из каждой вершины первой доли в каждую вершину третьей доли** надо перемножить матрицы двудольных графов $M1$ и $M2$. Элементы полученной матрицы $M = M1 \times M2$ соответствуют количеству путей из вершины a_i в вершину c_j, где i — номер строки, j — номер столбца матрицы M.

$$M=M1\times M2=\begin{pmatrix}1&0&1&0\\0&1&0&0\\0&0&1&1\\0&0&1&1\end{pmatrix}\times\begin{pmatrix}1&1&0&0\\0&1&0&0\\0&1&0&1\\0&0&1&0\end{pmatrix}=$$

$$=\begin{pmatrix}1&2&0&1\\0&1&0&0\\0&1&1&1\\0&1&1&1\end{pmatrix}$$

Таким образом, из вершины a_1 в вершину c_1 ведет один путь, в вершину c_2 — два пути, в вершину c_3 нет путей, в вершину c_4 — один путь и т.д.

Матрица M — это также матрица двудольного графа, в котором первая доля — вершины множества $\{a_i\}$, вторая доля — вершины множества $\{c_i\}$. При таком описании внутренние вершины многодольного графа "скрыты", мы "свернули" многодольный граф в двудольный (*рис. 3б*).

2) Для определения **количества путей из всех вершин первой доли в каждую вершину третьей доли** надо умножить единичную вектор-строку размерности 4 на матрицу M. В полученном векторе-строке i-й элемент соответствует общему количеству путей из первой доли в вершину c_i.

$$(1111)\times\begin{pmatrix}1&2&0&1\\0&1&0&0\\0&1&1&1\\0&1&1&1\end{pmatrix}=(1523)$$

В многодольном графе в вершину c_1 ведет один путь, в вершину c_2 — пять путей и т.д. (*рис. 3б*).

3) Для определения **количества путей из всех вершин первой доли во все вершины третьей доли** надо умножить вектор-строку, полученную на шаге 2, на единичный вектор-столбец размерности 4. Иными словами, надо просуммировать все элементы вектора-строки, полученные на предыдущем шаге. Количество всех путей из первой доли графа в последнюю долю равно:

$$1 + 5 + 2 + 3 = 11$$

Однородный многодольный граф

Рассмотрим последовательность K одинаковых двудольных графов, имеющих, соответственно, одинаковые матрицы двудольных графов M. В этом случае количество вершин Q во всех долях должно быть одинаковым. Определим количество путей.

Введем обозначения:

K — количество одинаковых двудольных графов, составляющих многодольный граф порядка $K + 1$;

Q — количество вершин в каждой доле;

F_0 — единичная вектор-строка размерности Q;

F_K — вектор-строка размерности Q, элементами которой являются количества путей из первой доли в каждую вершину K-й доли;

I — единичный вектор-столбец размерности Q;

R — общее количество путей из всех вершин первой доли многодольного графа во все вершины K-й доли.

Тогда можно записать:

$$R = F_0 \times M^K \times I$$

Или в виде рекуррентной формулы

$$F_i = F_{i-1} \times M,\ i = 1, 2, \ldots, K$$

$$R = F_K \times I$$

Далее рассмотрим предлагаемый метод на примере. Будем везде считать, что K — количество уравнений, N — количество логических переменных.

Пример 1 [2]

Сколько существует различных наборов значений логических переменных x_1, x_2, …, x_9, x_{10}, которые удовлетворяют всем перечисленным ниже условиям?

$$x_1 \vee x_2 \wedge x_3 = 1$$
$$x_2 \vee x_3 \wedge x_4 = 1$$
$$\ldots$$
$$x_7 \vee x_8 \wedge x_9 = 1$$
$$x_8 \vee x_9 \wedge x_{10} = 1$$

В ответе не нужно перечислять все различные наборы значений x_1, x_2, …, x_9, x_{10}, при которых выполнена данная система равенств. В качестве ответа нужно указать количество таких наборов.

Решение: Система уравнений однородная: все уравнения имеют одинаковую структуру. Пары соседних уравнений связаны двумя общими переменными. Например, для первого и второго уравнений общими являются переменные x_2, x_3, для второго и третьего — пары x_3, x_4 и т.д.

Решим систему уравнений последовательно, добавляя очередное уравнение к системе на каждом шаге. Сначала найдем решение первого уравнения, затем первых двух уравнений, первых трех и т.д.

Для решения первого уравнения составим таблицу истинности.

x_1	x_2	x_3	$x_2 \wedge x_3$	$x_1 \vee x_2 \wedge x_3$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Первое уравнение имеет пять решений.

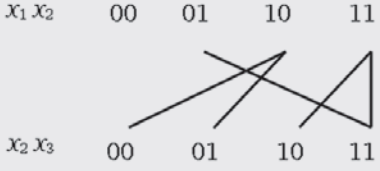
Рассмотрим первое и второе уравнения.

$$x_1 \vee x_2 \wedge x_3 = 1$$
$$x_2 \vee x_3 \wedge x_4 = 1$$

Во втором уравнении записаны переменные x_2, x_3 на тех же позициях, что переменные x_1, x_2 в первом уравнении.

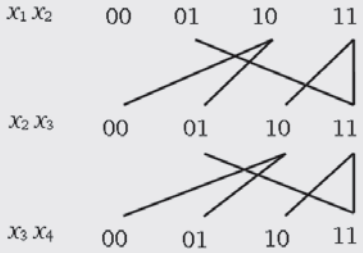
Переменные x_2, x_3 входят и в первое, и во второе уравнения, то есть являются для них общими.

Найдем связь между значениями пар переменных (x_1, x_2) и (x_2, x_3) из таблицы истинности первого уравнения. Представим значения пар (x_1, x_2) и (x_2, x_3) как доли двудольного графа. Свяжем дугами пары, образующие решение первого уравнения системы.



Получим двудольный граф, в котором количество ребер соответствует количеству решений первого уравнения.

Во втором уравнении роль пары (x_1, x_2) выполняет пара (x_2, x_3), а роль пары (x_2, x_3) выполняет пара (x_3, x_4). Продолжим построение графа, добавим третью долю — пару переменных (x_3, x_4), свяжем ребрами значения пар (x_2, x_3) и (x_3, x_4), при которых второе уравнение имеет решение, получим:



При добавлении очередного уравнения в систему в многодольный граф добавляется очередная доля, связи между долями имеют один и тот же вид.

Запишем матрицу двудольного графа, первой долей которого являются значения наборов пары (x_i, x_{i+1}), второй долей — значения наборов пары (x_{i+1}, x_{i+2}). Строки матрицы соответствуют всем наборам значений пары (x_i, x_{i+1}) (00, 01, 10, 11), столбцы — всем наборам значений пары (x_{i+1}, x_{i+2}) (00, 01, 10, 11). Заметим, что наборы долей совпадают.

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Количество ребер двудольного графа равно сумме элементов матрицы M.

Матрица графа, состоящего из K однородных двудольных графов, определяется как произведение K матриц двудольного графа

$$Matr = M^K = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}^K$$

Тогда общее количество решений R_K системы из K уравнений определяется как

$$R_K = (1111) \times \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}^K \times \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Введем обозначения:

F_0 — единичная вектор-строка размерности 4;

F_i — вектор-строка размерности 4, состоящая из элементов

$$F_i = (f_{00}^i, f_{01}^i, f_{10}^i, f_{11}^i),$$

где каждый элемент представляет собой количество путей в соответствующую вершину i-й доли многодольного графа из всех вершин первой доли.

Тогда можно записать рекуррентную формулу

$$F_i = F_{i-1} \times M,\ i = 1, 2, \ldots, K$$

Общее количество путей из первой доли в i-ю долю определяется по формуле

$$R_i = F_i \times \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

или как сумма элементов строки F_i:

$$R_i = f_{00}^i + f_{01}^i + f_{10}^i + f_{11}^i$$

Например, для системы из трех уравнений

$$F_1 = F_0 \times M = (1111) \times \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (1112)$$

$$F_2 = F_1 \times M = (1112) \times \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (1123)$$

$$F_3 = F_2 \times M = (1123) \times \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (2234)$$

Умножение полученного вектора-строки на единичный вектор-столбец дает значение R_i, например, при $i = 3$

$$(2234) \times \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = 2+2+3+4 = 11$$

Проведем вычисления без матричных операций. Из рекуррентной формулы (или по изображению двудольного графа) определим количества путей между вершинами соседних долей (i-й и (i–1)-й):

$$f_{00}^i = f_{10}^{i-1}$$
$$f_{01}^i = f_{10}^{i-1}$$
$$f_{10}^i = f_{11}^{i-1}$$
$$f_{11}^i = f_{01}^{i-1} + f_{11}^{i-1}$$

Общее количество решений R_i для системы из i уравнений равно сумме количеств наборов

$$R_i = f_{00}^i + f_{01}^i + f_{10}^i + f_{11}^i = f_{01}^{i-1} + 2f_{10}^{i-1} + 2f_{11}^{i-1}$$

Построим таблицу, в которой вычислим последовательно количество решений системы по приведенным формулам. Заметим, что для вычислений по формулам необходимо задать начальные значения. Здесь это нулевая строка — все наборы пары x_1, x_2.

Количество уравнений i	Пары переменных	Fi				Ri
		f_{00}^i	f_{01}^i	f_{10}^i	f_{11}^i	
0	x_1, x_2	1	1	1	1	
1	x_2, x_3	1	1	1	2	5
2	x_3, x_4	1	1	2	3	7
3	x_4, x_5	2	2	3	4	11
4	x_5, x_6	3	3	4	6	16
5	x_6, x_7	4	4	6	9	23
6	x_7, x_8	6	6	9	13	34
7	x_8, x_9	9	9	13	19	50
8	x_9, x_{10}	13	13	19	28	73

Значения элементов вектора F_i в строках таблицы соответствуют K последовательным произведениям вектора-строки F_{i-1} на матрицу двудольного графа M. Значения R_i — сумма элементов вектора F_i в соответствующей строке.

Ответ: 73

Метод решения системы однородных логических уравнений

1) Записать уравнение однородной системы в общем виде, при этом:
- если необходимо, преобразовать выражение в левой части уравнения;
- если необходимо, привести уравнение к логическому выражению, равному единице;

2) Записать два соседних уравнения в общем виде;

3) Выделить множество общих переменных для двух соседних уравнений;

4) Построить таблицу истинности логического выражения левой части первого из двух соседних уравнений в общем виде;

5) Определить по таблице истинности связь между всеми возможными комбинациями значений выделенных множеств общих переменных двух соседних уравнений и представить таблицу истинности в виде двудольного графа так, чтобы каждому решению соответствовало ровно одно ребро. Множества вершин первой и второй долей графа должны совпадать. Вершины представляют собой все наборы значений, которые могут принимать выделенные общие переменные;

6) Определить зависимость количества решений для системы i уравнений от количества решений системы $(i - 1)$ уравнений;

7) Найти количество решений системы уравнений одним из способов:
- построением таблицы последовательного решения систем i уравнений, $i = 1, 2, \ldots, K$;
- умножением векторов и матриц для определения количества путей в многодольном графе, составленном из однородных двудольных.

Пример 2 [2]

Сколько существует различных наборов значений логических переменных x_1, x_2, …, x_7, x_8, которые удовлетворяют всем перечисленным ниже условиям?

$$(x_1 \equiv x_2) \vee \overline{(x_1 \equiv x_3)} = 1$$
$$(x_2 \equiv x_3) \vee \overline{(x_2 \equiv x_4)} = 1$$
$$(x_3 \equiv x_4) \vee \overline{(x_3 \equiv x_5)} = 1$$
$$(x_4 \equiv x_5) \vee \overline{(x_4 \equiv x_6)} = 1$$
$$(x_5 \equiv x_6) \vee \overline{(x_5 \equiv x_7)} = 1$$
$$(x_6 \equiv x_7) \vee \overline{(x_6 \equiv x_8)} = 1$$

В ответе не нужно перечислять все различные наборы значений $x_1, x_2, \ldots, x_7, x_8$, при которых выполнена данная система равенств. В качестве ответа нужно указать количество таких наборов.

Решение

1) Запишем однородное уравнение системы в общем виде

$$(x_i \equiv x_{i+1}) \vee \overline{(x_i \equiv x_{i+2})} = 1,\ i = 1, 2, \ldots, 6$$

2) Запишем два соседних уравнения

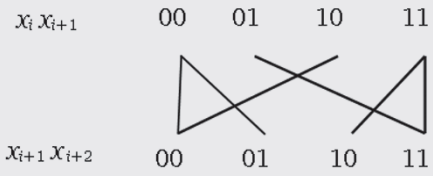
$$(x_i \equiv x_{i+1}) \vee \overline{(x_i \equiv x_{i+2})} = 1$$
$$(x_{i+1} \equiv x_{i+2}) \vee \overline{(x_{i+1} \equiv x_{i+3})} = 1$$

3) Общие переменные двух соседних уравнений: x_{i+1} и x_{i+2}, при этом переменная x_{i+1} записана на месте переменной x_i, а переменная x_{i+2} записана на месте переменной x_{i+1}. Таким образом, долями двудольного графа должны быть наборы значений пар (x_i, x_{i+1}) и (x_{i+1}, x_{i+2}).

4) Построим таблицу истинности для логического выражения левой части i-го уравнения

x_i	x_{i+1}	x_{i+2}	$x_i \equiv x_{i+1}$	$\overline{x_i \equiv x_{i+2}}$	$(x_i \equiv x_{i+1}) \vee \overline{x_i \equiv x_{i+2}}$
0	0	0	1	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

5) Определим связь между всеми наборами пар переменных и построим двудольный граф



6) Количество решений в общем виде

$$R_K = \begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix} \times \begin{pmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 1\end{pmatrix}^K \times \begin{pmatrix}1\\1\\1\\1\end{pmatrix}$$

7) Запишем формулы

$$f_{00}^i = f_{00}^{i-1} + f_{10}^{i-1}$$
$$f_{01}^i = f_{00}^{i-1}$$
$$f_{10}^i = f_{11}^{i-1}$$
$$f_{11}^i = f_{01}^{i-1} + f_{11}^{i-1}$$

Общее количество решений R_i для системы из i уравнений равно сумме количеств наборов

$$R_i = f_{00}^i + f_{01}^i + f_{10}^i + f_{11}^i = 2f_{00}^{i-1} + f_{01}^{i-1} + f_{10}^{i-1} + 2f_{11}^{i-1}$$

Построим таблицу вычислений по формулам (или выполним последовательные умножения строки на матрицу). Начальные значения — наборы пары x_1, x_2.

Количество уравнений i	Количество переменных	Пары	F_i				R_i
			f^i_{00}	f^i_{01}	f^i_{10}	f^i_{11}	
0		x_1, x_2	1	1	1	1	
1	3	x_2, x_3	2	1	1	2	6
2	4	x_3, x_4	3	2	2	3	10
3	5	x_4, x_5	5	3	3	5	16
4	6	x_5, x_6	8	5	5	8	26
5	7	x_6, x_7	13	8	8	13	42
6	8	x_7, x_8	21	13	13	21	68

Заметим, что последовательности значений в столбцах таблицы представляют собой числа Фибоначчи (см. Приложение 2). Таким образом, для системы K уравнений, содержащей $N = K + 2$ переменных, количество решений определяется по формуле

$$R_N = 2 \cdot \text{fib}(N)$$

Ответ: 68

Пример 3 [2]

Сколько существует различных наборов значений логических переменных x_1, x_2, …, x_8, x_9, которые удовлетворяют всем перечисленным ниже условиям?

$$\overline{(x_1 \equiv x_2)} \vee \overline{(x_1 \equiv x_3)} \wedge (x_2 \equiv x_3) = 1$$
$$\overline{(x_3 \equiv x_4)} \vee \overline{(x_3 \equiv x_5)} \wedge (x_4 \equiv x_5) = 1$$
$$\overline{(x_5 \equiv x_6)} \vee \overline{(x_5 \equiv x_7)} \wedge (x_6 \equiv x_7) = 1$$
$$\overline{(x_7 \equiv x_8)} \vee \overline{(x_7 \equiv x_9)} \wedge (x_8 \equiv x_9) = 1$$

В ответе не нужно перечислять все различные наборы значений x_1, x_2, …, x_9, при которых выполнена данная система равенств. В качестве ответа нужно указать количество таких наборов.

Решение

1) Напомним, что K — это количество уравнений, N — количество переменных. Запишем i-е уравнение системы в общем виде

$$\overline{(x_{2i-1} \equiv x_{2i})} \vee \overline{(x_{2i-1} \equiv x_{2i+1})} \wedge (x_{2i} \equiv x_{2i+1}) = 1$$

2) Запишем два соседних уравнения

$$\overline{(x_{2i-1} \equiv x_{2i})} \vee \overline{(x_{2i-1} \equiv x_{2i+1})} \wedge (x_{2i} \equiv x_{2i+1}) = 1$$
$$\overline{(x_{2i+1} \equiv x_{2i+2})} \vee \overline{(x_{2i+1} \equiv x_{2i+3})} \wedge (x_{2i+2} \equiv x_{2i+3}) = 1$$

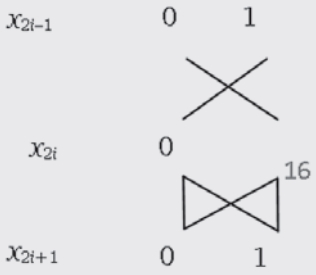
В соседних уравнениях есть только одна общая переменная — x_{2i+1}.

3) В $(i+1)$-м уравнении записана переменная x_{2i+1} на месте переменной x_{2i-1} i-го уравнения, следовательно, надо найти связь между этими двумя переменными.

4) Построим таблицу истинности для логического выражения левой части уравнения

x_{2i-1}	x_{2i}	x_{2i+1}	$\overline{x_{2i-1} \equiv x_{2i}}$	$\overline{x_{2i-1} \equiv x_{2i+1}}$	$(x_{2i} \equiv x_{2i+1})$	$\overline{(x_{2i+1} \equiv x_{2i+3})} \wedge (x_{2i+2} \equiv x_{2i+3})$	
0	0	0	0	0	1	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	1
1	1	0	0	1	0	0	0
1	1	1	0	0	1	0	0

5) Определим связь между возможными комбинациями переменных x_{2i-1} и x_{2i+1}. Для этого необходимо учесть также значения переменной x_{2i}, входящей в таблицу истинности. Построим трехдольный граф с долями, соответствующими значениям наборов переменных x_{2i-1}, x_{2i} и x_{2i+1}. Каждая доля графа состоит из двух вершин — 0 и 1.



6) Матрица полученного графа образуется перемножением матриц образующих его двудольных графов

$$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\times\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}=\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$$

7) Общее количество решений системы K уравнений

$$R_K=(11)\times\begin{pmatrix}11\\ 11\end{pmatrix}^K\times\begin{pmatrix}1\\ 1\end{pmatrix}$$

Запишем соотношения, определяющие количество путей от доли x_{2i-1} к доле x_{2i+1}:

$$f_0^i=f_0^{i-1}+f_1^{i-1}$$
$$f_1^i=f_0^{i-1}+f_1^{i-1}$$

Общее количество решений системы i уравнений равно $R_i=2f_0^{i-1}+2f_1^{i-1}$
Таблица вычислений имеет вид

Количество уравнений i	Количество переменных	Переменные	F_i		R_i
			f_0^i	f_1^i	
0		x_1	1	1	
1	3	x_3	2	2	4
2	5	x_5	4	4	8
3	7	x_7	8	8	16
4	9	x_9	16	16	32

Для системы K уравнений количество решений равно 2^{K+1}.
Ответ: 32

Пример 4 [2]

Сколько существует различных наборов значений логических переменных x_1, x_2, …, x_9, x_{10}, которые удовлетворяют всем перечисленным ниже условиям?

$$(x_1\vee x_2)\wedge(\overline{x_3}\vee\overline{x_4})=0$$
$$(x_3\vee x_4)\wedge(\overline{x_5}\vee\overline{x_6})=0$$
$$(x_5\vee x_6)\wedge(\overline{x_7}\vee\overline{x_8})=0$$
$$(x_7\vee x_8)\wedge(\overline{x_9}\vee\overline{x_{10}})=0$$

В ответе не нужно перечислять все различные наборы значений x_1, x_2, …, x_9, при которых выполнена данная система равенств. В качестве ответа нужно указать количество таких наборов.

Решение

1) Запишем i-е уравнение системы в общем виде

$$(x_{2i-1}\vee x_{2i})\wedge(\overline{x_{2i+1}}\vee\overline{x_{2i+2}})=0$$

Добавим отрицание к обеим частям уравнения, получим

$$\overline{x_{2i-1}}\wedge\overline{x_{2i}}\vee x_{2i+1}\wedge x_{2i+2}=1,\ i=1,2,3,4$$

2) Запишем два соседних уравнения

$$\overline{x_{2i-1}} \wedge \overline{x_{2i}} \vee x_{2i+1} \wedge x_{2i+2} = 1$$
$$\overline{x_{2i+1}} \wedge \overline{x_{2i+2}} \vee x_{2i+3} \wedge x_{2i+4} = 1$$

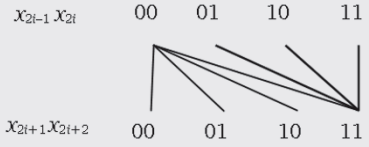
3) В i-м и $(i+1)$-м уравнениях две общие переменные x_{2i+1} и x_{2i+2}.

На месте переменной x_{2i-1} записана переменная x_{2i+1}, а на месте переменной x_{2i} записана переменная x_{2i+2}. Найдем связь между парами переменных (x_{2i-1}, x_{2i}) и (x_{2i+1}, x_{2i+2}).

4) Построим таблицу истинности

x_{2i-1}	x_{2i}	x_{2i+1}	x_{2i+2}	$\overline{x_{2i-1}} \wedge \overline{x_{2i}}$	$x_{2i+1} \wedge x_{2i+2}$	$\overline{x_{2i-1}} \wedge \overline{x_{2i}} \vee x_{2i+1} \wedge x_{2i+2}$
0	0	0	0	1	0	1
0	0	0	1	1	0	1
0	0	1	0	1	0	1
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	1	1

5) Определим связь между парами переменных, построим двудольный граф



6) Матрица двудольного графа имеет вид

$$M = \begin{pmatrix} 1111 \\ 0001 \\ 0001 \\ 0001 \end{pmatrix}$$

7) Количество решений определяется уравнением

$$R = (1111) \times \begin{pmatrix} 1111 \\ 0001 \\ 0001 \\ 0001 \end{pmatrix}^4 \times \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

8) Запишем соотношения:

$$f_{00}^i = f_{00}^{i-1}$$
$$f_{01}^i = f_{00}^{i-1}$$
$$f_{10}^i = f_{00}^{i-1}$$
$$f_{11}^i = f_{00}^{i-1} + f_{01}^{i-1} + f_{10}^{i-1} + f_{11}^{i-1}$$

Общее количество решений R_i для системы из i уравнений равно сумме количеств наборов

$$R_i = 4f_{00}^i + f_{01}^{i-1} + f_{10}^{i-1} + f_{11}^{i-1}$$

Построим таблицу вычислений по формулам (или рекуррентно-последовательным умножением строки на матрицу). Начальные значения — наборы пары x_1, x_2.

Количество уравнений i	Количество переменных	Пары	F_i				R_i
			f^i_{00}	f^i_{01}	f^i_{10}	f^i_{11}	
0		x_1, x_2	1	1	1	1	
1	4	x_3, x_4	1	1	1	4	7
2	6	x_5, x_6	1	1	1	7	10
3	8	x_7, x_8	1	1	1	10	13
4	10	x_9, x_{10}	1	1	1	13	16

Для системы K уравнений количество решений равно

$$R_K = 4 + 3K$$

Ответ: 16

Пример 5 [2]

Сколько существует различных наборов значений логических переменных $x_1, x_2, \ldots, x_9, x_{10}$, которые удовлетворяют всем перечисленным ниже условиям?

$$(x_1 \equiv x_2) \wedge \overline{(x_3 \equiv x_4)} = 0$$

$$(x_3 \equiv x_4) \wedge \overline{(x_5 \equiv x_6)} = 0$$

$$(x_5 \equiv x_6) \wedge \overline{(x_7 \equiv x_8)} = 0$$

$$(x_7 \equiv x_8) \wedge \overline{(x_9 \equiv x_{10})} = 0$$

В ответе не нужно перечислять все различные наборы значений $x_1, x_2, \ldots, x_9, x_{10}$, при которых выполнена данная система равенств. В качестве ответа нужно указать количество таких наборов.

Решение

1) Запишем i-е уравнение системы в общем виде

$$(x_{2i-1} \equiv x_{2i}) \wedge \overline{(x_{2i+1} \equiv x_{2i+2})} = 0$$

Добавим отрицание к обеим частям уравнения, получим

$$(x_{2i-1} \oplus x_{2i}) \vee (x_{2i+1} \equiv x_{2i+2}) = 1,\ i = 1, 2, 3, 4$$

2) Запишем два соседних уравнения

$$(x_{2i-1} \oplus x_{2i}) \vee (x_{2i+1} \equiv x_{2i+2}) = 1$$

$$(x_{2i+1} \oplus x_{2i+2}) \vee (x_{2i+3} \equiv x_{2i+4}) = 1$$

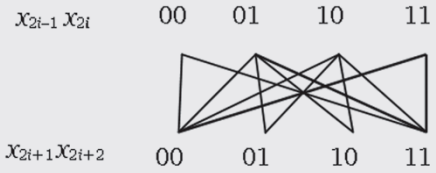
3) В i-м и $(i+1)$-м уравнениях две общие переменные x_{2i+1} и x_{2i+2}.

На месте переменной x_{2i-1} записана переменная x_{2i+1}, а на месте переменной x_{2i} записана переменная x_{2i+2}. Найдем связь между парами переменных (x_{2i-1}, x_{2i}) и (x_{2i+1}, x_{2i+2}).

4) Построим таблицу истинности

x_{2i-1}	x_{2i}	x_{2i+1}	x_{2i+2}	$x_{2i-1} \oplus x_{2i}$	$x_{2i+1} \equiv x_{2i+2}$	$(x_{2i-1} \oplus x_{2i}) \vee (x_{2i+1} \equiv x_{2i+2})$
0	0	0	0	0	1	1
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	1	1	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	1	1

5) Определим связь между парами переменных, построим двудольный граф



6) Матрица двудольного графа имеет вид

$$M=\begin{pmatrix}1001\\1111\\1111\\1001\end{pmatrix}$$

7) Количество решений определяется уравнением

$$R=(1111)\times\begin{pmatrix}1001\\1111\\1111\\1001\end{pmatrix}^4\times\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$$

8) Запишем соотношения:

$$f_{00}^i=f_{00}^{i-1}+f_{01}^{i-1}+f_{10}^{i-1}+f_{11}^{i-1}$$
$$f_{01}^i=f_{01}^{i-1}+f_{10}^{i-1}$$
$$f_{10}^i=f_{01}^{i-1}+f_{10}^{i-1}$$
$$f_{11}^i=f_{00}^{i-1}+f_{01}^{i-1}+f_{10}^{i-1}+f_{11}^{i-1}$$

Общее количество решений R_i для системы из i уравнений равно сумме количеств наборов

$$R_i=2f_{00}^{i-1}+4f_{01}^{i-1}+4f_{10}^{i-1}+2f_{11}^{i-1}$$

Построим таблицу вычислений по формулам (или рекуррентно-последовательным умножением строки на матрицу). Начальные значения — наборы пары x_1, x_2.

Количество уравнений i в системе	Количество переменных в системе	Пары	F_i				R_i
			f_{00}^i	f_{01}^i	f_{10}^i	f_{11}^i	
0		x_1, x_2	1	1	1	1	
1	4	x_3, x_4	4	2	2	4	12
2	6	x_5, x_6	12	4	4	12	32
3	8	x_7, x_8	32	8	8	32	80
4	10	x_9, x_{10}	80	16	16	80	192

Ответ: 192

Пример 6 [2]

Сколько существует различных наборов значений логических переменных $x_1, x_2, \ldots, x_7, x_8$, которые удовлетворяют всем перечисленным ниже условиям?

$$(x_1\equiv x_2)\equiv(x_1\equiv x_3)=0$$
$$(x_2\equiv x_3)\equiv(x_2\equiv x_4)=0$$
$$(x_3\equiv x_4)\equiv(x_3\equiv x_5)=0$$
$$(x_4\equiv x_5)\equiv(x_4\equiv x_6)=0$$
$$(x_5\equiv x_6)\equiv(x_5\equiv x_7)=0$$
$$(x_6\equiv x_7)\equiv(x_6\equiv x_8)=0$$

В ответе не нужно перечислять все различные наборы значений $x_1, x_2, \ldots, x_7, x_8$, при которых выполнена данная система равенств. В качестве ответа нужно указать количество таких наборов.

Решение

1) Запишем i-е уравнение системы в общем виде

$$(x_i \equiv x_{i+1}) \equiv (x_i \equiv x_{i+2}) = 0$$

Добавим отрицание к обеим частям уравнения, получим

$$(x_i \oplus x_{i+1}) \oplus (x_i \oplus x_{i+2}) = 1, \quad i = 1, 2, \ldots, 6$$

2) Запишем два соседних уравнения

$$(x_i \oplus x_{i+1}) \oplus (x_i \oplus x_{i+2}) = 1,$$

$$(x_{i+1} \oplus x_{i+2}) \oplus (x_{i+1} \oplus x_{i+3}) = 1$$

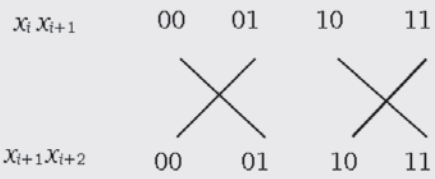
3) В i-м и $(i+1)$-м уравнениях две общие переменные x_{i+1} и x_{i+2}.

На месте переменной x_i записана переменная x_{i+1}, а на месте переменной x_{i+1} записана переменная x_{i+2}. Найдем связь между парами переменных (x_i, x_{i+1}) и (x_{i+1}, x_{i+2}).

4) Построим таблицу истинности

x_i	x_{i+1}	x_{i+2}		$x_i \oplus x_{i+1}$	$x_i \oplus x_{i+2}$	$(x_i \oplus x_{i+1}) \oplus (x_i \oplus x_{i+2})$
0	0	0		0	0	0
0	0	1		0	1	1
0	1	0		1	0	1
0	1	1		1	1	0
1	0	0		1	1	0
1	0	1		1	0	1
1	1	0		0	1	1
1	1	1		0	0	0

5) Определим связь между парами переменных, построим двудольный граф



6) Матрица двудольного графа имеет вид

$$M = \begin{pmatrix} 0100 \\ 1000 \\ 0001 \\ 0010 \end{pmatrix}$$

7) Количество решений определяется уравнением

$$R = (1111) \times \begin{pmatrix} 0100 \\ 1000 \\ 0001 \\ 0010 \end{pmatrix}^6 \times \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

8) Запишем соотношения:

$$f_{00}^i = f_{01}^{i-1}$$

$$f_{01}^i = f_{00}^{i-1}$$

$$f_{10}^i = f_{11}^{i-1}$$

$$f_{11}^i = f_{10}^{i-1}$$

Общее количество решений R_i для системы из i уравнений равно сумме количеств наборов

$$R_i = f_{00}^{i-1} + f_{01}^{i-1} + f_{10}^{i-1} + f_{11}^{i-1}$$

Построим таблицу вычислений по формулам (или рекуррентно-последовательным умножением строки на матрицу). Начальные значения — наборы пары x_1, x_2.

Количество уравнений i в системе	Количество переменных в системе	Пары	F_i				R_i
			f^i_{00}	f^i_{01}	f^i_{10}	f^i_{11}	
0		x_1, x_2	1	1	1	1	
1	3	x_2, x_3	1	1	1	1	4
2	4	x_3, x_4	1	1	1	1	4
3	5	x_4, x_5	1	1	1	1	4
4	6	x_5, x_6	1	1	1	1	4
5	7	x_6, x_7	1	1	1	1	4
6	8	x_7, x_8	1	1	1	1	4

Ответ: 4

Пример 7

Задание — из второго отборочного тура Московской олимпиады по информатике 2011–2012 года. Определить количество различных решений системы уравнений

$$(x_1 \to x_2) \wedge (y_1 \to y_2) \wedge (x_1 \to y_1) = 1$$
$$(x_2 \to x_3) \wedge (y_2 \to y_3) \wedge (x_2 \to y_2) = 1$$
$$\dots$$
$$(x_{n-1} \to x_n) \wedge (y_{n-1} \to y_n) \wedge (x_{n-1} \to y_{n-1}) = 1$$
$$(x_n \to y_n) = 1$$

при $n = 6$, $n = 100$.

Решение. Попытаемся найти общее решение заданной системы.

1) Перепишем систему уравнений в виде

$$(x_1 \to y_1) = 1$$
$$(x_1 \to x_2) \wedge (y_1 \to y_2) \wedge (x_2 \to y_2) = 1$$
$$(x_2 \to x_3) \wedge (y_2 \to y_3) \wedge (x_3 \to y_3) = 1$$
$$\dots$$
$$(x_{n-1} \to x_n) \wedge (y_{n-1} \to y_n) \wedge (x_n \to y_n) = 1$$

или в общем виде:

$$(x_1 \to y_1) = 1$$
$$(x_{i-1} \to x_i) \wedge (y_{i-1} \to y_i) \wedge (x_i \to y_i) = 1,\ i = 2, 3, \dots, n-1$$

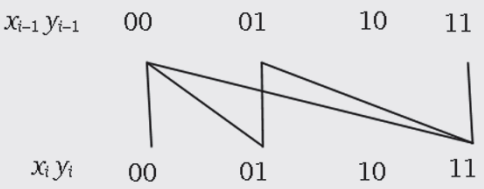
2) Запишем два соседних уравнения (пока первое уравнение не будем учитывать)

$$(x_{i-1} \to x_i) \wedge (y_{i-1} \to y_i) \wedge (x_i \to y_i) = 1$$
$$(x_i \to x_{i+1}) \wedge (y_i \to y_{i+1}) \wedge (x_{i+1} \to y_{i+1}) = 1$$

3) Соседние уравнения имеют две общие переменные x_i и y_i. В i-м уравнении записаны переменные x_i и y_i на месте переменных x_{i-1} и y_{i-1} в $(I-1)$-м уравнении. Найдем связь между парами переменных (x_{i-1}, y_{i-1}) и (x_i, y_i).

4) Построим таблицу истинности $(i-1)$-го уравнения. Заполним таблицу по строкам. Если в строке появляется 0 при выполнении импликации, выражение в левой части уравнения равно 0, поэтому дальше строку можно не заполнять.

5) Определим связь множества пар $x_{i-1}\,y_{i-1}$ с множеством пар $x_i\,y_i$ для строк, в которых выражение истинно. Паре значений 00 переменных $x_{i-1}\,y_{i-1}$ соответствуют пары значений 00, 01, 11 переменных $x_i\,y_i$. Паре 01 — пары 01 и 11, паре 11 — пара 11.



x_{i-1}	y_{i-1}	x_i	y_i	$x_{i-1} \to x_i$	$y_{i-1} \to y_i$	$x_i \to y_i$	$(x_{i-1} \to x_i) \wedge (y_{i-1} \to y_i) \wedge (x_i \to y_i)$
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	0	1	1	0	0
0	0	1	1	1	1	1	1
0	1	0	0	1	0		0
0	1	0	1	1	1	1	1
0	1	1	0	1	0		0
0	1	1	1	1	1	1	1
1	0	0	0	0			0
1	0	0	1	0			0
1	0	1	0	0			0
1	0	1	1	0			0
1	1	0	0	0			0
1	1	0	1	0			0
1	1	1	0	1	0		0
1	1	1	1	1	1	1	1

6) Для каждого следующего уравнения пары преобразуются таким же образом. Обозначим количество пар 00, 01, 10, 11 для системы i уравнений как

$$f_{00}^i, f_{01}^i, f_{10}^i, f_{11}^i$$

Тогда можно записать

$$f_{00}^i = f_{00}^{i-1}$$
$$f_{01}^i = f_{00}^{i-1} + f_{01}^{i-1}$$
$$f_{10}^i = 0$$
$$f_{11}^i = f_{00}^{i-1} + f_{01}^{i-1} + f_{11}^{i-1}$$

7) Общее количество решений R_i для системы из i уравнений равно сумме количеств наборов

$$R_i = f_{00}^i + f_{01}^i + f_{10}^i + f_{11}^i = 3f_{00}^{i-1} + 2f_{01}^{i-1} + f_{11}^{i-1}$$

Построим таблицу, в которой вычислим по формулам количество решений системы. Заметим, что значения наборов переменных x_1, y_1 в нулевой строке определяются первым уравнением системы и значение набора 10 равно 0.

Количество уравнений i	F_i				R_i
	f_{00}^i	f_{01}^i	f_{10}^i	f_{11}^i	
0	1	1	0	1	
1	1	1	0	1	3
2	1	2	0	3	6
3	1	3	0	6	10
4	1	4	0	10	15
5	1	5	0	15	21
6	1	6	0	21	28

Заметим, что

$$f_{11}^i = R_{i-1}$$

Тогда можно записать, что количество решений равно

$$R_i = \begin{cases} 3, & i = 1 \\ R_{i-1} + i + 1, & i > 1 \end{cases}$$

При $i = n$ можно записать

$$R_n = 3 + \sum_{i=2}^{n} (i+1)$$

Последовательность значений $i + 1$ представляет собой арифметическую прогрессию, первый член которой $a_1 = 3$, разность $d = 1$, количество членов прогрессии равно $(n - 1)$. Сумма арифметической прогрессии равна

$$S = \frac{3 + (n+1)}{2}(n-1) = \frac{n^2 + 3n}{2} - 2$$

Количество решений системы n уравнений при $n > 1$ равно

$$R_n = 3 + \frac{n^2 + 3n}{2} - 2 = \frac{n^2 + 3n}{2} + 1$$

При $n = 6$ получим $(36 + 18) / 2 + 1 = 28$ решений, при $n = 100$ получим $(10\,000 + 300) / 2 + 1 = 5151$ решение.

Заключение

Предложен метод решения систем однородных логических уравнений. Метод проиллюстрирован решением ряда примеров. Авторы стремились продемонстрировать универсальность предложенного метода. Решение некоторых примеров другими известными методами приводится на диске.

Список литературы

1. *Андреева Е.В.* Математические основы информатики. Элективный курс: Учебное пособие / *Е.В. Андреева, Л.Л. Босова, И.Н. Фалина*. М.: БИНОМ. Лаборатория знаний, 2005.
2. Самое полное издание типовых вариантов заданий ЕГЭ: Информатика, 2012 / авт.-сост. *Д.М. Ушаков, П.А. Якушкин.* М.: Астрель, 2012.
3. *Мирончик Ел.А., Мирончик Ек.А.* Системы логических уравнений. Метод отображений: Материалы Десятой открытой всероссийской конференции “Преподавание информационных технологий в Российской Федерации”. М., 2012.
4. Информатика и ИКТ. ЕГЭ: Учебно-справочные материалы (Серия “Итоговый контроль: ЕГЭ”) / *С.М. Авдошин, Р.З. Ахметсафина, О.В. Максименкова и др.* М.; СПб.: Просвещение, 2012.
5. Информатика и ИКТ: ЕГЭ-2012: Контрольные тренировочные материалы с ответами и комментариями (Серия “Итоговый контроль: ЕГЭ”) / *С.М. Авдошин, Р.З. Ахметсафина, О.В. Максименкова*. М.; СПб.: Просвещение, 2012.
6. Информатика: ЕГЭ-2011: Контрольные тренировочные материалы с ответами и комментариями (Серия “Итоговый контроль: ЕГЭ”) / *С.М. Авдошин, Р.З. Ахметсафина, О.В. Максименкова и др.* М.; СПб.: Просвещение, 2011.
7. *Поляков К.Ю.* Системы логических уравнений. М.: Информатика, № 14, 2011.
8. *Авдошин С.М., Ахметсафина Р.З., Максименкова О.В.* Информатика: Логика и алгоритмы. Пособие для самостоятельной подготовки (Серия “Сложные темы ЕГЭ”). М.; СПб.: Просвещение, 2013.

Приложение 1. Элементы теории матриц

Основные понятия и определения

Прямоугольная таблица чисел вида

$$\begin{pmatrix} a_{11} & a_{12} \ldots a_{1n} \\ a_{21} & a_{22} \ldots a_{2n} \\ \ldots & \ldots\ \ldots\ \ldots \\ a_{m1} & a_{m2} \ldots a_{mn} \end{pmatrix}$$

называется **прямоугольной матрицей** размера $m \times n$, где m — количество строк, а n — количество столбцов.

Числа, образующие матрицу a_{ij}, где $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, называются **элементами матрицы**.

Числа i и j называются **индексами элемента** a_{ij}. Первый индекс указывает номер строки, второй — номер столбца, в котором расположен элемент.

Если $m = n$, матрица называется **квадратной матрицей** порядка n (или порядка m).

На уроках информатики изучаются двумерные массивы — представление матриц в компьютере на языке программирования.

Матрица размером $m \times 1$ называется вектором-столбцом. Как правило, второй индекс, равный везде 1, не записывают

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}$$

Матрица размером $1 \times n$ называется вектором-строкой. Как правило, первый индекс, равный везде 1, не записывают

$$(a_1 \quad a_2 \quad \dots \quad a_n)$$

Вектор-строку и вектор-столбец обычно представляют на языках программирования как одномерные массивы.

Операции над матрицами

Операции над матрицами выполняются поэлементно.

Суммой двух матриц A и B одинаковой размерности $m \times n$ называется матрица C той же размерности, каждый элемент которой равен сумме соответствующих элементов исходных матриц.

$$c_{ij} = a_{ij} + b_{ij},\ i = 1, 2, \dots, m;\ j = 1, 2, \dots, n$$

Сложение матриц коммутативно и ассоциативно

$$A + B = B + A$$
$$(A + B) + C = A + (B + C)$$

Произведением двух матриц A размерности $(m \times q)$ и B размерности $(q \times n)$, в которых количество столбцов матрицы A равно количеству строк матрицы B, называется матрица C размерности $(m \times n)$, у которой количество строк m равно количеству строк первой матрицы, а количество столбцов n равно количеству столбцов второй. Каждый элемент c_{ij} матрицы C равен сумме попарных произведений элементов соответствующей строки первой матрицы и элементов соответствующего столбца второй.

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \dots a_{iq} \cdot b_{qj} = \sum_{k=1}^{q} a_{ik} \cdot b_{kj},\ i = 1, 2, \dots, m;\ j = 1, 2, \dots, n$$

Пример: $m = 2, q = 3, n = 2$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix}$$

Произведение матриц, вообще говоря, не коммутативно

$$A \cdot B \neq B \cdot A$$

Для лучшего понимания операции произведения двух матриц можно использовать схематическое изображение, показанное на *рис.* П1.

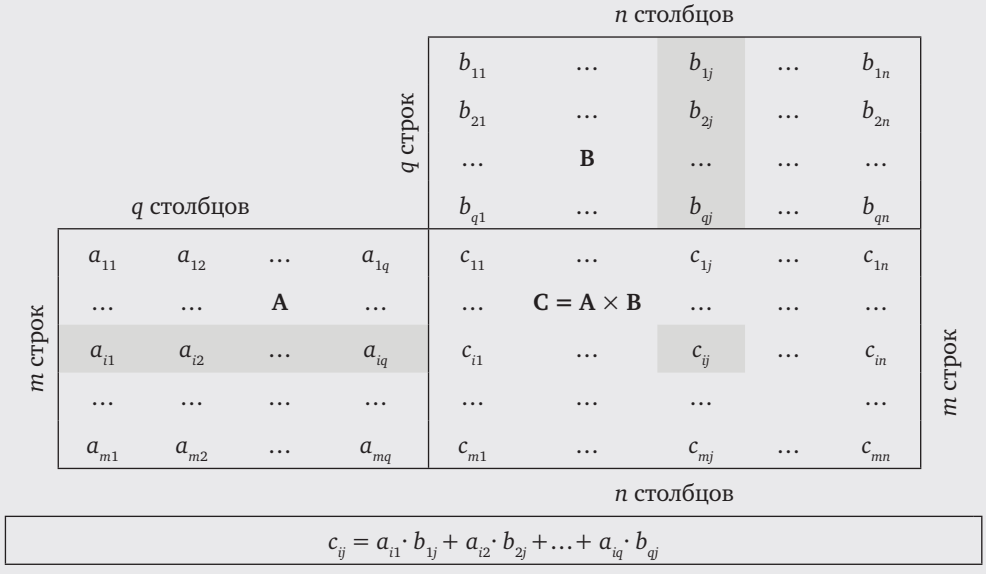


Рис. П1. Схема перемножения матриц

Произведение вектора-строки на матрицу. При умножении вектора-строки A размерности q на матрицу B размерности $(q \times n)$ количество элементов в векторе-строке должно быть равно количеству строк в матрице. Результатом является вектор-строка D, количество элементов которой равно количеству столбцов n матрицы B. Элементы вектора D определяются по формуле

$$d_j = a_1 \cdot b_{1j} + a_2 \cdot b_{2j} + \dots a_q \cdot b_{qj} = \sum_{k=1}^{q} a_k \cdot b_{kj},\ j = 1, 2, \dots, n$$

Пример:

$$\begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} a_1b_{11} + a_2b_{21} + a_3b_{31} & a_1b_{12} + a_2b_{22} + a_3b_{32} \end{pmatrix}$$

Произведение матрицы на вектор-столбец. При умножении матрицы A размерности ($m \times q$) на вектор-столбец B размерности q количество столбцов матрицы A должно быть равно количеству элементов в векторе-столбце B. Результатом является вектор-столбец D, количество элементов которого равно количеству строк m матрицы A. Элементы вектора D определяются по формуле

$$d_i = a_{i1} \cdot b_1 + a_{i2} \cdot b_2 + \ldots a_{iq} \cdot b_q = \sum_{k=1}^{q} a_{ik} \cdot b_k,\ i = 1, 2, \ldots, n$$

Пример:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 \end{pmatrix}$$

Произведение вектора-строки A на вектор-столбец B — число, являющееся суммой попарных произведений элементов векторов. Размерности векторов должны совпадать.

Пример:

$$\begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = a_1b_1 + a_2b_2 + a_3b_3$$

Произведение вектора-столбца на вектор-строку. В результате умножения вектора-столбца A размерности m на вектор-строку B размерности n получим матрицу C размерности ($m \times n$), элементы которой являются попарными произведениями элементов векторов A и B.

$$c_{ij} = a_i \cdot b_j,\ i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n$$

Пример:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}\begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix} = \begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{pmatrix}$$

Для лучшего запоминания операций произведения можно использовать схематические изображения, показанные на *рис.* П2. A — первый сомножитель, B — второй сомножитель. Произведение C = A × B выделено цветом.

На *рис.* П2*а* показана схема умножения вектора-строки на матрицу, П2*б* — умножение матрицы на вектор-столбец, П2*в* — умножение вектора-столбца на вектор-строку, П2*г* — умножение вектора-строки на вектор-столбец.

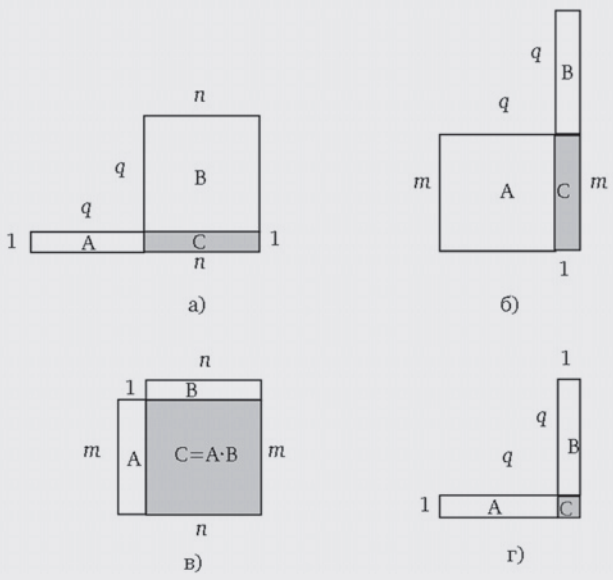


Рис. П2. Схемы умножения матриц и векторов

Приложение 2. Рекуррентные формулы

Рекуррентная формула — это формула вычисления n-го члена какой-либо последовательности (чаще всего числовой) с использованием нескольких ее предыдущих членов.

Рекуррентная формула k-го порядка — это формула вида

$$a_n = f(n, a_{n-1}, a_{n-2}, \ldots, a_{n-k})$$

выражающая каждый член последовательности a_n через k предыдущих членов.

Для вычисления значений последовательности по рекуррентной формуле k-го порядка необходимо определить первые k членов последовательности.

Примеры:

1. Вычисление факториала натурального числа.

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$$

$$n! = (n-1)! \cdot n,\ 0! = 1$$

Это — рекуррентная формула первого порядка, так как для вычисления очередного значения используется одно предыдущее. Первый член последовательности 0! принимается равным 1.

2. Вычисление чисел Фибоначчи. Последовательность вида 0, 1, 1, 2, 3, 5, 8, 13, … называется числами Фибоначчи. Вычисления проводятся по рекуррентной формуле второго порядка, поэтому два первых элемента последовательности должны быть заданы:

$$fib(n) = \begin{cases} 0, & n = 0; \\ 1, & n = 1; \\ fib(n-2) + fib(n-1), & n \geq 2 \end{cases}$$

Приложение 3. Элементы теории графов

Основные понятия и определения

Граф — это средство наглядного представления элементов объекта и связей между ними.

Граф состоит из множества **вершин**, все или часть из которых связаны линиями. Линии называют **ребрами** графа.

Вершины называют также **узлами** и изображают точками, кругами, овалами, прямоугольниками и т.д. На *рис.* П3 вершины A, B, C, D, E, F обозначены кругами.

Если ребро направленное, то есть со стрелкой, оно называется **дугой**. Граф с такими ребрами называется **ориентированным**.

Граф с ненаправленными ребрами называется **неориентированным** (*рис.* П3). Одно ребро заменяет две дуги между теми же вершинами, направленные в противоположные стороны.

Форма ребер несущественна, важен только сам факт соединения вершин. Ребра могут пересекаться, но точки пересечения не являются вершинами графа.

Две вершины, соединенные ребром, называются **смежными**, на *рис.* П3 смежными являются вершины A и B, A и C, B и E и др.

Ребра могут означать расстояние между вершинами, время перехода от одной вершины к другой и т.д. В таких случаях ребро может помечаться числом, которое называют весом или нагрузкой. Граф в таком случае называется **взвешенным или нагруженным.**

Маршрут в графе — это конечная последовательность вершин графа, где каждая из вершин соединена со следующей в последовательности вершиной как минимум одним ребром. На *рис.* П3 маршрутами являются последовательности вершин ACEF, ABCDECBA, BCDEB и др. Маршрутом называют также последовательность ребер, в которой конец одного ребра служит началом следующего.

Длина маршрута — это количество ребер в маршруте (или сумма длин ребер маршрута, если длины заданы). Примеры: количество ребер маршрута ACEF равно 3, длина маршрута равна 4 + 4 + 3 = 11. В маршрут ABCDECBA входят 7 ребер (ребра считаются с повторениями), длина маршрута равна 16.

Путь — это маршрут в ориентированном графе, вершины которого связаны дугами. **Длина пути** — это количество дуг пути (или сумма длин его дуг, если длины заданы).

Если конец последнего ребра в последовательности совпал с началом первого ребра, маршрут называют **циклическим**. Пример — маршруты ABCDECBA, BCDEB на *рис.* П3.

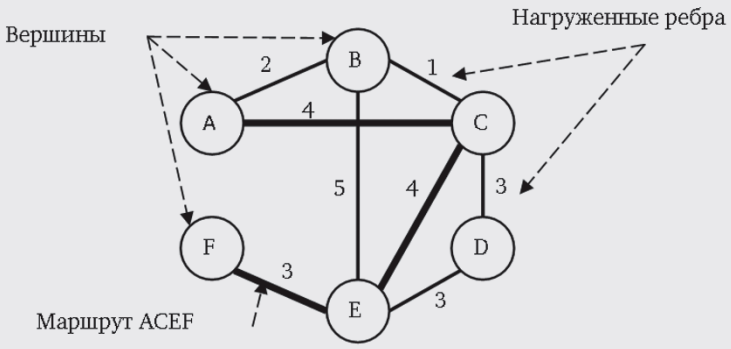


Рис. П3. Пример неориентированного графа

Маршрут называется **цепью**, если каждое ребро содержится в нем не более одного раза. Иными словами, **цепь — маршрут с различными ребрами**. Маршруты ACEF и BCDEB — цепи, а маршрут ABCDECBA — не цепь.

Цепь называется **простой**, если она проходит через каждую свою вершину не более одного раза. Иными словами, **простая цепь — маршрут с различными вершинами**. На *рис.* П3 цепь ACEF — простая, цепь ACEDCB — не простая.

Цепь, являющаяся циклическим маршрутом, называется **циклом**, например, ACEDCBA.

Если цикл является простой цепью, он называется **простым**, например, BCDEB.

Две вершины называются **связанными**, если существует цепь, начинающаяся в одной из них и заканчивающаяся в другой. Граф называется **связным**, если любые две его вершины связаны.

Способы представления графов

Графическое изображение графа наиболее удобно для восприятия (*рис.* П4*а*).

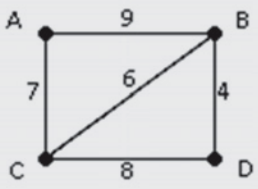
Граф может быть задан также таблицами, описывающими связи между вершинами. Количество строк и столбцов таблицы равно количеству вершин графа. В клетках таблицы на пересечении строки и столбца, соответствующих вершинам *X* и *Y*, указывается 1, если вершины *X* и *Y* связаны ребром, или 0, если между вершинами *X* и *Y* нет ребра. Такая таблица называется таблицей, или **матрицей смежности** (*рис.* П4*б*).

Для неориентированного графа матрица смежности симметрична, для ориентированного графа матрица смежности несимметрична.

Для нагруженного графа в клетках таблицы указывается вес, или нагрузка (расстояние, время и т.п.) ребер, соединяющих вершины. Такая таблица называется **весовой матрицей** (*рис.* П4*в*).

Графическая форма

а)



Матрица смежности

б)

	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	1
D	0	1	1	0

Весовая матрица

в)

	A	B	C	D
A		9	7	
B	9		6	4
C	7	6		8
D		4	8	

Рис. П4. Формы представления неориентированного нагруженного графа

Граф на *рис.* П4 может быть моделью четырех населенных пунктов, связанных дорогами заданной длины. Вершины обозначают населенные пункты, расстояния между ними являются весами ребер.

Вопросы по информатике для проведения школьных конкурсов “Что? Где? Когда?” и “Брейн-ринг”

Д.М. Златопольский,
Москва

1. Одну из радиостанций, вещающих в FM-диапазоне, — “Серебряный Дождь” — можно назвать “двоичной”. Почему?

Вариант вопроса

К так называемому “FM-диапазону” работы радиостанций относятся частоты от 88 до 108 мегагерц.

Одну из радиостанций, вещающих в этом диапазоне, — “Серебряный Дождь” — можно назвать “двоичной”. Почему?

2. Может ли быть такое, что при вычитании из меньшего значения большего получается положительное значение?

3. В текстовом редакторе Microsoft Word и некоторых других этот знак препинания может иметь три различных варианта длины. Что это за знак?

Вариант вопроса

В текстовом редакторе Microsoft Word и некоторых других этот символ может иметь три различных варианта длины. Что это за символ?

4. Какой символ в текстовом редакторе Microsoft Word и некоторых других может иметь два вида, которые называют “елочки” или “лапки”?

Вариант вопроса

Какой знак препинания в текстовом редакторе Microsoft Word и некоторых других может иметь два вида, которые называют “елочки” или “лапки”?

5. Пол комнаты, стол, колено, карман — что объединяет эти понятия?

Вариант вопроса

Пол комнаты, стол, колено. Назовите следующий элемент в этой последовательности.

6. В украинском языке есть слово *абетка*. Как оно переводится на русский язык?

Вариант вопроса

Этому русскому слову в украинском языке аналогом является слово *абет-*

ка, а в английском — слово *alphabet*. Какое это слово?

7. Вы знаете, что такое “крестовая отвертка”? А “обычная отвертка”? Как, по-вашему, назвал бы эти отвертки, чтобы отличить их друг от друга, любитель математики и информатики?

Примечание. При наличии указанных отверток их целесообразно показать участникам конкурса.

8. Хотя они не имеют возраста, расположены они так: чем левее, тем старше. Что именно?

Вариант вопроса

Хотя у них нет даты рождения, среди них есть старшие и младшие. О чем идет речь?

9. *Ведущий (обращаясь к одному из участников конкурса)*: “Какой символ будет отображен на экране, если на клавиатуре нажать клавишу с буквой “А” в режиме ввода русских букв? А если нажать клавишу с буквой “Б”? — Спасибо (*после ответа*)”. А какой символ отображается на экране, когда на клавиатуре нажата клавиша, на которой изображен не этот символ?

10. Они могут располагаться справа вверху или/и справа внизу, но не слева. Что это?

11. Этот символ можно назвать “невидимым” — при нажатии соответствующей ему клавиши на экране ничего не отображается. Что это за символ?

12. По одной из версий, английское название этого компьютерного термина (а его русский вариант звучит аналогично) представляет собой сокращение от двух слов, в переводе на русский язык означающего “по восемь”. Что это за термин?

13. Когда группу из десяти человек можно разбить на две группы, в одной из которых 7 человек, а в другой — 9?

Вариант вопроса

Посмотрите на экран и ответьте, пожалуйста, на представленный там вопрос:

Когда группу из 10 человек можно разбить на две группы, в одной из которых 7 человек, а в другой — 9?

14. Винни Пух нашел листок со следующим изображением:

Пожалуйста, помогите Пуху разобраться, что изображено на листке.

1-й вариант вопроса

Винни Пух нашел обрывок листка со следующим изображением:

Пожалуйста, помогите Пуху разобраться, что изображено на этом обрывке листка.

2-й вариант вопроса

Винни Пух нашел обрывок листка со следующим изображением:

Он понял, что на нем изображен фрагмент примера сложения двух чисел, записанных в системе счисления, в которой используются две цифры — “капуста” и “морковка”. А вот на вопрос: “Может ли капуста соответствовать единице, а морковка — нулю?” — у Пуха ответа не было. А у вас есть?

15. В информатике эти листы находятся не на дереве и не на кустарнике, а в одной из программ. Что это за листы?

Вариант вопроса

В информатике эти листы находятся не на дереве и не на кустарнике, а в одной из офисных программ. Что это за листы?

16. Посмотрите, пожалуйста, на картинку:

Как, по-вашему, что на ней изображено?

17. Какое число, по вашему мнению, должно быть записано вместо вопросительного знака в таблице:

93	10010011
64	?

Вариант вопроса (отличается содержанием таблицы)

93	10010011
51	01010001
64	?

18. Какое число, по вашему мнению, должно быть записано вместо вопросительного знака в таблице:

2	0101
8	1011
5	?

Вариант вопроса (отличается содержанием таблицы)

2	0101
8	1011
3	0110
5	?

Ответы

1. Потому что она работает на частоте, выражаемой с использованием цифр 0 и 1 (100,1 мегагерц).

2. Это может быть при вычитании значений времени — например, при вычитании из 1 часа (ночи) 23 часов получится 2.

3. Этот символ/знак препинания — черточка (дефис — короткий вариант, тире — среднее и длинное).

4. Этот символ/знак препинания — кавычки, которые могут выглядеть как «» (“елочки”) или “” (“лапки”).

Примечание. При ответе ведущим целесообразно показать участникам конкурса, как выглядят указанные два вида кавычек.

5. В списке перечислены, так сказать, места размещения различных поколений и типов компьютеров: первые ЭВМ размещались на *полу помещения*, персональные компьютеры — на *столе*; ноутбуки иногда называют “*наколенными* компьютерами”, а портативный персональный компьютер предельно малых размеров называют КПК (*карманный персональный* компьютер).

Ответ на вариант вопроса: *карман*.

6. Русскими аналогами украинского слова *абетка* являются *азбука*, *алфавит*.

Ответ на вариант вопроса

Это слово “алфавит”, или “азбука”. Украинское слово *абетка* происходит от двух первых букв (*а* и *бе*), английское слово *alphabet* образовано от двух первых букв греческого алфавита (альфа и бета).

7. “Плюсовая” и “минусовая”.

8. Речь идет о разрядах записи числа, в том числе об отдельных битах двоичного представления чисел.

Ответом на вариант вопроса могут быть также байты адреса оперативной памяти (“старший байт” и “младший байт”).

9. Это символ “*” (“звездочка”) или “•” (“кружочек”). Именно они отображаются на экране при нажатии буквенных или цифровых клавиш при вводе пароля.

10. Это верхний и нижний индексы.

Примечание. Ответы “показатель степени” и “основание системы счисления” можно считать правильными, так как в вопросе речь идет об объектах, которые могут присутствовать как одновременно (есть союз “и”), так и отдельно (союз “или”). Пример одновременного применения: C_m^n.

11. Это пробел.

12. Это слово “байт” (byte) — сокращение от слов “by eight” (“по восемь”).

Источник: *http://dic.academic.ru/dic.nsf/ntes/372*.

13. Группу из десяти человек разбить на две группы, в одной из которых 7 человек, а в другой — 9, можно, когда имеется в виду разбиение на группы по каким-то двум признакам, и при этом 6 человек из десяти имеют оба эти признака. Такими признаками могут быть:

1) умение играть в шахматы (7 человек) и в шашки (9 человек), при этом 6 человек умеют играть в обе игры;

2) владение одним иностранным языком (7 человек) и другим (9 человек), при этом 6 человек владеют обоими этими языками, и т.п.

Ответ на вариант вопроса

1-й возможный ответ — см. выше.

2-й возможный ответ — когда все три числа, фигурируемые в вопросе, записаны в шестнадцатеричной системе счисления.

Примечание. Так как о шестнадцатеричном числе 10 (см. 2-й возможный ответ) нельзя *сказать* “десять”, то вопрос должен обязательно предъявляться участникам конкурса на экране компьютера или в письменном виде.

14. На листке изображен пример сложения двух чисел, записанных в системе счисления, в которой используются две цифры — “капуста” (соответствует нулю) и “морковка” (соответствует единице).

Ответ на 1-й вариант вопроса

Здесь возможны два варианта:

1) “капуста” соответствует нулю, а “морковка” соответствует единице;

2) “капуста” соответствует единице, а “морковка” — нулю (если в крайний справа разряд перешла “в уме” единица из соседнего, не представленного на обрывке листа, разряда).

Ответ на 2-й вариант вопроса

Может (см. ответ на 1-й вариант вопроса).

15. Это листы электронной таблицы Microsoft Excel.

16. На картинке изображена нижняя половина примера деления числа 88 на 8.

17. 01100100 (в правом столбце таблицы записаны числа из левого столбца в двоично-десятичной системе счисления, в которой каждая цифра десятичного числа представляется в виде соответствующего четырехзначного двоичного числа).

18. 1000 (в правом столбце таблицы записаны цифры из левого столбца в так называемом “двоично-десятичном коде с избытком 3”, в котором каждая цифра десятичного числа представляется как двоичное число $b = a + 0011$, где a — четырехзначное двоичное число, соответствующее данной цифре).

Примечание. Правильным следует считать ответ, в котором будет указано на двоичное представление цифр из левого столбца (с учетом добавления двоичного числа 11, или десятичного числа 3).

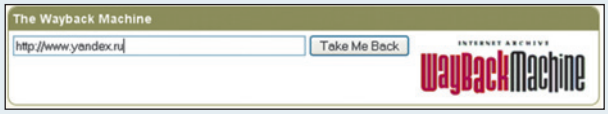
В Интернет — на “машине времени” ☺

Д.Ю. Усенков

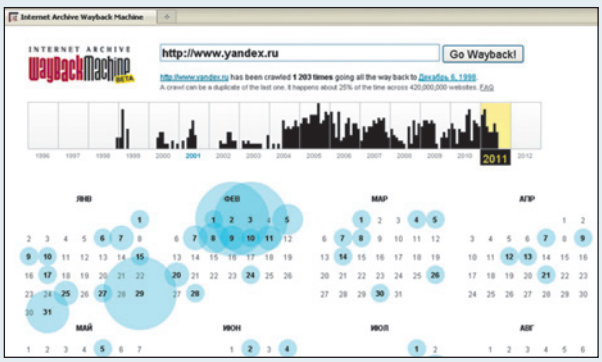
Как известно, Интернет очень скоротечен. Сайты в нем меняются довольно быстро, почти у нас на глазах их внешний вид может измениться до неузнаваемости. А как быть, если хочется увидеть тот или иной сайт таким, каким он был несколько лет тому назад? Есть ли в Интернете такая “виртуальная машина времени”?

Оказывается, есть! Сервер под названием “Wayback Machine” (работающий с 1996 года) действительно позволяет “вернуться назад, в прошлое” и не только увидеть, каким был интересующий вас сайт раньше, но и даже посетить сайт, который сегодня уже не существует. Этот интересный проект создан некоммерческой организацией Internet Archive и компанией Alexa, которая предоставляет серверу “Wayback Machine” архивную информацию.

Чтобы при помощи “Wayback Machine” вернуться в прошлое, достаточно указать в браузере адрес *http://archive.org/web/web.php*, в самом верхнем поле под названием “The Wayback Machine” ввести URL желаемого сайта и нажать кнопку “Take Me Back”. Например, вот так можно посмотреть, каким был когда-то популярный поисковый сервер “Яндекс”:



В ответ сервер “виртуальной машины времени” выведет на экран шкалу лет, на которой столбиками гистограммы отмечено количество снятых в разные месяцы копий заданного сайта. А для текущего выбранного года ниже дан календарь, в котором синими кружочками различного диаметра отмечены даты снятия копий этого сайта.

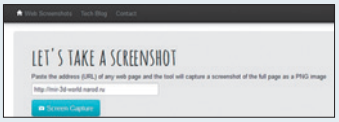


А дальше все делается так же, как при использовании сервиса “Восстановление системы” в Windows: достаточно в календаре щелкнуть мышью на помеченной дате, и в браузере откроется сохраненная копия искомого сайта — именно такая, каким он был в выбранный день. Возможно, некоторые ссылки на нем будут неработоспособными, но по крайней мере визуальное впечатление вы сможете оценить в полной мере.

Вот, например, как выглядел поисковик “Яндекс” 25 января 1999 года, — так просто и… немного трогательно…☺

Тем же, кому важно сохранить у себя графическую копию какого-нибудь сайта в его нынешнем, текущем состоянии, пригодится другой web-сервис. Как известно, стандартный способ — при помощи привычной всем клавиши PrintScreen позволяет поместить в буфер обмена (для последующей вставки в любой графический редактор) только то, что видно на экране дисплея в данный момент. А если web-страница длинная и/или широкая и просматривается только с использованием линейки прокрутки, то приходится долго возиться с копированием и “склейкой” ее отдельных частей.

Избавить от этой рутинной работы может сайт Web Screenshot (*http://ctrlq.org/screenshots*). Достаточно ввести в специально предусмотренном поле вместо указанного там примера URL требуемой web-страницы:



и нажать кнопку “Screen Capture”, и в верхней части страницы сайта Web Screenshot появится уведомление, что работа выполнена:

Остается нажать кнопку “Download Image” (а еще лучше — щелкнуть на ней правой кнопкой мыши и выбрать в контекстном меню команду “Сохранить объект как”) и загрузить графический файл (формата PNG с прозрачным фоном) с желаемой копией указанной вами web-страницы:

ИСТОРИЯ ИНФОРМАТИКИ

Этот зловредный “bug”

В.В. Шилов,
Москва

Легенды о происхождении тех или иных компьютерных терминов многочисленны и крайне живучи. Но все рекорды бьет одна из них — о слове *баг* (*bug*), под которым понимают любую ошибку в аппаратуре или программе (отсюда же *debugging* — отладка). В десятках журнальных и газетных статей, книг, словарей и энциклопедий можно прочитать нечто вроде: “Однажды в середине 1940-х гг. в работе предка современных компьютеров, релейной вычислительной машины Mark I, которую строили в Гарвардском университете, произошел сбой. Его причиной стал мотылек, который забился в одно из реле. Дежурный инженер извлек мотылька пинцетом, и с тех пор гарвардские ученые, когда в машине возникали те или иные неисправности, говорили: “Давай поищем жучка (bug)”. Постепенно этот термин прижился и получил широкое распространение”.

Особый вклад в популяризацию этой истории внесла выдающийся программист, первая женщина, дослужившаяся до звания адмирала флота США, Грейс М. Хоппер (1906–1992). Математик с университетским образованием, она преподавала в одном из колледжей Нью-Йорка, а во время Второй мировой войны добровольцем пошла в армию. Конечно, ее служба в армии (а точнее, на флоте) была связана со специальностью — математики занимались вычислением различных таблиц, необходимых морякам, летчикам и артиллеристам. Одна из первых построенных в США вычислительных машин — еще не электронных, а электромеханических (релейных) — Mark I как раз предназначалась для решения таких задач. Так лейтенант Грейс Хоппер, которая решала на этой машине свои уравнения, впервые вошла в мир вычислительной техники. Вошла, чтобы остаться в нем на всю жизнь.

По словам Грейс Хоппер, жизнь злосчастного мотылька оборвалась летом 1945 года, и это событие сослужило инженерам и математикам неплохую службу. Во-первых, пока мотылька вытаскивали из реле и пока реле почистили, они смогли немного передохнуть. А во-вторых, в их лексиконе появилось новое выражение. Когда в машинный зал входил грозный начальник — создатель Mark I Говард Айкен (1900–1973) — и спрашивал, почему его подчиненные не работают, а бьют баклуши, те дружно отвечали: “А мы очищаем машину от насекомых!” (по-английски “We are debugging the computer!”). Вот так, по словам Грейс Хоппер, и родилось слово *bug*.

Здесь надо заметить, что описанное выше событие действительно имело место и даже было задокументировано! Тот самый мотылек был не только аккуратно вытащен, но и высушен, и вклеен в рабочий журнал (сегодня этот журнал находится в одном из компьютерных музеев — см. *рис.* 1). Пристальный взгляд на страницу журнала убеждает нас

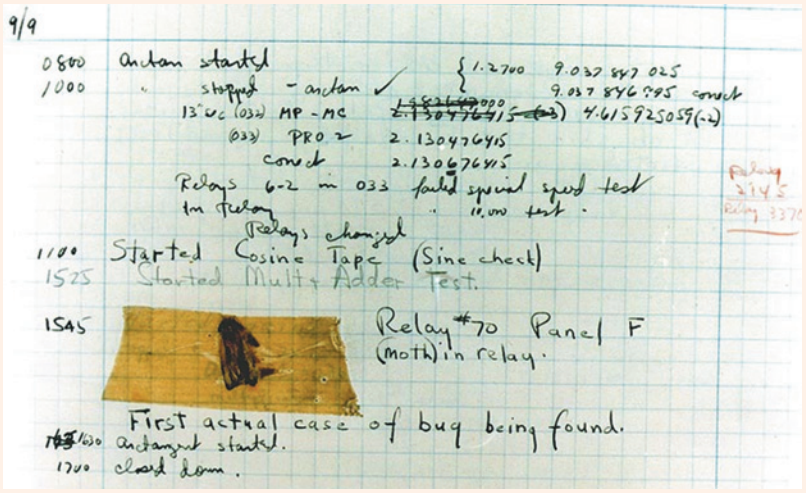


Рис. 1. Первый “bug” (страница из рабочего журнала)

в том, что хорошо известно историкам: любое свидетельство очевидца не может считаться достоверным фактом, пока не подкреплено документами… Ведь из записи следует, что случилась вся эта история не летом 1945 года, а двумя годами позже — 9 сентября 1947 года.

Правда, не слишком понятно, почему мотылек (по-английски *moth*) был вдруг назван "жуком"! Ведь *bug* — это даже не просто насекомое, а насекомое кровососущее, в первую очередь клоп. Разумеется, инженер не обязан быть специалистом в энтомологии[1], но разницу между бабочкой и клопом он все-таки обычно знает…

Чтобы разобраться, давайте прочитаем соответствующую запись в рабочем журнале полностью: "1545 Реле #70 Приборная панель F (мотылек) в реле. Первый случай, когда был найден настоящий bug". Последняя фраза неопровержимо свидетельствует о том, что слово *bug* к моменту обнаружения злосчастного мотылька уже было в ходу среди инженеров. Более того, обращение к различным источникам — техническим публикациям и словарям, показывает, что оно широко использовалось ими уже многие десятилетия!

И одним из первых его стал применять великий американский изобретатель Томас А. Эдисон (1847–1931). Он называл так различные неисправности, вызывающие ошибки в работе приборов и устройств, — как небольшие, которые просто требуется устранить, так и более серьезные, причины которых еще только предстоит установить. Вполне допустимо предположить, что многочисленные и неистребимые мелкие неисправности вызывали у инженеров ассоциации со столь же надоедливым и неприятным насекомым — клопом! Так или иначе, известно, что Эдисон использовал слово *bug* в одном из своих писем, написанных еще в 1878 г. В 1889 г. оно впервые попало на страницы печати: репортер английской газеты "Pall Mall Gazette" 11 марта сообщил читателям, что "Мистер Эдисон провел две бессонных ночи, отыскивая *bug* в своем фонографе[2]" (на *рис*. 2 мы как раз видим расстроенного и задумавшегося изобретателя рядом с неисправным фонографом). Приблизительно в это же время слово *bug* было включено в словарь английского языка в значении "неисправность при работе телеграфной или другой электрической аппаратуры".

Таким образом, смысл приведенной выше записи в рабочем журнале становится вполне понятным. Инженеры, постоянно занятые устранением ошибок в работе оборудования, называли их клопами (*bug*). И вдруг оказалось, что причиной ошибки в самом деле является насекомое (пусть даже совсем другого сорта)! Конечно, совпадение показалось

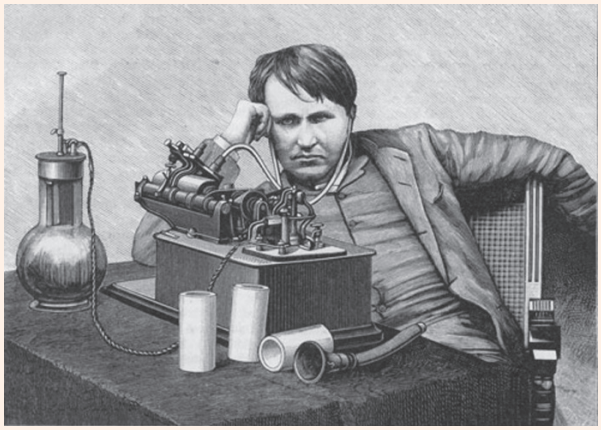


Рис. 2. Томас А. Эдисон
(очевидно, озабоченный поиском неисправностей ☺)

забавным, что и было отмечено в рабочем журнале. И, скорее всего, если бы не Грейс Хоппер, то вряд ли этот эпизод привлек бы к себе чье-либо внимание.

Вообще говоря, живучесть легенд — факт крайне удивительный. Версию Грейс Хоппер не раз пытались опровергнуть, но безуспешно. Например, один из создателей электронно-вычислительной машины ENIAC Преспер Эккерт (1919–1995) в данном в 1986 г. интервью так ответил на вопрос, что он думает о происхождении термина *bug*: "Я знаю, что думает об этом Грейс Хоппер. Она много раз рассказывала свою сказку. Насколько я знаю, инженеры — и механики, и электрики, называли так проблемы в работе оборудования задолго до того, как Грейс Хоппер услышала о самом существовании таких проблем. Это слово было новостью только для Грейс. Я никогда не говорил ей, что ее рассказ — это ерунда, но это все же полная ерунда. Слово давно было в ходу".

Слова Эккерта подтверждает другая запись в рабочем журнале разработчиков Mark I. 17 апреля 1944 г. в нем были зафиксированы "проблемы с тестированием оборудования" и что Гарвард посетил консультант от компании IBM, чтобы "помочь нам обнаружить неисправности (*bugs*)".

Да и сама Грейс Хоппер пользовалась этим словом задолго до гибели мотылька! На *рис*. 3 на с. 50 показан лист с четырьмя шуточными рисунками, сделанными рукой самой Хоппер в июле 1945 года. На них изображены "баги" трех видов. Первый — "table warm" (т.е. "табличный червяк"), представленный в виде куска перфоленты, которая использовалась для ввода чисел в вычислительную машину. Ошибки на перфоленте, конечно, влекли за собой сбои в работе ЭВМ. Второй — это "Kitchie Boo Boo Bug". Перевести это название можно, крайне приблизительно, как "дурацкая ошибка забодаю-забодаю". Этот баг специализировался на выводе из строя реле. Третий, со строгим именем "NRL bug" (NRL — это ***N**aval **R**esearch **L**aboratory*, научно-исследовательская лаборатория военно-морского флота, в интересах которого велись вычисления), занимался тем, что портил результаты

[1] Энтомология — раздел зоологии, изучающий насекомых. — **Прим. ред.**

[2] Фонограф — первый прибор для записи и воспроизведения звука, изобретенный Эдисоном.

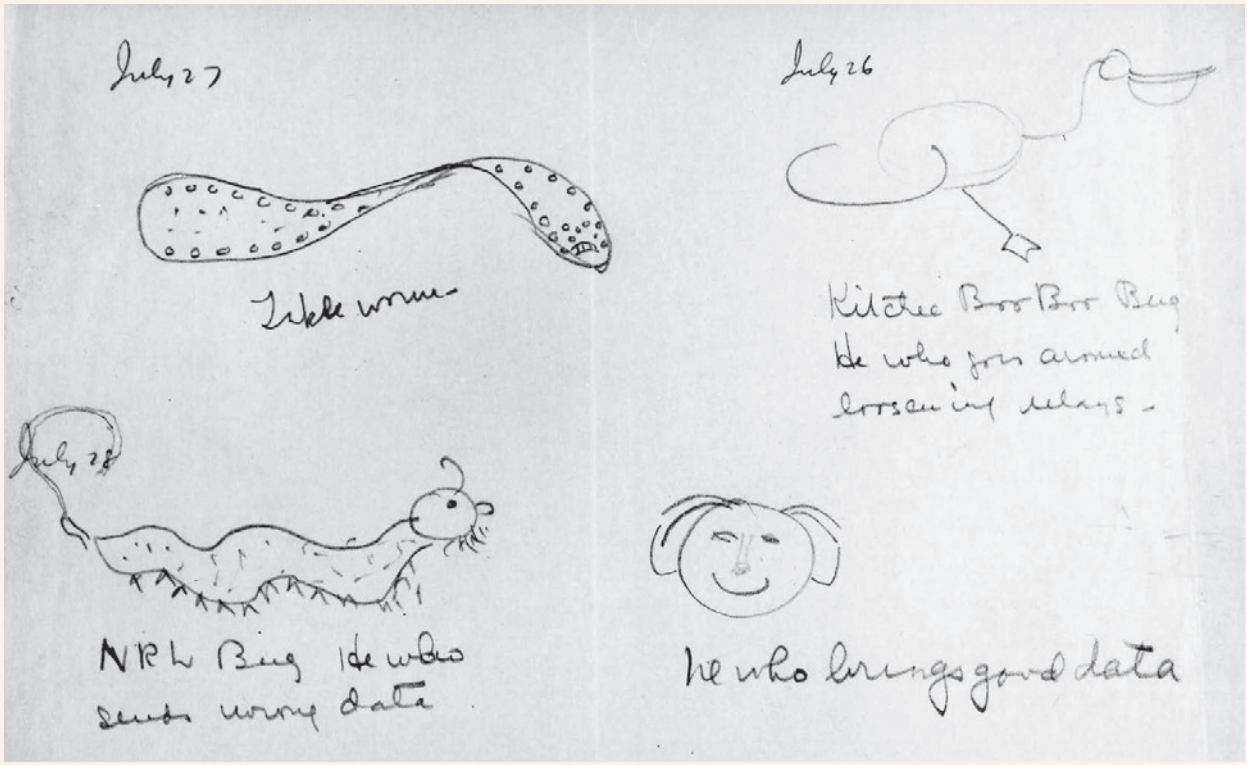


Рис. 3. Три вида багов ☺ (рисунок Грейс Хоппер)

расчетов. А улыбающееся лицо в правом нижнем углу рисунка — это уже не баг, а "тот, кто выдает правильные данные".

Таким образом, вопреки распространенному мнению, вовсе не погибший мотылек дал имя компьютерному термину (или даже двум!). На самом деле это были давно известные и широко распространенные в среде инженеров слова. А о мотыльке никто не вспоминал несколько десятилетий — пока о нем не вспомнила адмирал Грейс Хоппер.

Впрочем, путь слова *bug* в компьютерный лексикон отнюдь не был устлан ковровыми дорожками. Потребовалось немало времени, чтобы слово из жаргона инженеров-электриков было окончательно "принято" всеми компьютерщиками. Так, в первом компьютерном словаре, составленном в 1951 г. Институтом радиоинженеров (IRE), оно отсутствовало. Вскоре в одном из журналов впервые появилось слово *debug*, которое объяснялось как устранение неисправностей в работе аппаратуры компьютера. Спустя три года Г.Хоппер опубликовала свой словарик, в котором расширила значение понятия *debug*, включив в него также устранение ошибок в программах. Этот словарик лег в основу нескольких официальных изданий словаря компьютерных терминов, но слова *bug* ни в одном из них, как и у самой Хоппер, не было! И попало оно в него только в 1962 г.

Зато сегодня оно обрело права гражданства не только в английском, но и в других языках: например, во французском оно приняло вид *bogue*.

Казалось бы, вопрос о происхождении популярного компьютерного термина полностью выяснен. Однако в действительности первыми не были не только Грейс Хоппер, но даже Эдисон и его коллеги. Известно, что слово bug в значении "непредусмотренное событие, имеющее негативные последствия", использовал в своих пьесах Шекспир!

Рис. 4. Грейс Хоппер возле ЭВМ UNIVAC (начало 1950-х гг.)

ЗАДАЧНИК

Восточный календарь — своими руками

Д.М. Златопольский,
Москва

Как, конечно, известно читателям, в некоторых странах Дальнего Востока (Китае, Японии и др.) использовался (и неофициально используется в настоящее время) календарь, отличающийся от применяемого нами. Этот календарь представляет собой 60-летнюю циклическую систему. Каждый 60-летний цикл состоит из пяти 12-летних подциклов. В каждом подцикле года носят названия животных: Крыса, Корова, Тигр, Заяц, Дракон, Змея, Лошадь, Овца, Обезьяна, Петух, Собака и Свинья. Кроме того, в названии года фигурируют цвета животных, которые связаны с пятью элементами природы — Деревом (зеленый), Огнем (красный), Землей (желтый), Металлом (белый) и Водой (черный). В результате каждое животное (и его год) имеет символический цвет, причем цвет этот часто совершенно не совпадает с его "естественной" окраской — Тигр может быть черным, Свинья — красной, а Лошадь — зеленой. Например, 2012 год является годом Черного Дракона. Каждый цвет в цикле (начиная с зеленого) "действует" два года, поэтому через каждые 60 лет имя года (животное и его цвет) повторяется[3]. Новый год по древнему восточному календарю наступает в один из дней в промежутке между 21 января и 20 февраля.

Сегодня в Интернете можно легко найти информацию о том, какому году по восточному календарю соответствует некоторый год. А не хотите ли сами разработать и применять средство для решения этой задачи? Если да, то эта статья для вас.

В дальнейших рассуждениях год нашего летосчисления будем обозначать как *наш_год* ☺.

Итак, этапы решения задачи.

1. Перевести год нашего летосчисления (*наш_год*) в номер года в 60-летнем цикле. Так и назовем рассчитываемую величину — *номер_года*. Этот номер определяется как остаток от деления (*наш_год* + 2397) на 60[4] [1]. Но нетрудно увидеть, что, когда сумма (*наш_год* + 2397) кратна 60, значение *номер_года* будет равно нулю, хотя фактически номер соответствующего года в 60-летнем цикле равен 60. Поэтому можем записать такое правило:

```
если остаток от деления (наш_год + 2397)
     на 60 не равен 0
  то
    номер_года равен остатку от деления
    (наш_год + 2397) на 60
  иначе |остаток равен 0
    номер_года равен 60
все
```

2. Определить, какому животному соответствует значение *номер_года*. Если сопоставить 12 названиям животных следующие номера: 1 — "Крыса", 2 — "Корова", 3 — "Тигр", 4 — "Заяц", 5 — "Дракон", 6 — "Змея", 7 — "Лошадь", 8 — "Овца", 9 — "Обезьяна", 10 — "Петух", 11 — "Собака", 12 — "Свинья", то можно составить таблицу:

номер_года	*номер_животного*
1, 13, 25, 37, 49	1
2, 14, 26, 38, 50	2
…	…
11, 23, 35, 47, 59	11
12, 24, 36, 48, 60	12

Что общего у чисел 1, 13, 25, 37, 49? Ответ — у всех них остаток от деления на 12 равен 1. Проанализировав и другие группы чисел в левом столбце таблицы, можно утверждать, что

```
если остаток от деления номер_года
     на 12 не равен 0
  то
    номер_животного равен остатку
    от деления номер_года на 12
  иначе    |остаток равен 0
    номер_животного равен 12
все
```

А после этого можно по рассчитанному номеру животного определить его название:

номер_животного	Название
1	Крыса
2	Корова
3	Тигр
4	Заяц
5	Дракон
6	Змея
7	Лошадь
8	Овца
9	Обезьяна
10	Петух
11	Собака
12	Свинья

[3] Почему 60, а не 120 (12 × 2 × 5) — подумайте самостоятельно.

[4] Начало отсчета ведется от 2397 года до н.э.

3. Определить, какому цвету соответствует значение *номер_года*. Присвоим цветам, используемым в календаре, следующие номера: “зеленый” — 1, “красный” — 2, “желтый” — 3, “белый” — 4, “черный” — 5 и составим таблицу, в которой укажем номер цвета, соответствующий тем или иным значениям величины *номер_года*:

номер_года	*номер_цвета*
1, 2, 11, 12, …, 51, 52	1
3, 4, 13, 14, …, 53, 54	2
5, 6, 15, 16, …, 55, 56	3
7, 8, 17, 18, …, 57, 58	4
9, 10, 19, 20, …, 59, 60	5

Анализ последней таблицы показывает, что номер цвета зависит от остатка от деления номера года на 10. Если этот остаток обозначить *остаток*10, то закономерность для определения значения *номер_цвета* такая:

```
при остаток10 = 1 или
    остаток10 = 2: номер_цвета = 1
при остаток10 = 3 или
    остаток10 = 4: номер_цвета = 2
при остаток10 = 5 или
    остаток10 = 6: номер_цвета = 3
при остаток10 = 7 или
    остаток10 = 8: номер_цвета = 4
при остаток10 = 9 или
    остаток10 = 0: номер_цвета = 5
```

А нельзя ли упростить эту зависимость значения *номер_цвета* от величины *остаток*10? Оказывается — можно! Вот она[5]:

```
если остаток10 не равен 0
  то
    номер_цвета равен целой части
    частного от деления (остаток10 + 1)
    на 2
  иначе
    номер_цвета равен 5
все
```

Итак, номер цвета найден. Но надеемся, что в ходе сделанных сложных (точнее — объемных) рассуждений читатели не забыли, что цель этого этапа — определить не значение номера цвета, а название последнего. Но это уже можно сделать достаточно просто по такой таблице:

номер_цвета	Название
1	“Зеленый”
2	“Красный”
3	“Желтый”
4	“Белый”
5	“Черный”

Вот и все!

[5] Убедитесь в ее справедливости самостоятельно!

Можно автоматизировать (точнее — компьютеризировать ☺) вычисления, используя электронную таблицу Microsoft Excel или др.

Оформим лист этой программы, с помощью которого можно по номеру года определить, какому животному и какому цвету по восточному календарю он соответствует. Верхнюю часть листа оформим так:

	A	B	C	D
1	Введите номер года:	2011		
2	Этому году соответствует:			
3		*животное*	*цвет*	
4				

Видно, что в ячейке B1 происходит ввод номера года, а в ячейках B2 и C2 — вывод результатов “расчетов”. Так как все остальные величины (условные номера цветов и животных и др.) являются вспомогательными, их расчет и вывод будем проводить где-нибудь вне зоны видимости листа, например, в столбцах L-O:

	…	L	M	N	O
1		Номер_года		Наш_год + 2397	
2		Номер_животного		остаток12	
3		Номер_цвета		остаток10	
4					

В соответствии с рассуждениями, сделанными выше, запишем:

1) в ячейку O1 — формулу: =B1 + 2397;

2) в ячейку M1 — формулу для нахождения значения *номер_года*:

=ЕСЛИ(ОСТАТ(O1; 60) <> 0; ОСТАТ(O1; 60); 60),

где ОСТАТ — функция, возвращающая остаток от деления ее первого аргумента на второй;

3) в ячейку O2 — формулу для расчета остатка от деления *номер_года* на 12:

=ОСТАТ(M1; 12);

4) в ячейку M2 — формулу для определения условного номера животного:

=ЕСЛИ(ОСТАТ(O2; 12) <> 0; ОСТАТ(O2; 12); 12);

5) в ячейку O3 — формулу для расчета значения *остаток*10:

=ОСТАТ(M1; 10);

6) в ячейку M3 — формулу для нахождения значения *номер_цвета*:

=ЕСЛИ(O3 <> 0; ЦЕЛОЕ((O3 + 1)/2); 5),

где ЦЕЛОЕ — функция, возвращающая целую часть числа — ее аргумента.

И только теперь можно получить то, что требуется. Название животного в ячейке B2 можно получить, применив функцию ВЫБОР:

=ВЫБОР(M2; “Крыса”; “Корова”; “Тигр”; “Заяц”; “Дракон”; “Змея”; “Лошадь”; “Овца”; “Обезьяна”; “Петух”; “Собака”; “Свинья”)

С особенностями оформления и работы этой функции ознакомьтесь самостоятельно.

С помощью функции ВЫБОР можно определить также название цвета в ячейке C2:

=ВЫБОР(M3; “Зеленый”; “Красный”; “Желтый”; “Белый”; “Черный”)

Читатели, владеющие каким-либо языком программирования, для решения обсуждаемой задачи могут также разработать соответствующую программу.

Задания для самостоятельной работы

1. Оформив лист электронной таблицы Microsoft Excel или/и разработав компьютерную программу, определите, какому году по восточному календарю соответствуют:

1) 2013-й и 2014 года;

2) 1949 год;

3) год вашего рождения.

В компьютерной программе вывод ответа должен быть оформлен в виде:

```
Этот год по восточному календарю:
Животное:  Овца
Цвет: Желтый
```

2. Разработайте вариант программы или листа электронной таблицы Microsoft Excel, в которых результат будет выдаваться в виде “Это год Зеленой Собаки” и т.п. (с соответствием цвета женскому или мужскому роду названия животного).

Успехов!

Литература

1. *Климишин А.И.* Календарь и хронология. М.: Наука, 1990.

Сын профессора Алгоритмова

Профессор Бит Байтович Алгоритмов чрезвычайно гордится своим гениальным, по его мнению, сыном.

Он любит рассказывать, что:

1) день рождения сына является числом Фибоначчи;

2) сумма цифр числа — дня рождения — и произведение его цифр также являются числами Фибоначчи;

3) хотя порядковый номер месяца, в котором родился сын, и не является числом Фибоначчи, зато является произведением двух не соседних чисел Фибоначчи;

4) год рождения сына представляет собой удвоенное число Фибоначчи.

Определите точную дату рождения сына уважаемого профессора, а именно, укажите соответствующий день, месяц и год его рождения.

Числами Фибоначчи называют числа, образующие последовательность 1, 2, 3, 5, 8, … (каждый член последовательности, начиная с третьего, равен сумме двух предыдущих).

Задачу предложил Лейб Штейнгарц, г. Иерусалим, Израиль

Блондины и брюнеты в классе

В классе девочек-блондинок столько же, сколько мальчиков-брюнетов. Кого в классе больше, девочек или учащихся с темными волосами? Принять, что шатенов и шатенок, а также учащихся с другим цветом волос в классе нет ☺.

Пятеро друзей в сети

Пятеро друзей (Павел, Бадри, Семен, Ахмед и Владимир) сидят за своими домашними компьютерами. По каналам связи они могут обмениваться информацией. Скорость передачи информации между компьютерами (килобайт в секунду) отражена в таблице:

	Компьютер Павла	Компьютер Бадри	Компьютер Семена	Компьютер Ахмеда	Компьютер Владимира
Компьютер Павла	–	25	50	125	20
Компьютер Бадри	25	–	100	40	100
Компьютер Семена	50	100	–	100	40
Компьютер Ахмеда	125	40	100	–	25
Компьютер Владимира	20	100	40	25	–

Павлу необходимо передать Владимиру файл размером 1000 килобайт. Он может передавать файл любому из друзей, те, в свою очередь, — тоже любым друзьям. Однако передавать его можно только тогда, когда он полностью получен (все 1000 килобайт). За какое наименьшее время при таких условиях Павел может передать файл Владимиру?

Еще задачи Анания Ширакаци

Предлагаем читателям решить еще ряд задач из старинного учебника арифметики, написанного в VII веке знаменитым армянским ученым Анания Ширакаци (см. раздел “В мир информатики” в журнале № 3 за 2012 год). Их подготовил Севак Карапетян, ученик гимназии № 1530 г. Москвы.

Задача 1

Был у меня породистый конь; я продал его и на четвертую часть его стоимости купил корову, на седьмую часть — козу и на одиннадцатую часть — быков, а на триста восемнадцать дахеканов я купил овец. Итак, узнай, сколько всего дахеканов это составляет.

Задача 2

Один из моих близких приобрел драгоценные жемчужины в городе Багл. По возвращении домой в Гандзак он продал половину жемчужин по 50 драмов за штуку. Доехав до Нахичевана, продал четверть жемчужин по 70 драмов за штуку. Далее в городе Двин продал двенадцатую часть по 50 драмов. Когда он добрался до Ширака, у него осталось всего 24 жемчужины. Узнай, сколько было всего жемчужин и сколько денег он получил.

Задача 3

Я слышал от отца своего про героические победы Зорака Камсаракана во время войны против персов. Рассказывают, что в течение месяца трижды нападал он на персидское войско. В первый раз он перебил половину войска. Преследуя во второй раз, перебил четверть оставшегося войска, в третий раз — одиннадцатую часть. Оставшиеся в живых, числом двести восемьдесят, убежали в Нахичеван. Посчитай, сколько воинов было в персидском войске в начале войны.

Примечание. Дахекан и драм — старинные армянские денежные единицы (в настоящее время драм — денежная единица Республики Армения).

Расположить ребят по росту

Встретились четыре школьных товарища: Андрей, Борис, Иван и Григорий. Расположите имена ребят в порядке возрастания их роста, если известно, что Борис не самый высокий, но он выше Андрея и Григория, а Андрей ниже Григория.

Задание предназначено для учащихся начальной школы.

На катке

Четыре подруги пришли на каток каждая со своим братом. Они разбились на пары и начали кататься. Оказалось, что в каждой паре “кавалер” выше “дамы”, и никто не катался со своей сестрой. Самый высокий из компании — Юра Воробьев, следующий по росту — Андрей Егоров, потом Люся Егорова, Сережа Петров, Оля Петрова, Дима Крымов, Инна Крымова и Аня Воробьева. Кто с кем катался?

ШКОЛА ПРОГРАММИРОВАНИЯ

Еще раз о фибоначчиевой системе счисления

В “основном” журнале “Информатика” была опубликована статья [1], посвященная так называемой “фибоначчиевой системе счисления” (далее — ФСС). Эта система относится к позиционным системам счисления — в которых вклад той или иной цифры в записи числа в общее значение зависит от места (позиции, разряда), которое занимает эта цифра в записи. Напомним, что в ней для записи чисел используются две цифры — 0 и 1, а ее базисом — последовательностью чисел, каждое из которых задает “вес” соответствующего разряда, являются числа Фибоначчи: 1, 2, 3, 5, 18, 13 (каждое последующее число, начиная с третьего, равно сумме двух предыдущих). Первые 10 весов представлены в табл. 1, а примеры записи нескольких десятичных чисел в ФСС приведены в табл. 2.

Таблица 1

Номер разряда	10	9	8	7	6	5	4	3	2	1
Вес	89	55	34	21	13	8	5	3	2	1

Таблица 2

Десятичное число	**Запись в ФСС**
1	1
2	10
3	100, или 11
4	101
5	1000, или 110
6	1001, или 111
7	1010
8	10000, или 1100, или 1011
9	10001, или 1101
10	10010, или 1110
20	101010
50	10100100, или 10011100, или 10100011, или 10011011

Обратим внимание на тот факт, что некоторые десятичные числа могут быть записаны в ФСС разными способами.

Разработаем программы для перевода чисел из десятичной системы в ФСС и обратно.

1. Перевод чисел из фибоначчиевой системы счисления в десятичную[6]

Напомним, что для позиционных систем счисления перевод чисел из них в десятичную можно провести с помощью так называемой "развернутой формы записи числа" — суммы произведений каждой цифры числа на вес соответствующего разряда.

Примеры:

1) $1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}$;

2) $423_5 = 4 \times 5^2 + 2 \times 5^1 + 3 \times 5^0 = 113_{10}$;

3) $423_8 = 4 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 = 275_{10}$.

Так как в записи числа в ФСС используются две цифры — 0 и 1, то можем упростить правило перевода применительно к этой системе — "сложить веса тех разрядов, в которых записаны единицы". Кстати, такое же правило действует при переводе из двоичной системы в десятичную.

На его основе можно привести пример перевода[7]:

$$101011_{\text{ФСС}} = 13 + 5 + 2 + 1 = 21_{10}.$$

Теперь можем перейти к разработке программы.

Число в ФСС будем представлять в программе в виде строковой (в терминах языка программирования Паскаль) величины. Имя этой величины в приведенной ниже программе — *число_фиб*. Другие основные используемые величины:

— *число*10 — искомое десятичное число;

— *вес* — массив для хранения весов первых десяти (условно) разрядов;

— *вес_i* — вес некоторого конкретного разряда.

Прежде чем представлять программу, заметим, как можно определить вес того или иного разряда, имея в виду, что обработка заданной строки будет проводиться, начиная со старших цифр, а в начале массива *вес* записаны веса младших разрядов. Удобно составить табличку:

Номер цифры	Номер элемента массива с весом соответствующего разряда
длин(число_фиб)	1
длин(число_фиб) – 1	2
длин(число_фиб) – 2	3
…	…
2	длин(число_фиб) – 1
1	длин(число_фиб)

— из которой можно установить, что для *i*-й слева цифры вес определяется так:

```
вес_i := вес[длин(число_фиб) - i + 1]
```

Примечание. длин — функция школьного языка программирования, возвращающая общее число символов в строке — ее аргументе.

[6] Начинаем с этого варианта, как более простого.

[7] Для удобства перевода можно выписывать цифры числа в табличке под весами разрядов.

Итак, программа решения задачи:

```
алг Перевод чисел из фибоначчиевой
    системы в десятичную
нач лит число_фиб, цел число10,
          вес_i, i, цел таб вес[1:10]
  |Заполнение массива вес
  вес[1] := 1
  вес[2] := 2
  нц для i от 3 до 10
    вес[i] := вес[i - 1] + вес[i - 2]
  кц
  |Ввод исходного значения
  вывод нс, "Введите число в ФСС "
  ввод число_фиб
  |Рассчитываем искомую величину
  число10 := 0 |Начальное значение
  |Рассматриваем каждый символ-цифру
  нц для i от 1 до длин(число_фиб)
    |Если это 1
    если число_фиб[i] = "1"
     то
      |Определяем вес данного разряда
      вес_i := вес[длин(число_фиб) - i + 1]
      |и учитываем его в сумме
      число10 := число10 + вес_i
     все
  кц
  вывод нс, "Соответствующее десятичное"
  вывод " число равно ", число10
кон
```

2. Перевод чисел из десятичной системы счисления в фибоначчиеву

Задача может быть решена двумя способами.

Первый из них основан на выделении в базисе (см. выше) максимальных весов, которые "умещаются" сначала в заданном десятичном числе, а затем, при необходимости, в разности между числом и найденным весом.

Опишем этот метод на примере. Пусть дано десятичное число 32. Из табл. 1 можно установить, что максимальный вес, не превышающий 32, равен 21. Это означает, что в записи числа 31 в ФСС в седьмом разряде будет стоять единица. Вычитаем 32 – 21 = 11. Следующий максимальный вес, не превышающий 11, — 8, то есть единица будет стоять в пятом разряде. Процесс повторяется до тех пор, пока уменьшающееся число не станет равно нулю (в нашем случае это произойдет после вычитания весов в первом и втором разрядах). Разрядам, в которых цифра 1 не записывается, естественно, сопоставляется цифра 0. Итак, имеем $21_{10} = 1010011_{\text{ФСС}}$.

Заметим попутно, что аналогичный метод применяется и при переводе чисел из десятичной системы счисления в двоичную (или другую позиционную) систему.

В программе, работающей по описанному методу, искомое число в ФСС (*число_фиб*) получим как строку символов. Для этого отдельные цифры чис-

ла (как символы) предварительно запишем в массив *цифры_фиб* из 10 (условно) элементов.

Еще три особенности программы.

1. Используется новая величина — *осталось* — число, меняющееся в ходе вычислений (сначала оно равно заданному числу).

2. Массив *вес* для хранения весов разрядов целесообразно заполнить так, как показано в табл. 3.

Таблица 3

Вес	89	55	34	21	13	8	5	3	2	1
Номер элемента	1	2	3	4	5	6	7	8	9	10

Почему именно так, проанализируйте самостоятельно.

3. При поиске максимального веса, не превышающего значения *осталось*, возможны два случая (в программе они описаны в комментариях).

Другие используемые величины аналогичны применявшимся в программе решения предыдущей задачи.

Вся программа имеет вид:

```
алг Перевод чисел из десятичной системы
    в фибоначчиеву
нач цел число10, осталось, вес_i, i,
    лит число_фиб, цел таб вес[1:10],
    лит таб цифры_фиб[1:10]
  |Заполнение массива вес нулями
  вес[10] := 1
  вес[9] := 2
  нц для i от 8 до 1 шаг -1
    вес[i] := вес[i + 1] + вес[i + 2]
  кц
  |Заполнение массива цифр
  нц для i от 1 до 10
    цифры_фиб[i] := "0"
  кц
  |Ввод исходного значения
  вывод нс, "Введите десятичное число "
  ввод число10
  осталось := число10
  нц
    |Ищем максимальный вес, не превышающий
    |значения осталось
    i := 10
    нц пока вес[i] < осталось и i >= 1
      i := i - 1
    кц
    |Возможны два случая
    если вес[i] = осталось
    то
      |Номер найденного веса i совпадает
      |с номером разряда с единицей
      |Записываем в этот разряд 1
      цифры_фиб[i] := "1"
      |Вычитаем соответствующий вес
      осталось := осталось - вес[i]
    иначе
      |Разряд с единицей - соседний справа
      цифры_фиб[i + 1] := "1"
      |Вычитаем соответствующий вес
      осталось := осталось - вес[i + 1]
    все
  кц_при осталось = 0
  |Формируем искомое число
  |Ищем первую (ненулевую) цифру
  i := 1
  нц пока цифры_фиб[i] = "0"
    i := i + 1
  кц
  |Из этой и остальных цифр формируем число
  число_фиб := ""
  нц пока i <= 10
    число_фиб := число_фиб + цифры_фиб[i]
    i := i + 1
  кц
  |Выводим ответ
  вывод нс, "Соответствующее число в ФСС"
  вывод "равно ", число_фиб
кон
```

Второй способ перевода чисел из десятичной системы счисления в фибоначчиеву похож на метод последовательного деления на основание, используемый при переводе десятичного числа в двоичную (или другую позиционную) систему с заданным основанием. Проиллюстрируем его также на примере перевода числа 32. Будем последовательно делить нацело величину *осталось* (см. первый метод) на значения весов в массиве *вес*, начиная с его 1-го элемента. Получаемые при этом целочисленные частные будем записывать в массив *цифры_фиб* (теперь с данными целого типа). Если в некотором разряде частное (цифра) равно 1, то уменьшим значение *осталось*.

Соответствующая программа:

```
алг Перевод чисел из десятичной системы
    в фибоначчиеву Вариант 2
нач цел число10, осталось, вес_i, i,
  лит число_фиб, цел таб вес[1:10],
  цел таб цифры_фиб[1:10]
  |Заполнение массива вес
  …
  |Ввод исходного значения
  …
  вывод нс, "Введите десятичное число "
  ввод число10
  осталось := число10
  нц для i от 1 до 10
    цифры_фиб[i] := div(осталось, вес[i])
    если цифры_фиб[i] = 1
    то
      осталось := осталось - вес[i]
    все
  кц
  |Формируем и выводим искомое число
  |Ищем первую (ненулевую) цифру
  i := 1
  нц пока цифры_фиб[i] = 0
    i := i + 1
  кц
```

```
|Из этой и остальных цифр формируем число
число_фиб := ""
нц пока i <= 10
  число_фиб := число_фиб +
               цел_в_лит(цифры_фиб[i])
  i := i + 1
кц
|Выводим ответ
вывод нс, "Соответствующее число в ФСС"
вывод "равно ", число_фиб
кон
```

Задания для самостоятельной работы

1. Разработайте два варианта программы для перевода числа из ФСС в десятичную систему:

1) в котором расчет проводится по “полной” развернутой форме записи числа;

2) в котором заданное число представляется не в виде строковой величины, а как десятичное число (совпадающее с числом в ФСС).

2. Исследуйте, можно ли при переводе числа из ФСС в десятичную систему использовать массив весов, заполненный как при решении задачи 2 (табл. 3). Рационально ли делать это?

3. Установите, какое максимальное десятичное число можно переводить в ФСС по приведенным программам.

4. Разработайте вариант программы решения задачи 2 по второму способу, в котором условный оператор (команда `если`) не используется.

5. Разработав любой вариант программы решения задачи 2, переведите в ФСС следующие десятичные числа:

1) 175; 2) 1000; 3) 5000.

6. Определите, в чем заключаются особенности числа, получаемого по программе решения задачи 2, когда возможны несколько вариантов записи десятичного числа в ФСС.

Результаты присылайте в редакцию (можно выполнять не все задания).

Литература

1. *Фалина И.Н.* Компьютер и фибоначчиева система счисления. / Информатика, № 15/2011.

“ЛОМАЕМ” ГОЛОВУ

Две шарады и логогриф

Н.А. Владимирова,
учитель информатики гимназии № 2,
г. Заозерный Красноярского края

Шара́да (от фр. *charade*) — разновидность загадки, в которой надо отгадать некоторое слово. Это слово разбивается на слоги таким образом, что каждый слог имеет смысл самостоятельного слова. После чего, как и в загадке, дается описание каждого из этих слов-слогов (например, факт + ура = фактура).

Отгадайте, пожалуйста, две шарады.

Шарада № 1

Мой первый слог —
Союз известный и предлог.
Ищи другой слог в огороде,
А целое — вид исполнителя.

Шарада № 2

Три слова в слове:
Первый слог —
большой снеговика кусок.
Осуществляют слог второй
Слоны, придя на водопой.
А третий слог зовется так,
Как прежде звался твердый знак.
Соедини все три, как надо, —
И любимая “игрушка” предстанет пред тобой.

“Логогрифом” называют загадку, в которой задуманное слово может иметь различные значения в результате добавления, пропуска или отбрасывания буквы. Например, форма — формат, роль — рояль, волк — вол, канава — канва.

Отгадайте слово в логогрифе:

Готов я вас напоить водой,
Но если “Э” в начале вставить,
Программа сразу пред тобой
Решение вам предоставит.

Последнее задание предназначено для учеников 1–7-х классов.

Числовой ребус “Лестница с ЗАЗОРОМ”

Решите, пожалуйста, числовой ребус:

```
          Р
        О Р
+     З О Р
    А З О Р
З А З О Р
5 5 5 5 0
```

Как обычно, одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры.

Ребус из звездочек

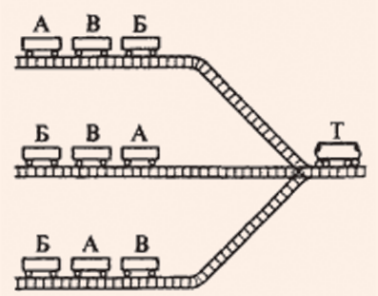
Определить, какие цифры заменены звездочками:

```
* * * * * * * | * *
* * *         | * * 8 * *
-------       |
      * *     |
      * *     |
    -----     |
      * * *   |
      * * *   |
    ---------
        1
```

Звездочкой может быть зашифрована любая цифра (она может повторяться).

Переставить вагоны

На каждом из трех путей стоят вперемешку вагоны с арбузами (А), бананами (Б) и виноградом (В) так, как это показано на рисунке:



Разработайте алгоритм действий машиниста, необходимых для того, чтобы сформировать на каждом из путей составы с одинаковыми плодами, если маневровый тепловоз (Т) может передвигать любое количество вагонов одновременно в любую сторону пути по любому пути. Алгоритм оформите в виде последовательности таблиц, иллюстрирующих изменение положения вагонов на путях, например:

0) исходное положение:

АВБ
БВА
БАВ

1)

АВБ
БВААВ
Б

2) …

Софизм “Все числа равны между собой”

Напомним, что *софизмом* называют рассуждение, умозаключение, “внешне” правильное, но противоречащее общепринятым представлениям (от греч. *sóphisma* — уловка, ухищрение, выдумка, головоломка).

Вот, например, “доказательство” того, что вынесено в заголовок.

Пусть $m \neq n$. Возьмем тождество:

$$m^2 - 2mn + n^2 = n^2 - 2nm + m^2.$$

Имеем $(m-n)^2 = (n-m)^2$. Отсюда $m - n = n - m$, или $2n - 2m$, а значит, $n = m$.

Как такое может быть?

Россия — чемпион! ☺

Представьте себе, что вам приснилась такая таблица розыгрыша финальной части последнего чемпионата Европы:

	В	Н	П	Мячи	Очки
Россия	2	1	0	9–1	5
Испания	2	1	0	5–1	5
Италия	1	0	2	2–9	2
Португалия	0	0	3	0–5	2

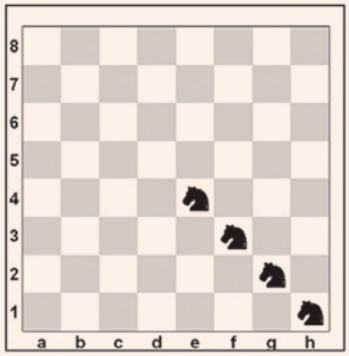
Каждая команда провела с каждой по одному матчу. С каким счетом сыграли команды между собой, если известно, что только два матча закончились одинаково?

Расстановка четырех шахматных фигур

В.А. Полоудин,
кандидат педагогических наук,
кандидат в мастера спорта по шахматам,
Москва

Необходимо разделить шахматную доску на четыре части так, чтобы в каждой части находилось по одной фигуре, например коню, и чтобы все части состояли из одного и того же количества клеток.

Эта задача широко известна среди шахматистов. Но людям, не играющим в шахматы регулярно, решить эту задачу достаточно сложно. Поэтому сделаем маленькую подсказку: коней надо расположить подряд на четырех полях одной из больших диагоналей. Например, так, как это указано на рисунке.



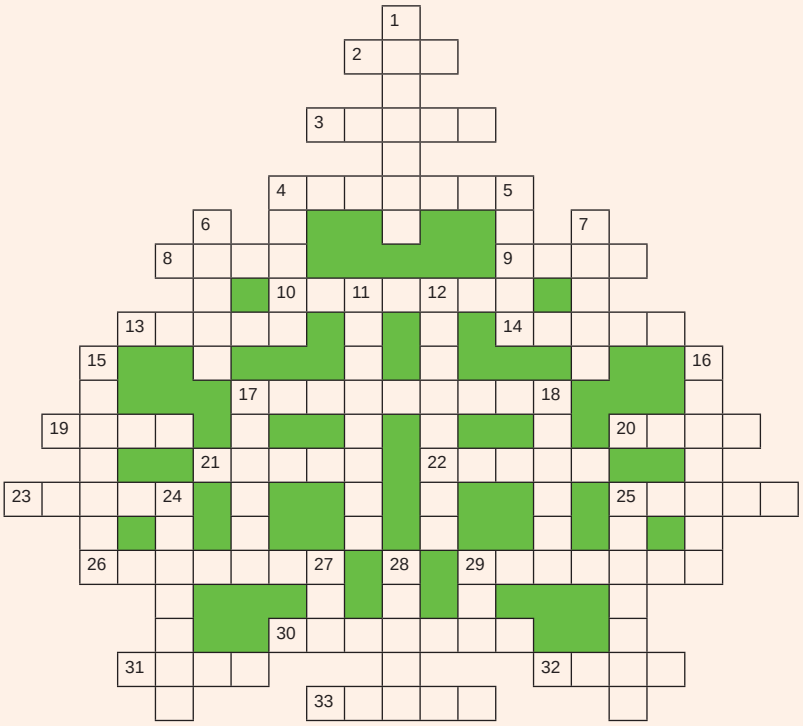
Решение оформите в виде графического файла или листа электронной таблицы Microsoft Excel или другой, или в виде рисунка в документе текстового редактора Microsoft Word так, чтобы части шахматной доски, отвечающие условиям задачи, были выделены разным цветом.

Крест-накрест

Переставив буквы в строках приведенного ниже квадрата, получите слова, при этом в диагоналях квадрата соберутся еще два слова, связанные с информатикой и компьютерами. Найдите все слова и дайте комментарии к ним.

С	Т	А	Р	Р
О	Т	К	О	П
Б	Е	Р	О	Р
В	О	Д	Ы	В
Т	О	Л	Ь	В

Новогодний кроссворд



По горизонтали:

2. Задний план, на котором изображается символ на экране.

3. Последовательность букв и цифр, ограниченная с обоих концов пробелами, запятыми, точками, дефисами и т.п.

4. Так называли гибкий магнитный диск.

8. Наименьшая химически неделимая часть химического элемента, являющаяся носителем его свойств.

9. Это слово происходит от итальянского слова, означающего “ноль”.

10. Цифра двоичной системы счисления.

13. Место вывода изображения на мониторе.

14. Знак, обозначающий число.

17. Метод обращения к устройству или элементу данных по его адресу, метод идентификации местоположения объекта.

19. Популярный вид компьютерных программ.

20. Вычислительное устройство у древних греков и римлян, похожее на счеты.

21. … обмена (место временного хранения копируемого фрагмента текста).

22. Взломщик компьютерных программ.

23. Непрерывная последовательность данных.

25. Операция переработки информации в компьютере.

26.

29. Устройство, осуществляющее преобразование представления и скорости передачи информации между ЭВМ и внешним устройством.

30. Знак препинания.

31. Разновидность носителя информации.

32. Алгоритмическая конструкция, обеспечивающая повторение операций.

33. Первая буква греческого алфавита.

По вертикали:

1. Участок магнитного диска в виде двух концентрических окружностей, образуемый при разметке диска.

4. Часть адреса в электронной почте или адреса web-сайта.

5. В текстовом редакторе Microsoft Word — текст, набранный до нажатия клавиши **“Enter”**.

6. Запуск устройства, системы.

7. Комплект символов, воспроизводящий знаки алфавита на экране или на принтере.

11. Одно из важнейших понятий математики, обозначаемое знаком “$\int$”.

12. Расположение объектов некоторой классификации в порядке от высшего к низшему.

15. Оригинальное начертание полного или сокращенного наименования организации или товара.

16. Программа для обслуживания периферийного устройства.

17. Предмет, которому приписываются магические силы и который, по мнению многих программистов (и не только), должен принести счастье и удачу.

18. Элемент электронной таблицы.

24. Единица измерения количества информации, равная 1024 битам.

25. Один из двух режимов ввода символов в текстовых редакторах.

27. Язык программирования, названный в честь первой женщины-программиста.

28. Возможность передачи информации на расстоянии, а также общение, взаимодействие при этом.

29. Род попугаев.

ЗАДАЧНИК

Ответы, решения, разъяснения

к заданиям, опубликованным в разделе “В мир информатики” ранее

Числовой ребус с “ПТИЧКАМИ”

Напомним, что требовалось решить числовой ребус:

$$\begin{array}{cccccc} \mathbf{П} & \mathbf{Т} & \mathbf{И} & \mathbf{Ч} & \mathbf{К} & \mathbf{А} \\ + \ \mathbf{П} & \mathbf{Т} & \mathbf{И} & \mathbf{Ч} & \mathbf{К} & \mathbf{А} \\ \hline \mathbf{П} & \mathbf{Т} & \mathbf{И} & \mathbf{Ч} & \mathbf{К} & \mathbf{А} \\ \hline \mathbf{С} & \mathbf{Т} & \mathbf{А} & \mathbf{Й} & \mathbf{К} & \mathbf{А} \end{array}$$

Решение

Из анализа разрядов единиц и десятков следует, что **А** = 0 или **А** = 5 и **К** = 0 или **К** = 5. Если **А** = 5, то тогда **К** не может быть равно 0. Значит, возможен только вариант: **А** = 0, **К** = 5.

Запишем найденные значения в ребус:

	П	**Т**	**И**	**Ч**	5	0
+	**П**	**Т**	**И**	**Ч**	5	0
	П	**Т**	**И**	**Ч**	5	0
	С	**Т**	0	**Й**	5	0

Далее, сумма **Т** + **Т** + **Т** может оканчиваться на **Т** с учетом того, что из разряда справа может “в уме” перейти не больше, чем 2, только в двух случаях:

1) когда **Т** = 4 (“в уме” переходит 2);

2) когда **Т** = 9 (“в уме” переходит 2).

Аналогично, сумма **И** + **И** + **И** может оканчиваться на 0 только в двух случаях:

1) когда **И** = 3 (“в уме” переходит 1);

2) когда **И** = 6 (“в уме” переходит 2).

Но поскольку для получения любого из возможных значений **Т** при сложении в разряде тысяч “в уме” в соседний слева разряд должно перейти 2, то вариант **И** = 3 не подходит. Значит, **И** = 6:

	П	**Т**	6	**Ч**	5	0
+	**П**	**Т**	6	**Ч**	5	0
	П	**Т**	6	**Ч**	5	0
	С	**Т**	0	**Й**	5	0

Проанализируем возможные варианты значения числа **Т**.

Вариант 1. **Т** = 4

Имеем:

	П	4	6	**Ч**	5	0
+	**П**	4	6	**Ч**	5	0
	П	4	6	**Ч**	5	0
	С	4	0	**Й**	5	0

В этом случае **П** = 2 (при **П** = 1 получается **С** = 7, что невозможно), т.е. **С** = 7.

Кроме того, видно, что **Ч** = 7 или **Ч** = 8 или **Ч** = 9. Первое значение недопустимо, так как **С** = 7. При **Ч** = 8 имеем **Й** = 5, что также недопустимо. Третий вариант (**Ч** = 9) возможен — при нем **Й** = 8.

Вариант 2. **Т** = 9:

	П	9	6	**Ч**	5	0
+	**П**	9	6	**Ч**	5	0
	П	9	6	**Ч**	5	0
	С	9	0	**Й**	5	0

В этом случае возможные значения **Ч:** 7 или 8. Второй вариант не подходит (при нем **Й** = 5). Остается **Ч** = 7 и **Й** = 2. Но тогда нельзя подобрать значение **П**: при **П** = 1 получается **С** = 5, что невозможно. Значение **П** = 2 также не подходит (**Й** = 2).

Следовательно, возможно только одно решение ребуса:

	2	4	6	9	5	0
+	2	4	6	9	5	0
	2	4	6	9	5	0
	7	4	0	8	5	0

Правильные ответы представили:

— Абрамов Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Алешин Григорий и Копова Алена, г. Москва, филиал школы № 227, учитель **Уколова О.А.**;

— Ахметшин Адэль, Душутин Денис, Колесников Антон, Костылев Игорь, Михайлов Валерий, Разживина Ирина, Сетто Александра и Храбрых Ангелина, Удмуртская Республика, г. Можга, школа № 1, учитель **Колесникова С.В.**;

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Беляева Мария, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Гайсина Галия, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Голик Екатерина, Миноцкий Ян, Пономарева Татьяна, Тананаева Анастасия и Тананаева Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Дегтярь Анатолий и Новиченко Владимир, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Дукач Светлана, Кирсанова Алеся и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Егоров Дмитрий, Игнатова Анна, Исмаилова Альбина, Каличкин Иван, Лебедиков Денис, Прохорова Диана, Строкина Кристина, Храмова Виктория и Щербакова Евгения, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Кисленко Анастасия и Наделяев Денис, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Коркина Анна и Одинцова Екатерина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Кулябин Максим, Сергеева Полина и Скворцов Сергей, Удмуртская Республика, г. Можга, Центр дополнительного образования детей, руководитель кружка по информатике **Колесникова С.В.**;

— Пронина Вероника, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Решетников Виталий, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**;

— Стороженко Степан, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Чмелюк Николай и Шнайдер Наталья, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

— Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Отметим ответ Вероники Прониной, подробно обосновавшей решение ребуса.

Задача “Уравнение”

Напомним, что требовалось найти все решения уравнения *X*6 × 2 = *Y*5, записанного в некоторой системе счисления, где *X*6 — двузначное число с неизвестной первой цифрой и второй цифрой 6, *Y*5 — двузначное число с неизвестной первой цифрой и второй цифрой 5. Обе неизвестные цифры (и *X*, и *Y*) не могут быть равны нулю.

Решение

Запишем уравнение “в столбик”, заменив умножение сложением:

	X	6	
+	**X**	6	
	Y	5	

Сумма чисел 6 и 6 может оканчиваться на 5 только в семеричной системе счисления ($6_7 + 6_7 = 15_7$).

Далее, так как получающаяся сумма — двузначная, а из крайнего справа разряда “в уме” переходит 1, то для семеричной системы счисления возможные варианты решения:

1) **X** = 1, **Y** = 3; 2) **X** = 2, **Y** = 5.

Ответы представили:

— Александрова Алена, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Березин Василий, Демьянова Елена и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Васильев Андрей, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Герасимова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Голик Екатерина, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Иванов Николай и Левченко Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Калинина Марина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Леоненко Степан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Лёвина Татьяна и Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Новиков Сергей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Пронина Вероника, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

Числовой ребус «“БЛОКНОТ” из трех “ЛИСТКОВ”»

Напомним, что требовалось решить числовой ребус:

	Л	**И**	**С**	**Т**	**О**	**К**
+	**Л**	**И**	**С**	**Т**	**О**	**К**
	Л	**И**	**С**	**Т**	**О**	**К**
	Л	**И**	**С**	**Т**	**О**	**К**
Б	**Л**	**О**	**К**	**Н**	**О**	**Т**

Решение

Исследуем разряд десятков и определим возможные значения цифры **О**, при которых 4**О** + число, перешедшее “в уме” из разряда единиц, оканчивается на **О**:

О	4**О**	**“В уме” из разряда единиц должно перейти**	**Допустим ли вариант?**	**Возможные значения К**
0	0	0	Да	1, 2
1	4	7	Нет	
2	8	4	Нет	
3	12	1	Да	4
4	16	8	Нет	
5	20	5	Нет	
6	24	2	Да	5, 7
7	28	9	Нет	
8	32	6	Нет	
9	36	3	Нет	

Проверка пяти допустимых вариантов показывает, что возможно единственное решение ребуса в варианте **О** = 0, **К** = 1:

	9	7	5	4	0	1
+	9	7	5	4	0	1
	9	7	5	4	0	1
	9	7	5	4	0	1
3	9	0	1	8	0	4

Правильные ответы представили:

— Абрамов Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Герасименкова Александра, Исмаилова Альбина и Лопатинова Елена, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Заседателев Никита, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Иванов Николай, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Молева Александра и Фортыгина Юлия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Стороженко Степан, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Торопов Александр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Головоломка “Переставить четыре спички”

Напомним, что необходимо было, переставив четыре спички, получить из фигуры, показанной на *рис.* 1, фигуру, состоящую из трех квадратов.

Решение представлено на *рис.* 2.

Рис. 1 *Рис.* 2

Правильные ответы прислали:

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Вострикова Яна, Гордюшенкова Анна, Громова Елизавета, Иванова Карина, Королёва Виктория, Малютин Максим, Мицик Дарья, Паршин Вячеслав и Перминова Анастасия, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Гололобов Дмитрий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Джурко Марина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Коробов Сергей, Марков Алексей и Яснов Федор, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Николаева Наталья, Республика Чувашия, село Комсомольское, школа № 1, учитель **Родионов П.В.**;

— Новикова Анна и Потапова Алевтина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Орехов Сергей, Москва, школа № 276, учитель **Соколова Л.И.**;

— Федоров Дмитрий, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**;

— Цыганков Евгений, в прошлом учебном году — ученик 1-го класса Вадьковской средней школы, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.** (Евгений выполнил задание в “натуре” — перекладывая спички);

— Шестакова Яна, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**

Головоломка “61 монета, две фальшивые”

Напомним условие: “Есть 61 монета одинакового внешнего вида. Известно, что две из них — фальшивые. Все настоящие одинакового веса, обе фальшивые — тоже одинакового веса, отличающегося от веса

настоящих монет. Как можно узнать с помощью трех взвешиваний на чашечных весах без гирь, легче или тяжелее настоящих фальшивые монеты? (Определять фальшивые монеты не требуется.)”

Решение

Положим на каждую чашку весов по 20 монет.

1. Если весы в равновесии, то это значит, что среди сравниваемых сорока монет либо нет ни одной фальшивой, либо обе фальшивые, причем по одной монете в каждой чашке.

Разделим одну из этих групп монет на две равные части (по 10 монет) и сравним их веса.

1.1. Если одна из чашек весов перевесит, то среди этих 20 монет есть ровно одна фальшивая, а среди 21 монеты, не участвовавших во взвешиваниях, фальшивых монет нет. Соотношение весов 20 монет, среди которых есть фальшивая, и 20 монет из 21 оставшейся будет соответствовать соотношению весов фальшивой и настоящей монет.

1.2. Если весы будут в равновесии, то среди этих 20 монет (и всех 40 монет, участвовавших в первом взвешивании) нет ни одной фальшивой, а все фальшивые монеты находятся среди оставшихся 21. Выберем из 40 монет 21 монету и положим на одну чашку весов, а на вторую положим оставшиеся 21 монету. Соотношение весов будет соответствовать соотношению весов настоящей и фальшивой монет.

2. Если при первом взвешивании весы перевесят одну сторону, то среди 40 использованных одна или две фальшивые, причем обе фальшивые находятся на одной чашке. Разделим более тяжелую группу монет на две равные части (по 10 монет) и положим на чашки весов (это будет второе взвешивание).

2.1. Если весы в равновесии, то тогда либо фальшивых монет в этой группе из 20 монет нет, либо их две, причем по одной в каждой чашке. Разделим кучку монет с одной из чашек на две равные части (по 5 монет) и сравним их веса. Возможные варианты:

1) весы в равновесии. Тогда фальшивых монет в нашей кучке нет, а в более легкой кучке (при первом взвешивании) — одна или две фальшивые, значит, фальшивая монета легче настоящей;

2) весы перевесили в одну сторону. Тогда среди этих 20 монет находятся обе фальшивые монеты, а в более легкой кучке их вообще нет, значит, фальшивая монета — более тяжелая.

2.2. Если при втором взвешивании весы перевесят в одну сторону, то это значит, что среди этих двадцати монет хотя бы одна фальшивая, значит, фальшивая монета — более тяжелая (ведь мы отобрали более тяжелую группу монет).

Ответы представили:

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Васильев Андрей, Герасименкова Александра, Егоров Дмитрий, Игнатова Анна, Исмаилова Альбина, Лопатинова Елена, Пазухин Данил, Прохорова Диана, Роду Мария, Храмова Виктория, Чириков Александр и Щербакова Евгения, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Галкин Димитрий, Республика Чувашия, село Комсомольское, школа № 1, учитель **Родионов П.В.**;

— Джурко Марина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Иванова Алена, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Молева Александра, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Новиков Сергей и Хромченкова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Орехов Сергей, Москва, школа № 276, учитель **Соколова Л.И.**;

— Стороженко Степан, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Федоров Дмитрий, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Кроссворд

Ответы

По горизонтали: 1. Ось. 2. Имя. 8. Основание. 9. Цифра. 10. Ответ. 11. Копирование. 14. Архив. 16. Кисть. 18. Спам. 19. Файл.

По вертикали: 2 . Сканер. 4. Магнит. 5. Монитор. 6. Два. 7. Деление. 12. Истина. 13. Африка. 15. Рост. 17. Тело.

Ответы прислали:

— Абдрашитов Артем, Марченко Никита, Молева Александра и Фортыгина Юлия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Васильев Андрей, Герасименкова Александра, Егоров Дмитрий, Игнатова Анна, Исмаилова Альбина, Лопатинова Елена, Пазухин Данил, Прохорова Диана, Роду Мария, Храмова Виктория, Чириков Александр и Щербакова Евгения, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Добрецов Иван и Максимов Александр, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Ильин Иван, Республика Чувашия, село Комсомольское, школа № 1, учитель **Родионов П.В.**;

— Мальцев Степан, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Орехов Сергей, Москва, школа № 276, учитель **Соколова Л.И.**;

— Трептау Татьяна и Решетников Виталий, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**;

— Удалова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Федосеева Анастасия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Задание “Три города и страна” (рубрика “Поиск информации”)

Ответы

1. На съемках нашего сериала о Шерлоке Холмсе город Лондон заменил город Рига, в котором была снята основная часть фильма (отдельные съемки проводились также в других городах).

2. Провал второго крестового похода связан с городом Дамаском.

3. Сразу трое “детей лейтенанта Шмидта” — Остап Бендер, Шура Балаганов и Паниковский, герои романа Ильи Ильфа и Евгения Петрова “Золотой теленок”, — нагрянули в исполком города Арбатов (название — вымышленное).

4. Единственная европейская страна, в которой нет своих источников пресной воды, — Мальта.

Памятник, фотография которого была опубликована в задании, находится в украинском городе Бердянске. На нем изображены герои указанного выше романа И.Ильфа и Е.Петрова.

Ответы прислали:

— Аветисян Мариам и Сорокина Анна, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Аксененко Сергей, Перова Валентина, Иванов Иван и Яковлева Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Болдарева Анастасия, Ветлугин Никита, Грабовская Александра, Емелина Анастасия, Молева Александра и Фортыгина Юлия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Бостанова Розалия и Катюрина Александра, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Горшкова Оксана, МБОУ "Кадетская школа", Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Добрякова Светлана, Козина Мария и Чернова Наталья, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Дубовик Мария, Исаков Максим, Калиничева Анастасия, Майорихина Кристина, Мещанинова Ксения, Нестерова Полина, Никифоров Алексей, Ошуев Анатолий, Праслова Анжела, Рабизо Анастасия, Смирнов Илья, Старцев Александр, Шеплякова Екатерина и Шибков Максим, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Колеватых Дмитрий, Свердловская обл., г. Кушва, школа № 6 (фамилия учителя в письме, к сожалению, не указана);

— Леонгардт Анастасия и Однокозова Анна, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Образцова Наталия, Сертаков Игорь и Степаненков Игорь, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Орехов Сергей, Москва, школа № 276, учитель **Соколова Л.И.**;

— Селин Владислав, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Федоров Дмитрий, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Отметим ответ Дмитрия Федорова, подготовившего развернутые комментарии к ответам, снабженные красочными иллюстрациями. Редакция решила наградить Дмитрия дипломом. Поздравляем!

Мамергов Никита (Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.А.**) подготовил видеоролик, иллюстрирующий решение задачи "Пять друзей" (см. *youtube.com/watch?v=CN_0HgL0AKk&feature=youtu.be*).

Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**, разработал программы решения задач, предложенных для самостоятельной работы в статье "Одинаковые ли цифры?" (рубрика "Школа программирования").

Ответы на задания конкурса № 89 представили:

— Волков Владимир и Глушаков Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Молева Александра, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**

Кузнецов Денис и Сосновцев Илья, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**, разработали листы электронной таблицы Microsoft Excel, на которых моделируется игра "Морской бой" (используется язык программирования VBA), причем Денис Кузнецов подготовил вариант игры с квадратным игровым полем. С использованием разработанных моделей игр ребята провели чемпионат класса по игре в "Морской бой". Редакция решила наградить Дениса и Илью дипломами. Поздравляем!

Дорогие ребята! Будьте, пожалуйста, внимательны при оформлении ответов на задания конкурсов и приводите все необходимые сведения. Просим вас не представлять решения в виде графических файлов, на которых изображен текст.

ВНИМАНИЕ! КОНКУРС

Конкурс № 97 "Информатика на шахматной доске". Тур 2

Напомним, что в рамках данного конкурса (он проводится среди учащихся не старше 7-го класса) предлагается выполнить ряд заданий на разработку алгоритмов решения задач, связанных с шахматами.

Задача "Король выходит из лабиринта"

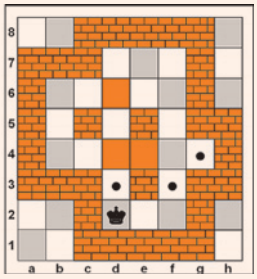
На шахматной доске расположен лабиринт, в котором находится фигура — черный король (см. рисунок). Ему нужно выйти из лабиринта. Король сможет это сделать, если сумеет отремонтировать лабиринт, переставив квадратные цельные блоки, закрашенные однотонным кирпичным цветом, на поля, помеченные точками. В этом случае ему откроется потайная дверь, которую на рисунке не видно.

Составьте алгоритм перестановки блоков. При разработке алгоритма должны выполняться следующие ограничения:

1) короля можно передвигать только на соседнее поле, если оно свободно (то есть шахматные правила перемещения короля соблюдаются);

2) король не может перепрыгивать через блоки;

3) король не может тянуть блоки "на себя", а только толкать блоки "от себя" и только по горизонталям или вертикалям шахматной доски;



4) на каждом поле одновременно может находиться только король или блок.

Алгоритм оформите в виде последовательности ходов короля с указанием номера хода. При этом в каждом ходе сначала запишите обозначение поля доски, на котором находился король, после чего через тире — поля, на которое он перемещается. Напомним, что поля шахматной доски обозначаются двумя символами — строчной латинской буквой (соответствует вертикальным рядам на доске) и числом (соответствует горизонтальным рядам). Например, левое нижнее поле — *a*1, верхнее правое — *h*8.

Итоги конкурса будут подводиться с учетом всех туров в целом.

Задачи конкурса подготовил кандидат педагогических наук, кандидат в мастера спорта по шахматам В.А. Полоудин, Москва, на основе своей книги "Шахматное образование младших школьников. Методические рекомендации" (М.: Научно-методический центр шахматного образования "Два короля", 2012).

ИНФОРМАТИКА

Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)