БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра информационных систем управления

С.И. Кашкевич, А.А. Толстиков

ЗАДАЧИ ШКОЛЬНЫХ ОЛИМПИАД ПО ИНФОРМАТИКЕ

Практикум для студентов факультета прикладной математики и информатики

В трёх частях

Часть 3

Задачи командных чемпионатов г. Минска по программированию (2011 – 2013 годы)

МИНСК
2014

из электронной библиотеки БГУ

УДК 004(075.3)
ББК 32.81к2
К31

Утверждено на заседании кафедры информационных систем управления факультета прикладной математики и информатики
19 сентября 2014 г., протокол № 2

Р е ц е н з е н т ы :
кафедра прикладной математики и технологий программирования Брестского государственного университета им. А.С. Пушкина;
доктор физико-математических наук, профессор *В. М. Котов*

Кашкевич, С.И.
К31 Задачи школьных олимпиад по информатике: практикум для студентов фак. прикладной математики и информатики. В 3 ч. Ч. 3: Задачи командных чемпионатов г. Минска по программированию (2011 – 2013 годы) / С. И. Кашкевич, А.А. Толстиков. – Минск : БГУ, 2014. – 60 с.

Практикум содержит условия, а также разборы решений задач командных чемпионатов г. Минска по программированию (2011 – 2013 годы).

Предназначен для студентов факультета прикладной математики и информатики, а также для школьников, готовящихся к поступлению в вузы.

УДК 004(075.3)
ББК 32.81к2

© Кашкевич С.И., Толстиков А.А., 2014
© БГУ, 2014

из электронной библиотеки БГУ

Предисловие

Третья часть сборника школьных олимпиадных задач по информатике содержит задачи командных чемпионатов по программированию для школьников г. Минска, проводившихся в 2011 – 2013 годах.

В практикуме представлены задачи для командных чемпионатов школьников по программированию, которые проводились в г. Минске в 2011 – 2013 годах. В него включено 47 задач; для каждой из них написан разбор решения. Тесты к задачам доступны по ссылке https://drive.google.com/file/d/0Bymq5SA5EB3NSTZxT0pFMHl6RUE/view?usp=sharing

Особенности проведения соревнований в формате ACM (в котором проводятся командные чемпионаты г. Минска по программированию) подробно описаны во второй части документа.

Первые две части документа можно скачать из электронной библиотеки БГУ по адресам соответственно elib.bsu.by/handle/123456789/41655 и elib.bsu.by/handle/123456789/48379.

Практикум рекомендуется студентам факультета прикладной математики и информатики, школьникам, участвующим в олимпиадном движении по информатике или планирующим принимать такое участие, а также всем, кто интересуется олимпиадным движением.

Авторы благодарят Э.О. Гамисония и А.Е. Климанского, которые внимательно прочли рукопись и сделали ряд замечаний и предложений (особенно касающихся разборов задач), улучшивших её качество.

Выражаем также благодарность доктору физико-математических наук, профессору В.М. Котову за рецензирование рукописи.

Замечания и предложения по усовершенствованию этой работы просим присылать на электронный адрес kash@bsu.by.

из электронной библиотеки БГУ

Условия задач

Четвёртый чемпионат (2011 год)

Чемпионат состоялся 24 сентября 2011 года. Набор задач чемпионата частично совпадает с набором задач студенческой олимпиады БГУ, проходившей в это же время.
На Всероссийскую командную олимпиаду школьников по программированию, проходившую в Санкт-Петербурге, по итогам чемпионата было отобрано шесть команд.
Авторы задач – С.И. Кашкевич (задачи A, C, E-H), Д.А. Ярец (задача D), А.А. Толстиков (задачи B, I). Кроме них, в работе над задачами принимал участие П. А. Иржавский.

Задача A: Сложить ящики

Имя входного файла: boxes.in
Имя выходного файла: boxes.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

В Вашем распоряжении имеется N ящиков. Горизонтальное сечение каждого ящика представляет собой прямоугольник с целочисленными размерами и стенками толщиной 1.

Один ящик можно вставить в другой при выполнении следующих условий:

- ящик A можно вложить в ящик B, если по крайней мере две противоположные внешние стороны ящика A касаются внутренних сторон ящика B;
- дна каждого ящика может касаться не более одного другого ящика.

Так, в ящик с размерами (по внешней стороне) 15×17 можно вставить ящики размерами 13×8 или 5×13, но нельзя вставить ящик размерами 10×10. Кроме того, в этот же ящик нельзя вставить одновременно ящики с размерами 13×5 и 13×6.

Назовём *матрёшкой* конструкцию из нескольких ящиков, последовательно вставленных друг в друга при соблюдении указанных правил. Определите максимальное количество ящиков, которые можно использовать для сборки одной матрёшки.

Из электронной библиотеки БГУ

Формат входных данных. Первая строка файла содержит величину N – количество ящиков ($1 \le N \le 5000$). Каждая из последующих N строк описывает один ящик и содержит его размеры (по внешней стороне). Все размеры – целые положительные числа, не превосходящие 10^6.

Формат выходных данных. Выведите искомую величину – количество ящиков, которые можно вставить друг в друга.

Пример входных и выходных данных

5 20 67 66 45 2 17 64 40 19 38	4

Задача B: Кактус

Имя входного файла: cactus.in
Имя выходного файла: cactus.out
Ограничения по времени: 4 секунды
Ограничения по памяти: 256 мебибайт

Кактус – это связный неориентированный граф без петель, в котором каждая вершина принадлежит не более чем одному простому циклу. Ваша задача – по заданным парам вершин найти кратчайшие расстояния между ними.

Формат входных данных. В первой строке входного файла записаны три целых числа N, M и Q ($2 \le N \le 100\,000$, $1 \le M \le 1\,000\,000$, $1 \le Q \le 100\,000$) – количество вершин и рёбер в графе, количество пар вершин, для которых нужно найти кратчайшее расстояние. Далее следует M строк с описанием рёбер: два целых числа x и y ($1 \le x$, y $\le$ N). Между любой парой вершин нет дублирующихся рёбер. Далее следует Q строк с описанием запросов: два целых числа q_x и q_y ($1 \le q_x$, $q_y \le$ N) – номера вершин графа.

Формат выходных данных. Выведите Q строк – в i-й строке кратчайшее расстояние между вершинами i-го запроса.

из электронной библиотеки БГУ

Пример входных и выходных данных

14 16 6 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 3 2 14 2 9 9 14 9 10 10 11 11 12 12 13 13 10 1 8 6 14 11 7 9 13 11 11 12 10	3 5 6 2 0 2

из электронной библиотеки БГУ

Задача C: Турнир по «минёру»

Имя входного файла: contest.in
Имя выходного файла: contest.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Антон и Борис очень любят играть в известную игру «минёр». Они никак не могут выяснить, кто из них играет лучше – ведь результат игры зависит не только от мастерства игрока, но и от начального расположения мин. Для выяснения этого вопроса друзья решили устроить турнир.

В течение определённого времени Антон и Борис непрерывно играют в «минёра», успевая сыграть достаточно большое количество партий. Результатом одной игры ребята договорились считать количество обезвреженных мин (от 0 до 99). В зачёт каждого игрока идут пять луч-

ших партий из сыгранных им. Сумма этих результатов, делённая на общее количество сыгранных игроком партий, и будет оставлять рейтинг этого игрока.

Увы, очень скоро Антон и Борис запутались в расчётах… Помогите им и рассчитайте рейтинг каждого игрока. Более того, рассмотрите случай, когда в турнире участвует произвольное число игроков!

Формат входных данных. Первая строка входного файла содержит величины N ($2 \le N \le 100$) – количество игроков, и M ($10 \le M \le 10000$) – общее количество партий, сыгранных на турнире всеми игроками. Каждая из последующих M строк описывает одну партию и содержит номер игрока (от 1 до N) и результат партии. Гарантируется, что каждый игрок сыграл не менее пяти партий.

Формат выходных данных. Выведите M строк, каждая из которых содержит рейтинг одного игрока, рассчитанный с точностью до 10^{-4}. Строки упорядочиваются по номеру игрока.

Пример входных и выходных данных

2 15 1 0 1 78 2 3 2 5 2 98 2 81 1 40 1 0 2 0 1 70 2 6 2 71 2 0 2 0 2 15	37.6000 27.1000

из электронной библиотеки БГУ

Задача D: «Flood it!»

Имя входного файла: floodit.in
Имя выходного файла: floodit.out
Ограничения по времени: 3 секунды
Ограничения по памяти: 256 мебибайт

«Flood it!» – это простая, но крайне увлекательная игра. Правила её не очень затейливы: имеется прямоугольное поле, которое разбито на N строк и M столбцов. Перед началом игры каждая из $N \times M$ клеточек покрашена в один из K цветов.

Прежде чем продолжать изложение условий игры, дадим несколько определений.

Две клетки назовём *соседними*, если они имеют общую сторону.

Несколько клеток назовём *областью*, если выполняются следующие условия:

- между любыми двумя клетками области существует путь, проходящий по соседним клеткам этой области;
- все клетки области окрашены в один цвет – цвет области;
- ни одна клетка, не принадлежащая области, но соседствующая с ней, не окрашена в цвет области.

Цель игры состоит в захвате (овладевании) возможно большего числа клеток игрового поля. Изначально во владение игрока отдаётся область, содержащая верхнюю левую клетку. На каждом ходу игрок выбирает какой-либо из K заданных цветов и красит свои владения в этот цвет. Если перед ходом с одной из захваченных ранее клеток соседствует область, имеющая выбранный цвет, то игрок захватывает эту область.

К счастью, Вам не придётся составлять программу для моделирования всей игры. Вместо этого решите более простую задачу: на основании исходного состояния игрового поля и информации о L первых ходах игрока определите игровое поле, получившееся после этих ходов.

Формат входных данных. В первой строке через пробел заданы 4 целых положительных числа: N и M ($1 \le N, M \le 100$) – размеры поля, K ($1 \le K \le 10^6$) – количество различных цветов, и L ($1 \le L \le 1000$) – длина последовательности ходов. Далее следует N строк, в каждой из которых через пробел записано по M чисел в диапазоне от 1 до K, эти числа обозначают цвета начальной раскраски. В последней строке через пробел даны L чисел в диапазоне от 1 до K – последовательность ходов.

Формат выходных данных. В выходной файл необходимо вывести N строк, в каждой из которых через пробел записано M чисел в диапа-

ЭБ электронной библиотеки БГУ

зоне от 1 до K – состояние игрового поля после применения L заданных ходов.

Примеры входных и выходных данных

3 4 5 3 1 2 3 5 2 3 1 1 2 1 3 2 2 3 2	2 2 2 5 2 2 1 1 2 1 3 2
2 5 6 2 1 3 3 3 6 2 4 6 3 4 3 1	1 1 1 1 6 2 4 6 1 4

Задача E: «Дурак»

Имя входного файла: fool.in
Имя выходного файла: fool.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Запись в медицинской карте:
«Признаков психических заболеваний не выявлено. Просто дурак»

Игра в «дурака» ведётся колодой из 36 карт. Каждая карта характеризуется своим *достоинством* и *мастью*. Существуют 9 достоинств: шестёрка (6), семёрка (7), восьмёрка (8), девятка (9), десятка (10), валет (J), дама (Q), король (K), туз (A) – и четыре масти: пики (S), кресты (C), бубны (D), черви (H)[1]. Обозначение карты состоит из обозначений её достоинства и масти, записанных без пробела, например: AS – туз пик, 10D – десятка бубен. Внутри одной масти карты упорядочены по возрастанию их достоинств в соответствии с приведённым выше списком. Перед началом игры одна из мастей объявляется *козырной*.

Рассмотрим случай игры двух игроков. Игра состоит из серии атак. Начало атаки заключается в том, что атакующий игрок выкладывает на стол одну из карт, имеющихся у него на руках (*заходную* карту). Защищающийся игрок может отбить её либо картой той же масти и большего достоинства, либо козырной картой (при условии, что такие карты есть у

[1] В скобках записаны обозначения для достоинств и мастей, состоящие из прописных латинских букв и цифр.

из электронной библиотеки БГУ

него на руках). Если заходная карта – козырь, то её можно отбить лишь козырной картой большего достоинства.

Прекратим на время изложение правил этой игры и зададим несложную задачу. На основании данных о козыре, заходной карте и картах, имеющихся на руках у защищающегося игрока, определите количество вариантов для отражения заходной карты.

Формат входных данных. В первой строке записано обозначение козыря, во второй – описание заходной карты. Третья строка содержит величину *K* – количество карт на руках у защищающегося игрока. Каждая из последующих *K* строк содержит описание одной из таких карт. Все перечисленные во входных данных карты различные.

Формат выходных данных. Выведите единственное число – количество возможных вариантов отражения заходной карты.

Примеры входных и выходных данных

S 9D 5 8D KH 10D 6S AS	3
S 9D 1 8D	0

Задача F: «Дурак» – продолжение следует

Имя входного файла: fool2.in

Имя выходного файла: fool2.out

Ограничения по времени: 2 секунды

Ограничения по памяти: 256 мебибайт

Игра в «дурака» тем временем продолжается… Если заходная карта отбита, атакующий игрок может продолжить атаку, выложив на стол другую карту из имеющихся у него на руках, при соблюдении следующего условия: достоинство этой карты должно совпадать с достоинством одной из карт, лежащих на столе. Так, если заходная карта 9D отбита ко-

из электронной библиотеки БГУ

зырной картой 6S, то атаку можно продолжить девяткой или шестёркой любой масти. Защищающийся игрок может отбить эту карту по правилам, описанным в предыдущей задаче. Количество карт, выложенных атакующим игроком, не может превосходить шести.

Длиной атаки назовём количество карт, выложенных в её ходе атакующим игроком.

Сможете ли Вы на основании информации о картах, находящихся на руках у игроков, смоделировать самую длинную из отбитых атак?

Формат входных данных. В первой строке записано обозначение козыря, во второй – величины K_1 и K_2 – количество карт на руках у атакующего и защищающегося игрока ($1 \le K_1, K_2 \le 10$). Каждая из последующих K_1 строк содержит описание одной из карт атакующего, а каждая из последующих K_2 строк – защищающегося игрока.

Формат выходных данных. Выведите единственное число – длину наибольшей из отбитых атак.

Пример входных и выходных данных

S 4 5 6H KD 7S QS 7H AS KS 6S 10D	4

Пояснение: Одна из возможных атак выглядит следующим образом: карта QS отбивается картой KS, далее следуют ходы: KD – 6S, 6H – 7H, 7S – AS.

из электронной библиотеки БГУ

Задача G: Преобразование дроби

Имя входного файла: fraction.in
Имя выходного файла: fraction.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Задана правильная непериодическая дробь, большая нуля, в десятичной системе счисления. Требуется перевести её в шестнадцатеричную систему счисления, выделяя периодическую и непериодическую части.

Формат входных данных. Единственная строка входного файла содержит десятичное представление дроби (без ведущего нуля и десятичной точки). Длина строки не превосходит 6 символов.

Формат выходных данных. Выведите две строки, содержащие соответственно непериодическую и периодическую части шестнадцатеричного представления исходной дроби. Если одна из частей отсутствует, соответствующая строка должна быть пустой. Напомним, что шестнадцатеричные цифры задаются символами '0' .. '9', 'A' .. 'F'.

Примеры входных и выходных данных

24	– 3D70A
7	B 3

Пояснение. Для наглядности в примерах пустая строка заменена строкой с символом «минус».

Задача H: Гири и весы – 2

Имя входного файла: scales.in
Имя выходного файла: scales.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Комплект гирь состоит из N гирь различного веса. Сможете ли Вы уравновесить на рычажных весах мешок весом S некоторыми гирями из этого комплекта? Естественно, гири должны быть на одной чашке весов, а мешок – на другой…

Более того, определите количество вариантов такого уравновешивания. Варианты считаются различными, если множества гирь, лежащие на весах, различны.

из электронной библиотеки БГУ

Формат входных данных. Первая строка файла содержит величину N – количество гирь ($1 \le N \le 100$). Вторая строка содержит N чисел – веса каждой гири (целые положительные числа, не превосходящие 1000). Наконец, в третьей строке записано одно целое число – вес мешка S ($1 \le S \le 5000$).

Формат выходных данных. Выведите искомую величину – количество вариантов уравновешивания мешка. Если сделать это невозможно, выведите ноль.

Примеры входных и выходных данных

5 1 2 3 4 5 8	3
2 999 2 1000	0

Примечание: Первая задача из этого цикла была предложена на районной олимпиаде г. Минска по информатике 2010 г.

Задача I: Квадрат

Имя входного файла: square.in
Имя выходного файла: square.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

На плоскости заданы N различных точек. Вам нужно построить квадрат минимальной площади, содержащий внутри и на границе все данные точки.

Формат входных данных. В первой строке записано единственное целое число N ($2 \le N \le 30$). Далее в N строках задаются точки на плоскости (x_i, y_i). Все координаты – целые и по абсолютной величине не превосходят 10^4.

Формат выходных данных. Выведите две строки, по два вещественных числа в каждой – координаты противоположных вершин искомого квадрата. Относительная погрешность площади найденного вами квадрата не должна превышать 10^{-9}.

из электронной библиотеки БГУ

Пример входных и выходных данных

5 1 0 0 1 1 1 2 1 1 2	1.000000000 0.000000000 1.000000000 2.000000000

Пятый чемпионат (2012 год)

Чемпионат состоялся 30 сентября 2012 года. Набор задач чемпионата частично совпадает с набором задач студенческой олимпиады БГУ, проходившей в это же время.
На Всероссийскую командную олимпиаду школьников по программированию, проходившую в Санкт-Петербурге, по итогам чемпионата было отобрано пять команд.
Авторы задач – С.И. Кашкевич (задачи A-D, F,H), А.А. Толстиков (задача E), П. А. Иржавский (задача G). Кроме них, в работе над задачами принимали участие В.В. Керус и Ю.Писарчик.

Задача A: Я буду долго гнать велосипед...

Имя входного файла: bicycle.in
Имя выходного файла: bicycle.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Наконец-то, и город Нью-Васюки приблизился к цивилизации - его власти построили целых две велодорожки, по каждой из которых можно кататься в двух направлениях! Каждая из них имеет форму прямоугольника, стороны которого параллельны координатам, нанесённым на официальную карту города. Велосипедистам разрешается сворачивать с одной дорожки на другую, но категорически запрещено выезжать за их пределы (например, внутрь прямоугольника, который ограничивается дорожкой). Дорожки могут пересекать или касаться друг друга, и даже иметь общие участки ненулевой длины.

Вам необходимо проехать как можно быстрее из пункта *A* в пункт *B*. Известно, что оба этих пункта находятся на одной из велодорожек, а единицу расстояния Вы проезжаете за единицу времени.

Формат входных данных. Первая и вторая строка входного файла содержит координаты северо-западного и юго-восточного углов прямо-

Электронный библиотеки БГУ

угольника, соответствующего одной из велодорожек. В третьей и четвёртой строках записаны соответственно координаты пунктов *A* и *B*. Все координаты – целые числа, не превосходящие по модулю 10^9. Прямоугольники, которые описывают каждую из дорожек, имеют ненулевую высоту и ширину.

Формат выходных данных. Вычислите минимальное время, за которое Вы сможете доехать из пункта *A* в пункт *B*, соблюдая приведенные выше условия. Если такая поездка невозможна, выведите -1.

Примеры входных и выходных данных

0 10 10 0 10 20 20 10 0 0 20 15	35
0 10 10 0 2 8 8 2 0 0 2 2	-1

Задача B: И будет вам щасце!

Имя входного файла: gluck.in
Имя выходного файла: gluck.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Земная колония на далёкой планете Новый Китай процветает (в отличие от перенаселённой метрополии), и ей нужны новые жители. Руководство колонии прислало заявку на *N* супружеских пар, которые согласились бы навсегда поселиться в Новом Китае. Увы, стольких желающих не нашлось… Зато власти Земли смогли отобрать *N* незамужних женщин и столько же холостых мужчин, согласившихся променять земные трущобы на «рай» в колонии. К концу межпланетного путешествия все они должны составить супружеские пары, причём подбор супругов выполняют прикомандированные психологи, а у переселенцев нет права изменить этот выбор.

Во время долгого перелёта психологи, наблюдая за контактами между переселенцами, смогли определить т.н. взаимный уровень симпатии S_{ij} между *i*-м мужчиной и *j*-й женщиной из них. Психологи считают,

из электронной библиотеки БГУ

что чем больше величина взаимной симпатии, тем более прочным и счастливым получится брак.

Помогите психологам, рассчитав такой вариант формирования из переселенцев супружеских пар, при котором суммарная величина взаимной симпатии в парах будет максимальной.

Формат входных данных. Первая строка входного файла содержит величину N ($1 \le N \le 100$). Далее следуют N строк из N чисел каждая – матрица взаимных симпатий S_{ij}. Величины S_{ij} – целые, неотрицательные, не превосходящие 100.

Формат выходных данных. Первая строка выходного файла должна содержать рассчитанную суммарную величину взаимной симпатии. Каждая из последующих N строк соответствует одной супружеской паре и содержит номера мужчины и женщины, которые эту пару составят (нумерация начинается с единицы). Если задача допускает несколько решений, выведите любое из них.

Примеры входных и выходных данных

3 3 2 4 2 1 2 2 2 5	9 1 1 2 2 3 3
3 1 2 3 3 2 1 2 2 2	8 1 3 2 1 3 2

Задача C: Фонари у дороги – 1

Имя входного файла: lantern.in

Имя выходного файла: lantern.out

Ограничения по времени: 1 секунда

Ограничения по памяти: 256 мебибайт

Прямой отрезок дороги из Нью-Васюков в Старую Москву длины M снискал дурную славу из-за большого количества аварий на нём, особенно в ночное время. Дорожный департамент в конце концов решил установить фонари и осветить весь этот отрезок. И тут чиновники обратили внимание, что отдельные участки дороги уже освещаются N фонарями, стоящими у заправочных станций, придорожных кафе и т.д., так

из электронной библиотеки БГУ

что выделенные деньги можно сэкономить… В частности, фонарь с номером i ($1 \le i \le N$) освещает участок дороги $[a_i, b_i]$, где $0 \le a_i < b_i \le M$.

Участок дороги считается безопасным, если он освещен хотя бы одним фонарём.

По информации о расположении освещённых участков дороги определите суммарную длину безопасных участков дороги и количество неосвещённых участков.

Формат входных данных. Первая строка содержит два целых числа: M – длину дороги, и N – количество фонарей ($1 \le M \le 10^6$, $1 \le N \le 10^5$). Каждая из последующих N строк задаёт информацию об одном фонаре и содержит два целых числа a_i и b_i.

Формат выходных данных. Выведите два числа, разделённые пробелом – суммарную длину безопасных участков дороги и количество неосвещённых участков.

Примеры входных и выходных данных

8 2 1 5 4 8	7 1
8 2 0 6 6 8	8 0

Задача D: Фонари у дороги – 2

Имя входного файла: lantern2.in
Имя выходного файла: lantern2.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Прямой отрезок дороги из Нью-Васюков в Старую Москву длины M снискал дурную славу из-за большого количества аварий на нём, особенно в ночное время. Дорожный департамент в конце концов решил установить фонари и осветить весь этот отрезок. И тут чиновники обратили внимание, что отдельные участки дороги уже освещаются N фонарями, стоящими у заправочных станций, придорожных кафе и т.д., так что выделенные деньги можно сэкономить… В частности, фонарь с номером i ($1 \le i \le N$) освещает круг с центром в точке (a_i, b_i) с радиусом R_i. Дорога проходит из точки (x_1, y_1) в точку (x_2, y_2), и её шириной можно пренебречь.

из электронной библиотеки БГУ

Участок дороги считается безопасным, если он освещен хотя бы одним фонарём.

По информации о расположении освещённых участков дороги определите суммарную длину безопасных участков.

Формат входных данных. Первая строка содержит два целых числа: M – длину дороги и N– количество фонарей ($1 \le M \le 10^6$, $1 \le N \le 10^5$). Во второй строке задаются координаты x_1, y_1, x_2, y_2 – целые числа, не превосходящие по модулю 10^6.

Каждая из последующих N строк задаёт информацию об одном фонаре и содержит три целых числа a_i, b_i и R_i. Диапазоны значений для этих чисел: $-10^6 \le a_i, b_i \le 10^6$, $1 \le R_i \le 10^6$.

Формат выходных данных. Выведите единственное число – суммарную длину безопасных участков дороги с относительной или абсолютной погрешностью не более 10^{-6}.

Примеры входных и выходных данных

1 0 0 9 0 3 2 3	4.4721359550
2 -1 -4 5 4 6 -3 5 3 3 3	4.8284271247

Задача E: Палиндромы

Имя входного файла: palindroming.in
Имя выходного файла: palindroming.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайт

According to Guinness World Records, the Finnish word saippuakivikauppias (soapstone vendor), a 19-letter word, is claimed to be the world's longest palindromic word in everyday use.

© Wikipedia

Вам дана строка S из маленьких латинских букв. Вы можете за один ход вычеркнуть какую-то букву в ней или поменять две буквы местами.

из электронной библиотеки БГУ

Сколько разных палиндромов вы можете получить, сделав несколько ходов (возможно, ноль)?

Формат входных данных. Во входном файле содержится одна непустая строка S (длиной до ста символов).

Формат выходных данных. Выведите единственное число – количество различных палиндромов, которые можно получить. Так как это число может оказаться слишком большим, то выведите только остаток от деления этого числа на $10^9 + 2012$.

Примеры входных и выходных данных

Acb	3
abracadabra	157

Примечание. Палиндромом называется непустая строка, которая одинаково читается в обоих направлениях.

Задача F: Прогрессия

Имя входного файла: progression.in
Имя выходного файла: progression.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Задана последовательность из N целых чисел. Выделите из нее подпоследовательность максимальной длины из стоящих подряд чисел, которая образовывала бы арифметическую прогрессию.

Напомним, что последовательность $a_1, a_2, \ldots, a_k$ является арифметической прогрессией, если разность $a_i - a_{i-1}$ постоянна для $i = 2, \ldots, k$.

Формат входных данных. Первая строка содержит величину N ($2 \le N \le 10^5$). Каждая из последующих N строк содержит одно число, не превосходящее по модулю 10^9 – очередной элемент последовательности.

Формат выходных данных. Выведите единственное число – длину найденной прогрессии.

из электронной библиотеки БГУ

Примеры входных и выходных данных

7 1 2 3 4 5 4 3	5
5 3 1 8 -4 0	2

Задача G: Серии

Имя входного файла: runs.in

Имя выходного файла: runs.out

Ограничения по времени: 1 секунда

Ограничения по памяти: 64 мебибайта

Для заданной перестановки π чисел от 1 до *N* *серией* назовём максимальный (по включению) интервал индексов [*i; j*], для которого выполняется:

$\pi(k) < \pi(k+1)$ при всех $i \le k < j$, или

$\pi(k) > \pi(k+1)$ при всех $i \le k < j$.

Например, перестановка

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 2 & 3 & 4 \end{pmatrix}$$

содержит 3 серии: [*1; 2*], [*2; 3*], [*3; 5*], поскольку $\pi(1) < \pi(2)$, $\pi(2) > \pi(3)$ и $\pi(3) < \pi(4) < \pi(5)$.

Пусть при заданном *N* выбирается случайная перестановка из *N* чисел (все возможные перестановки имеют равные вероятности).

Требуется посчитать вероятность того, что выбранная перестановка будет содержать ровно *R* серий.

Формат входных данных. Единственная строка содержит два целых числа *N* и *R* ($1 \le N, R \le 500$) – количество элементов в перестановке и требуемое количество серий.

из электронной библиотеки БГУ

Формат выходных данных. Единственная строка должна содержать искомую вероятность с абсолютной погрешностью не более 10^{-9}.

Примеры входных и выходных данных

3 1	0.333333333333
4 2	0.500000000000

Задача H: По спирали

Имя входного файла: spiral.in
Имя выходного файла: spiral.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Матрица, состоящая из M строк и N столбцов, заполняется натуральными числами по спирали: первый столбец сверху вниз, оставшиеся позиции последней строки – слева направо, оставшиеся элементы последнего столбца – снизу вверх, оставшиеся элементы первой строки – справа налево, и т. д.

Так, матрица из 4 строк и 5 столбцов имеет вид

1	14	13	12	11
2	15	20	19	10
3	16	17	18	9
4	5	6	7	8

Найдите значение элемента матрицы, стоящего в i-й строке и j-м столбце.

Формат входных данных. Первая строка входного файла содержит величины M и N ($1 \le M, N \le 10^9$).

Вторая строка содержит значения i и j ($1 \le i \le M$, $1 \le j \le N$).

Формат выходных данных. Единственная строка выходного файла должна содержать искомое значение элемента матрицы.

Примеры входных и выходных данных

4 5 2 2	15
20 30 1 2	96

из электронной библиотеки БГУ

Шестой чемпионат (2013 год)

Чемпионат состоялся 5 октября 2013 года. Как и в прошлые годы, набор задач чемпионата частично совпал с набором задач студенческой олимпиады БГУ, проходившей в это же время.
На Всероссийскую командную олимпиаду школьников по программированию, проходившую в Санкт-Петербурге, по итогам чемпионата было отобрано шесть команд.
Авторы задач – С.И. Кашкевич (задачи A-F, H), П.А. Иржавский (задача G), С.А. Соболь (задача I), А.А. Толстиков (задача J).

Задача A: «Балда»

Имя входного файла: balda.in
Имя выходного файла: balda.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

«Балда» — это игра со словами. Смысл игры состоит в образовании новых слов, связанных с уже выставленными на поле, путем прибавления одной буквы.

Поле для игры представляет собой матрицу 5×5, в каждую клетку которой может быть вписана ровно одна буква. Изначально средняя строка матрицы заполняется слева направо словом, использовать которое во время игры нельзя.

Игроки делают ходы по очереди. Выполняя ход, игрок пишет в одной из незаполненных клеток букву, а потом сообщает слово, которое он придумал. Это слово должно быть прочитано передвижением из одной заполненной клетки в соседнюю заполненную по горизонтали или вертикали. Каждую клетку можно посетить не более одного раза. Кроме того, в придуманном слове должна быть буква, которую написал игрок.

Побеждает тот игрок, который придумал слова максимальной суммарной длины.

Рассмотрим пример игры. Пусть в неё играют два человека, и изначальное слово — «БАЛДА».

Исходное поле выглядит так:

Б	А	Л	Д	А

из электронной библиотеки БГУ

Первый игрок записал букву «Л» и придумал слово «БАЛЛ». Поле принимает вид

Б	А	Л	Д	А
		Л		

После этого второй игрок записывает букву «А» и находит слово «БАЛЛАДА», поле принимает вид

Б	А	Л	Д	А
		Л	А	

Второй игрок ведёт со счётом 7:4.

Вам требуется определить, соответствует ли правилам игры очередное придуманное слово.

Формат входных данных. Первые пять строк описывают текущее состояние игрового поля; каждая такая строка состоит из пяти символов. В заполненных клетках помещаются прописные буквы латинского алфавита, незаполненные клетки помечаются точкой. Гарантируется, что средняя строка не содержит пустых клеток.

В следующей строке находятся два числа: номер строки и номер столбца с последней записанной буквой (нумерация начинается с единицы). Гарантируется, что в этой позиции будет присутствовать буква и что эта позиция не соответствует средней строке игрового поля.

Наконец, последняя строка входного файла содержит придуманное слово. Оно состоит из не более чем 25 прописных латинских букв.

Формат выходных данных. Единственная строка должна содержать слово «Yes» (без кавычек), если придуманное слово удовлетворяет правилам игры, и «No» в противном случае.

из электронной библиотеки БГУ

Примеры входных и выходных данных

.... BALDA ..LA. ..... 4 4 BALLADA	Yes
.... BALDA ..LA. ..... 4 4 LLAB	No
.... BALDA ..LA. ..... 4 4 LABUDA	No

Задача B: Байкеры – 2

Имя входного файла: bikers.in

Имя выходного файла: bikers.out

Ограничения по времени: 2 секунды

Ограничения по памяти: 1 гибибайт

В стране насчитывается N городов, и в каждом городе есть команда байкеров. Логично провести фестиваль в одном из этих городов. Наибольшим влиянием пользуются команды из городов, пронумерованных как 1 и N, поэтому их мнение о месте проведения фестиваля будет решающим...

Указанные города соединены сетью из M двусторонних платных дорог (между двумя городами может быть проложено несколько таких дорог). Команда из города 1 должна заплатить за проезд по i-й дороге ($1 \le i \le M$) сумму в a_i, а команда из города N — сумму b_i. Лидеры этих команд в телефонном разговоре договорились предложить в качестве ме-

из электронной библиотеки БГУ

ста проведения фестиваля такой город, что суммарная стоимость проезда к нему членов обеих команд будет минимальной.

Сможете ли вы определить такой город?

Формат входных данных. В первой строке входного файла записаны величины N и M ($2 \le N \le 1000, 1 \le M \le 5000$). Далее следуют M строк, описывающих каждую дорогу и содержащих четыре числа — номера городов x и y, которые соединяет эта дорога ($1 \le x, y \le N, x < y$), а также величины a_i и b_i для этой дороги (целые, $0 < a_i, b_i \le 1000$). Гарантируется, что команды из всех городов смогут встретиться, путешествуя только по описанным дорогам.

Формат выходных данных. Выведите два числа - номер города, в котором будет проведен фестиваль, и суммарную величину стоимости поездки на этот фестиваль для команд из городов *1* и *N*. Если задача допускает несколько решений, выведите город с минимальным номером.

Пример входных и выходных данных

4 4 1 2 1 6 2 3 4 1 4 3 1 1 1 3 4 7	3 2

Примечание: Первая задача из этой серии была предложена на минском четвертьфинале чемпионата ACM в 2008 году

Задача C: Что больше – 2?

Имя входного файла: compare.in
Имя выходного файла: compare.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

Заданы два числа N_1 и N_2, причём первое из них записано в системе счисления с основанием a_1, а второе - в системе счисления с основанием a_2. Определите, какое из чисел больше.

Напомним, что для записи чисел в системе счисления с основанием, большим 10, кроме цифр, используются заглавные буквы латинского алфавита: A - вместо 10, B - вместо 11, и т. д.

из электронной библиотеки БГУ

Формат входных данных. Первая строка содержит значения a_1 и a_2 ($2 \le a_1, a_2 \le 36$). Во второй и третьей строке записаны соответственно числа N_1 и N_2. Каждое из этих чисел содержит не более 1000 цифр и не имеет ведущих нулей.

Формат выходных данных. Если $N_1 < N_2$, выведите в единственную строку выходного файла значение −1. Если $N_1 > N_2$, выведите значение 1. Если эти числа равны, выведите 0.

Примеры входных и выходных данных

2 10 1011 10	1
10 16 255 FF	0

Примечание: Первая задача из этой серии была предложена на 2-м командном чемпионате школьников г. Минска в 2009 году.

Задача D: Осторожно, злая собака!

Имя входного файла: dog.in
Имя выходного файла: dog.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

На ровной поверхности земли в несовпадающих точках с координатами (a_1, b_1) и (a_2, b_2) вбиты два кола, между которыми натянут канат. К этому канату привязана цепью собака таким образом, что один конец цепи может скользить по канату. Кроме того, крепление каната позволяет собаке беспрепятственно переходить с одной его стороны на другую. Длина цепи не даёт собаке отойти от каната на расстояние больше чем d.

Определите площадь участка земли, по которому может ходить собака.

Формат входных данных. Единственная строка входного файла содержит величины a_1, b_1, a_2, b_2, d. Все величины - целые, первые четыре числа не превосходят по модулю 10^3, а пятое находится в диапазоне от 1 до 10^3.

из электронной библиотеки БГУ

Формат выходных данных. Выведите единственное число - искомую площадь участка, относительная погрешность не должна превышать 10^{-9}.

Примеры входных и выходных данных

0 0 0 2 1	7.1415926535897931
0 0 10 10 5	219.9611725770543270

Задача E: Живая вода

Имя входного файла: lifewater.in
Имя выходного файла: lifewater.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

Герой компьютерной игры зашёл в лавку со снадобьями, чтобы поправить здоровье, потерянное в схватках с врагами. Текущая величина здоровья героя равна P, а максимальная величина здоровья, соответствующая его опыту и навыкам, равна Q.

Хозяин лавки предлагает герою купить бутылочки с живой водой. В продаже имеется неограниченное количество бутылочек с живой водой трёх видов: малая, средняя и большая. Использование одной бутылочки каждого вида увеличивает здоровье персонажа на a_s, a_m и a_l единиц соответственно. При этом максимальное здоровье героя не должно превосходить Q, а бутылочку с живой водой, после того как её открыли, повторно использовать нельзя.

Малая, средняя и большая бутылочка живой воды стоят в лавке соответственно b_s, b_m и b_l золотых монет, а герой может потратить на их покупку не более K монет. Он решил полностью восстановить своё здоровье, потратив минимально возможную сумму денег. Если же полное восстановление здоровья невозможно, герой должен восстановить здоровье на максимально возможную величину (если существует несколько способов восстановления здоровья, то герой должен выбрать тот, у которого сумма денег минимальна).

Помогите герою сделать этот непростой выбор!

Формат входных данных. Первая строка файла содержит величины P, Q и K ($1 \le P, Q, K \le 1000$, $P < Q$). Во второй строке задаются величины a_s, a_m и a_l ($1 \le a_s < a_m < a_l \le 1000$). Наконец, в третьей строке входного файла записаны величины b_s, b_m и b_l ($1 \le b_s, b_m, b_l \le 1000$).

ИС электронной библиотеки БГУ

Формат выходных данных. Выведите в единственную строку выходного файла два числа: количество денег, потраченное героем на покупку снадобий, и величину здоровья, которую он получит после употребления всех бутылочек.

Примеры входных и выходных данных

2 5 2 1 2 3 1 1 2	2 5
2 5 2 1 2 3 1 2 3	2 4

Задача F: Построить треугольник

Имя входного файла: medians.in
Имя выходного файла: medians.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

Определите площадь невырожденного треугольника *ABC* по длинам трёх его медиан.

Формат входных данных. Единственная строка входного файла содержит три числа - длины медиан, проведённых соответственно из вершин A, B, C. Эти числа не превосходят 1000 и содержат не более четырёх знаков в дробной части.

Формат выходных данных. В единственную строку выходного файла выведите рассчитанную площадь с относительной или абсолютной погрешностью не более 10^{-6}. Если решения не существует, то выведите −1.

Примеры входных и выходных данных

3 6 7	11.925695880
3.2 6.1 10.8	-1

из электронной библиотеки БГУ

Задача G: Спички — не игрушка!

Имя входного файла: nim.in
Имя выходного файла: nim.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

Родители мальчика Вовы часто ему повторяют, что со спичками играть нельзя. Однако любознательный мальчик постоянно придумывает новые игры со спичками, не рискуя зажигать их. Не так давно папа Вовы рассказал ему занимательную игру «Ним», в которой камушки разложены по нескольким кучкам, и два игрока, делая ходы поочерёдно, могут брать произвольное количество камушков из любой непустой кучки. Тогда же папа научил его определять, может ли он выиграть (при оптимальной стратегии противника), если перед его ходом определенное количество камушков в кучках.

Но игры с камушками совсем не привлекают Вову, а спичек у него дома очень-очень много. Вот он и решил посчитать, сколько существует начальных позиций в игре, в которых выигрывает первый игрок. Для начала он легко решил эту задачу для одной кучки спичек. А сейчас его интересует, сколькими способами можно разложить спички в N кучек, в каждой должно быть менее 2^M спичек, и при этом в разных кучках должно быть различное количество спичек.

Для уточнения условия смотрите примеры.

Формат входных данных. В единственной строке записаны два целых числа N и M ($1 \le N, M \le 10\,000\,000$).

Формат выходных данных. В единственной строке выведите искомое количество способов по модулю 1 000 000 007.

Примеры входных и выходных данных

1 10	1023
4 2	0
3 3	168

из электронной библиотеки БГУ

Задача H: Запачканный номер

Имя входного файла: nummernschild.in
Имя выходного файла: nummernschild.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

Автомобильные номера Германии, за исключением специальных номеров, имеют следующий формат:

- код региона, содержащий от одной до трёх прописных латинских букв (например, **M** — Мюнхен, **MA** — Мангейм, **MAB** — Мариенберг);
- пробел, на котором клеится стикер о прохождении техосмотра;
- серия, содержащая одну или две прописных латинских буквы;
- собственно номер – не более чем четырёхзначное число без ведущих нулей.

Общее количество символов в номере, включая пробел, не превосходит девяти.

Комбинации **SS**, **SA**, **HJ**, **NS**, **KZ**, напоминающие о нацистском прошлом Германии, запрещены к использованию в качестве кодов регионов и серий. Все остальные комбинации символов в рамках данной задачи считаются допустимыми. Так, комбинация **SSS** допустима в качестве кода региона.

Камера слежения, установленная над автобаном, зафиксировала нарушение правил движения и сделала снимок автомобиля-нарушителя. Однако номер этого автомобиля был запачкан грязью, так что отдельные его символы не распознались. Помогите дорожной полиции установить нарушителя, рассчитав общее количество номеров, которые могут быть установлены на автомобиль-нарушитель.

Формат входных данных. Единственная непустая строка длиной не более 9 символов - результат сканирования автомобильного номера. Строка содержит прописные латинские буквы, цифры, знак подчёркивания, который ставится вместо пробела, и звёздочки, соответствующие нераспознанным символам. Гарантируется, что в строке есть хотя бы одна звёздочка, и что существует хотя бы один номер, который можно получить после замены звёздочек допустимыми символами (так, строка **AB_KZ1**** не будет присутствовать в тестах).

Формат выходных данных. В единственную строку выходного файла выведите искомое количество допустимых номеров.

из электронной библиотеки БГУ

Примеры входных и выходных данных

IX_N**9	315
N**Z128	50

Задача I: Подсчёт страниц

Имя входного файла: printer.in
Имя выходного файла: printer.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 1 гибибайт

Во многих прикладных программах при выводе на печать документа открывается диалоговое окно, в котором можно настроить принтер и выбрать страницы для печати. Как правило, множество номеров страниц, которые нужно напечатать, задаётся в текстовом поле строкой особого формата, например

```
9-14,23-27,8-4,24-27,9,8-8
```

Здесь запятыми разделены отдельные диапазоны печати. Каждый диапазон - это либо одно число (номер страницы), либо два числа, разделённые дефисом, которые задают отрезок страниц для печати. Диапазон «L-R» в случае, когда L больше R, просто игнорируется. Если диапазоны перекрываются и какая-то страница входит в более чем один диапазон, она будет напечатана один раз. Печатаются только существующие страницы: если диапазон включает номера страниц, которых нет в документе, они пропускаются. Страницы нумеруются с единицы.

В приведённом выше примере, если документ содержит 25 страниц, будут напечатаны страницы 8, 9, 10, 11, 12, 13, 14, 23, 24, 25, то есть всего десять страниц.

Напишите программу, которая по числу страниц в документе и введённой пользователем строке указанного формата определит, сколько в результате страниц напечатает принтер.

Формат входных данных. В первой строке содержится одно ло - количество страниц в документе. После этого записывается непустая строка, описывающая диапазоны страниц для печати. Длина этой строки не превосходит 100000 символов. Все числа во входном файле (количество страниц в документе и все номера, присутствующие в строке) целые, положительные и не превосходят 1 000 000 000.

из электронной библиотеки БГУ

Формат выходных данных. В единственную строку выходного файла выведите искомое количество страниц, которые должны быть напечатаны.

Пример входных и выходных данных

25 9-14,23-27,8-4,24-27,9,8-8	10

Задача J: Бело-чёрное дерево

Имя входного файла: wb-tree.in
Имя выходного файла: wb-tree.out
Ограничения по времени: 4 секунды
Ограничения по памяти: 1 гибибайт

После изучения темы «красно-чёрные деревья» студент Вася Пупкин предложил на семинаре по структурам данных использовать улучшенную (как ему кажется) версию этой структуры. Суть новой версии состоит в следующем:

- дерево состоит их N вершин и $N-1$ ребра, причём ни одна из вершин не выделена в качестве корня;
- каждое ребро выкрашено либо в красный, либо в чёрный цвет.

Для каких задач эта структура будет удобна, Вася пока не знает... Преподаватель, тем не менее, заинтересовался этой идеей, и предложил, во-первых, заменить красный цвет на белый (чтобы не создавать ненужной путаницы), и, во-вторых, задал Васе следующую задачу: «Перейти из вершины x в вершину y ($1 \le x, y \le N$, $x \ne y$), перекрашивая по пути все рёбра в противоположный цвет. На перекраску одного ребра тратится одно ведро краски. Сколько краски каждого цвета будет израсходовано при последовательном выполнении Q таких операций?»

Увы, Вася так и не смог справиться с этой задачей... Справитесь ли вы?

Формат входных данных. Первая строка файла содержит значения N и Q ($2 \le N \le 100\,000$, $1 \le Q \le 100\,000$). Далее следует $N - 1$ строка с описанием одного ребра дерева, содержащая номера вершин, которые оно соединяет, и первоначальный цвет: **W** (от англ. «white» — белый) или **B** (от англ. «black» — чёрный). Наконец, каждая из последующих Q строк соответствует одной операции перекраски и содержит номера вершин x и y.

Из электронной библиотеки БГУ

Формат выходных данных. Выведите Q строк, каждая из которых состоит из двух чисел: количество чёрной и белой краски, затраченной на выполнение очередной операции перекраски.

Примеры входных и выходных данных

5 5 1 2 W 2 3 W 3 4 W 4 5 W 1 5 1 5 1 4 1 3 1 5	4 0 0 4 3 0 0 2 3 1
7 5 1 2 B 2 3 W 1 4 W 4 5 B 1 6 W 6 7 B 2 3 3 5 5 1 5 4 5 7	1 0 1 3 1 1 0 1 3 1

из электронной библиотеки БГУ

Разбор задач

Четвёртый чемпионат (2011 год)

Первоначально планировалось включить олимпиаду БГУ в состав Открытого кубка по программированию им. Е.В. Панкратьева (Гран-при Беларуси). Набор задач, представленных на это соревнование, обычно достаточно сложен, поэтому при подготовке командного чемпионата школьников задачи студенческой олимпиады были взяты не в полном объёме (на обоих соревнованиях предлагались лишь задачи A – B, D – F, I). Остальные задачи были намеренно выбраны из более простых задач.

На наш взгляд, набор задач для школьного чемпионата получился достаточно удачным. Были решены все задачи, кроме очень сложных задач B и I. Победители командного чемпионата решили 7 задач из 9.

В соревновании приняло участие 39 команд, 26 команд решили хотя бы одну задачу. Как и во второй части книги, в таблице 1 показана оценка сложности задач с точки зрения их составителей.

Разбор выполнили С.И. Кашкевич (задачи A – H) , П.А. Иржавский (задача I).

Таблица 1

Задача	Оценка сложности	Количество решений
Сложить ящики	2.5	5
Кактус	5.0	0
Турнир по «минёру»	1.0	24
«Flood it!»	2.33	12
«Дурак»	1.0	24
«Дурак» – продолжение следует	3.0	7
Преобразование дроби	2.5	4
Гири и весы – 2	2.75	8
Квадрат	4.5	0

Задача A: Сложить ящики

Определим прежде всего условия, при которых ящик размерами $A \times B$ может быть вставлен в ящик размерами $C \times D$. Поскольку никаких

из электронной библиотеки БГУ

ограничений на соотношения между A и B, а также между C и D не накладывается, требуется проверить четыре условия:

$$\big((C-A)=2\big) \text{ and } \big((D-B)\geq 2\big)$$
$$\big((C-B)=2\big) \text{ and } \big((D-A)\geq 2\big)$$
$$\big((D-A)=2\big) \text{ and } \big((C-B)\geq 2\big)$$
$$\big((D-B)=2\big) \text{ and } \big((C-A)\geq 2\big)$$

Первый ящик можно вставить во второй, если выполняется хотя бы одно из этих условий.

Построим ориентированный граф, вершины которого соответствуют ящикам, а дуга ведёт из вершины x в вершину y, если ящик x можно вставить в ящик y. Задача сводится к поиску самого длинного пути в построенном графе. Очевидно, этот граф не имеет циклов, и при его анализе можно использовать идеи топологической сортировки.

Представим полученный граф списками смежности. Кроме того, для каждой вершины будем в массиве M хранить количество дуг, которые входят в эту вершину.

Назовём уровнем вершины x величину самого длинного пути в эту вершину. Вершины 0-го уровня найти просто – для них M[x] = 0. Удаляем эти вершины из графа, пересчитывая значения массива M. После этого повторяем этот алгоритм для 1-го и всех последующих уровней до тех пор, пока в графе будет оставаться хотя бы одна вершина. Ответ в задаче – максимальный уровень вершины + 1.

Ограничения задачи допускают, чтобы трудоёмкость алгоритма составляла O(N^2).

Задача B: Кактус

Казалось бы, нахождение кратчайшего пути между двумя вершинами в неориентированном графе – стандартная задача с хорошо известными алгоритмами её решения. Однако наличие большого количества запросов приведёт к тому, что лобовое решение, при котором каждый запрос обрабатывается независимо от других, будет слишком медленным. Можно, конечно, с помощью алгоритма Флойда найти матрицу расстояний между любыми парами вершин, но и этот алгоритм оказывается слишком медленным для заданных ограничений…

Выход заключается, во-первых, в учёте особенностей структуры графа, а во-вторых, в выполнении некоторых предварительных расчётов, результаты которых будут использоваться в дальнейшем.

из электронной библиотеки БГУ

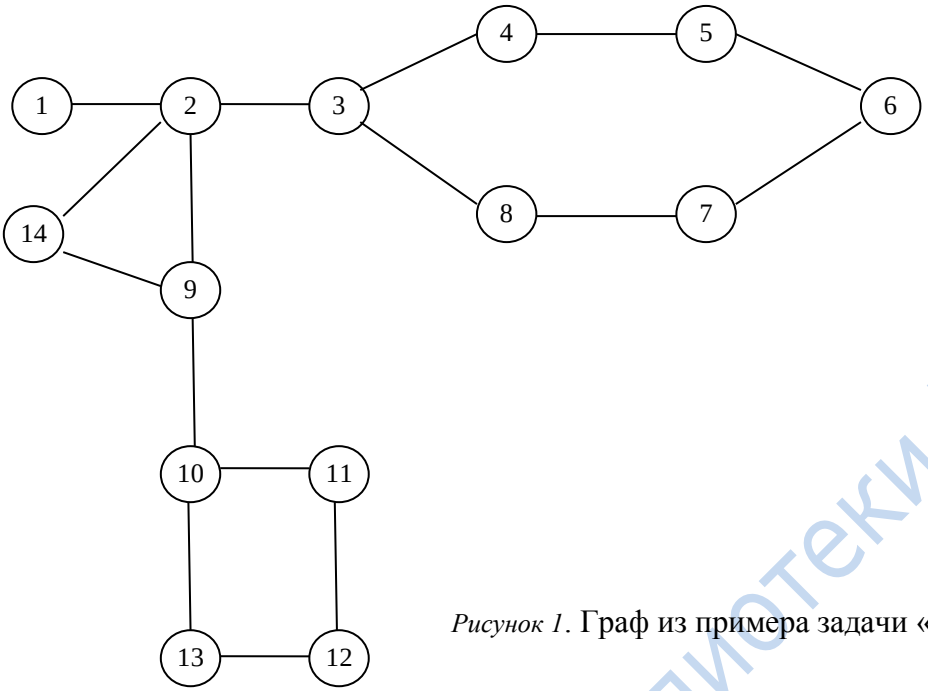

Рисунок 1. Граф из примера задачи «Кактус»

Рассмотрим граф, приведённый в примере к условию задачи (рисунок 1). Построим фиктивные циклы для всех вершин, которые изначально не входят ни в один цикл (в нашем примере – это вершина 1). После этого граф будет представлять собой совокупность циклов, соединённых *перемычками*. Так как циклы не пересекаются, то их легко выделить с помощью обхода графа в глубину.

Для каждого цикла (в процессе его нахождения) будем определять и хранить количество вершин p, входящих в этот цикл. Кроме того, присвоим вершинам, входящим в цикл, метки t_i – числа от 1 до p в порядке продвижения по циклу. Тогда расстояние между вершинами i и j, при условии, что они входят в один цикл, вычисляется за константное время и равно $\min(|t_i - t_j|, p - |t_i - t_j|)$. Кроме того, в каждом цикле выделим особую вершину, которая была посещена первой при обходе графа в глубину, и назовем ее *выходом*, все остальные вершины цикла будем считать *входами* в цикл.

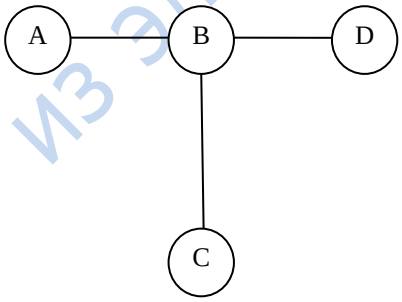

Рисунок 2. Замена циклов новыми вершинами

Заменим каждый цикл одной новой вершиной, как показано на рисунке 2 (вершина B, например, соответствует циклу из вершин 2, 9, 14). Полученный таким образом граф будет деревом, это следует из того, что каждая вершина исходного графа участвует

из электронной библиотеки БГУ

только в одном цикле.

Теперь путь из вершины i в вершину j, если эти вершины находятся в разных циклах, состоит из:

- путей внутри циклов, которые соответствуют начальной, конечной и промежуточным вершинам дерева;
- пути между вершинами дерева.

Заметим, что путь от вершины i в вершину j можно разбить на две части от i до ближайшего общего предка (least common ancestor, *LCA*) и от *LCA* до j. А в цикле *LCA* найти расстояний аналогично формуле описанной выше. Отметим, что расстояние между циклами на пути от i до *LCA* (и от *LCA* до j) будут взвешенными (*перемычка* и часть цикла от *входа* до *выхода*).

Итак, пути в циклах i, j и *LCA* рассчитываются за константное время. Для нахождения *LCA*, длин путей от i до *LCA* и от *LCA* до j необходимо использовать один из алгоритмов получения ближайшего общего предка (*LCA*), например, алгоритм Тарьяна, который особенно эффективно работает с серией запросов.

Таким образом, трудоёмкость алгоритма равна $O(N)$ для предварительного расчёта величин p и t_i и $O(N+Q)$ для обработки запросов. Этого вполне достаточно для решения задачи в существующих ограничениях.

Задача C: Турнир по «минёру»

Одна из двух «утешительных» задач, которую, по мнению авторов, должны были сдать все команды, за исключением абсолютно не готовых к соревнованию (а такие, увы, тоже были!).

Самый простой способ решения задачи – перенести входные данные в N массивов (для каждого игрока заводится отдельный массив), отсортировать каждый из этих массивов по убыванию, а затем использовать пять первых элементов для расчёта рейтинга. Такое решение имеет трудоёмкость $O(M \log M)$, и этого достаточно для решения задачи. Более аккуратный алгоритм предусматривает хранение для каждого игрока только пяти результатов (с обновлением этого массива в ходе чтения исходных данных) и имеет трудоёмкость $O(M)$. Однако ужесточение ограничений вывело бы задачу из разряда утешительных…

Задача D: «Flood it!»

Эта задача предлагалась и на студенческих соревнованиях, но с более высокими ограничениями. Алгоритм, который будет описан ниже, не

из электронной библиотеки БГУ

является оптимальным, но позволяет решить задачу с ослабленными ограничениями.

Заметим, что нам не нужно выделять все области, изначально образующиеся на карте. Достаточно строить единственную область, состоящую из захваченных клеток.

На начальном шаге формируем область, включающую клетку (1, 1), используя механизмы поиска в ширину. Далее в цикле от 1 до L пытаемся захватить новые клетки. При этом перед запуском поиска в ширину в очередь помещаем все клетки, захваченные ранее, так как мы не знаем, из какой из них может быть начат захват новых клеток.

После окончания расчётов остаётся перекрасить захваченную область в цвет последнего хода. Трудоёмкость описанного алгоритма $O(NML)$.

Продемонстрируем работу алгоритма на первом примере из условия задачи (захваченная область отмечена более тёмной заливкой), заполняя последовательно таблицы 2 – 5.

Начальное выделение областей не приводит к увеличению количества захваченных клеток:

Таблица 2

1	2	3	5
2	3	1	1
2	1	3	2

Расположение областей после первых двух ходов:

Таблица 3

2	2	3	5
2	3	1	1
2	1	3	2

Таблица 4

3	3	3	5
3	3	1	1
3	1	3	2

из электронной библиотеки БГУ

Последний ход не увеличивает захваченную область, а лишь изменяет её цвет:

Таблица 5

2	2	2	5
2	2	1	1
2	1	3	2

Задача E: «Дурак»

А вот и пример парных задач, которые на самом деле объединены только общей фабулой и форматом входных данных, а на самом деле для их решения требуется придумывать различные алгоритмы.

Первая задача является утешительной. Для её решения достаточно аккуратно запрограммировать условие «одна карта отбивает другую» и выполнить соответствующую проверку для всех карт, которые находятся на руках у второго игрока.

Сложность, возможно, представит лишь сравнение достоинств карт (напрямую сравнивать по обозначениям достоинств нельзя, необходимо заводить дополнительный массив из 9 элементов) и тот факт, что карта может обозначаться двумя или тремя символами. В последнем случае для достоинства выбирается первый символ, а для масти – последний. Таким образом, в программе логично обозначить десятку как ’1’.

Задача F: «Дурак» – продолжение следует

Вторая задача «дурацкой» серии решается с помощью механизма поиска в глубину.

Разобьём ход на два полухода: атака и отбивание, и будем искать карты, пригодные для выполнения очередного полухода. Первый шаг, естественно, заключается в переборе всех карт, находящихся на руках у атакующего игрока. В дальнейшем следует различать нечётные и чётные уровни поиска в глубину: первые соответствуют атаке, вторые – её отражению. Правила для выбора допустимых карт для продолжения атаки, естественно, будут различаться.

Глубина поиска не превысит 12, что, вместе с ограничением на количество карт, находящихся на руках у каждого из игроков, позволяет решить задачу за приемлемое время.

из электронной библиотеки БГУ

Задача G: Преобразование дроби

Для студентов эта задача является скорее учебной, чем олимпиадной – алгоритм преобразования числа с дробной частью из одной системы счисления в другую рассматривается в рамках курса «Программирование» в 1-м семестре. Школьники же этот алгоритм не изучают… Возможно, это и привело к заниженной оценке сложности задачи со стороны авторов.

Алгоритм перевода правильной дроби из системы счисления с основанием S в систему счисления с основанием T заключается в следующем. Записываем дробь в виде нескольких S-ричных цифр (возможно, с ведущими нулями) и умножаем это число на T. Целая часть произведения, величина которой, очевидно, меньше T, становится после перевода в новую систему счисления очередной цифрой результата. С дробной выполняем аналогичные вычисления. Алгоритм заканчивает работу, когда дробная часть становится равной нулю (и в этом случае мы получаем конечную дробь), либо когда дробные части, полученные на i-м и j-м шагах работы алгоритма ($i < j$), совпадают. В последнем случае выделяем периодическую и непериодическую части: в непериодическую часть попадают цифры с 1-й по i-ю, в периодическую – цифры с (i+1)-й по j-ю.

Рассмотрим работу алгоритма на втором примере из условия задачи.

$0.7 \times 16 = 11.2$ Целая часть образует первую шестнадцатеричную цифру B.

$0.2 \times 16 = 3.2$ Целая часть образует вторую шестнадцатеричную цифру 3. Поскольку дробные части, полученные на первом и втором шаге, совпали, периодическая часть дроби состоит из единственной цифры.

Входную дробную часть мы считаем результатом нулевого шага алгоритма.

Трудоёмкость алгоритма – $O(N^2)$, где N – количество полученных шестнадцатеричных цифр. Учитывая, что при небольших ограничениях на длину входных данных (6 символов), общее число шестнадцатеричных цифр не превышает 3200, этого вполне достаточно для решения задачи за приемлемое время.

Задача H: Гири и весы – 2

Задача решается с использованием идей динамического программирования.

Обозначим через t_i вес гири с номером i ($1 \le i \le N$), и через $F(k, P)$ – количество способов уравновесить груз весом P ($0 \le P \le S$) первыми k

из электронной библиотеки БГУ

гирями из набора ($1 \le k \le N$). Тогда ответом на задачу, очевидно, будет величина F(N, S).

Построим рекуррентное соотношение для вычисления F(k, P).

F(k, 0) = 1 для всех k (единственный способ уравновесить пустую чашку весов – ничего не класть и на вторую чашку).

F(1, t_1) = 1, F(1, P) = 0 для остальных P (единственной гирей можно уравновесить только мешок соответствующего веса).

F(k, P) = F(k – 1, P) для $P < t_k$ (гирю с номером k в этом случае использовать нельзя)

F(k, P) = F(k – 1, P) + F(k – 1, $P - t_k$) (первое слагаемое соответствует ситуации, когда гиря с номером k не используется, а в случае использования этой гири берётся второе слагаемое).

Таблица 6

K	1	2	3	4	5
T_i	1	2	3	4	5
0	1	1	1	1	1
1	1	1	1	1	1
2	0	1	1	1	1
3	0	1	2	**2**	2
4	0	0	1	2	2
5	0	0	1	2	3
6	0	0	1	2	3
7	0	0	0	2	3
8	0	0	0	**1**	**3**

Рассмотрим работу алгоритма на втором примере из условия задачи, построив значения F(k, P) (см. таблицу 6). Результат вычислений, а также ячейки, используемые в качестве слагаемых этого результата, выделены центрированием.

Заметим, что величина F(k, P) растёт очень быстро, и для заданных ограничений может достигать 30 десятичных цифр. Следовательно, для полного решения задачи требуется реализовать длинную арифметику, в частности, операцию сложения.

Задача I: Квадрат

Несложно найти сторону квадрата, если известен угол наклона стороны к оси координат: сторона равна большей из разностей наибольшего и наименьшего отклонений от прямой, проходящей через сторону, и перпендикулярной ей, то есть проходящей через смежную сторону.

из электронной библиотеки БГУ

Пусть f(α) равна наименьшей возможной стороне квадрата, если одна из его сторон наклонена к оси координат под углом α. Можно утверждать, что для угла α_0 функция f(α) будет достигать локального максимума только в том случае, когда прямая, проходящая через две точки из исходного набора, имеет угол наклона α_0 или $\alpha_0 + \pi/2$. В противном случае, поворачивая квадрат в любую сторону, т.е. увеличивая или уменьшая угол α, его сторону можно уменьшить.

Разбив интервал [0, $\pi/2$] углами наклона прямых, проходящих через пары точек, и перпендикуляров к ним, мы получаем серию интервалов, на каждом из которых f(α) заведомо унимодальна, так как внутри нет локальных максимумов. Значит, ответ на каждом из интервалов можно искать троичным поиском.

Получаем O(N^2) интервалов, O(|log ε|) итераций троичного поиска и O(N) для вычисления f(α) в каждой точке, итого O(N^3 |log ε|) операций, что достаточно для решения задачи.

Пятый чемпионат (2012 год)

Как и в 2011 году, задачи чемпионата частично пересекались с задачами студенческой олимпиады БГУ (на обоих соревнованиях предлагались лишь задачи A – B, D – E, G). Остальные задачи были намеренно выбраны из более простых задач.

На наш взгляд, набор задач для школьного чемпионата вновь получился удачным. Были решены все задачи, кроме сложной задачи G (всё-таки изучение вероятностей лежит далеко за пределами школьной программы). Победители командного чемпионата решили 7 задач из 8.

В этом году было принято решение провести отборочный интернет-тур, задачи которого были взяты из задач интернет-олимпиад ИТМО (базовый уровень). Это было связано, во-первых, с большим количеством заявившихся команд (ФПМИ не может предоставить 76 рабочих мест), и во-вторых, с опасениями, что в чемпионате примут участие абсолютно неподготовленные команды. Решение провести отбор, как показали его результаты, оказалось правильным: из 76 команд лишь 46 решили хотя бы одну задачу.

Кроме того, была установлена квота для иногородних команд (всё-таки это чемпионат Минска, а не Беларуси!). Для команд, не вошедших в квоту или для тех команд, которые по каким-то причинам не смогли приехать в Минск, была организована дополнительная сессия – интернет-соревнование на том же наборе задач, что и основной тур, и проходящее одновременно с основным туром. Участники дополнительной сессии, однако, не получают никаких наград.

Из электронной библиотеки БГУ

В основном соревновании приняло участие 27 команд, 25 команд решили хотя бы одну задачу. Из 13 команд, участвовавших в дополнительной сессии, хотя бы одну задачу решили 12 участников.

Как и ранее, в таблице 7 показана оценка сложности задач с точки зрения их составителей.

Разбор выполнили С.И. Кашкевич (задачи A – F, H), А.А. Толстиков (задача G).

Таблица 7

Задача	Оценка сложности	Количество решений
Я буду долго гнать велосипед…	2.63	2
И будет вам щасце!	3.13	2
Фонари у дороги – 1	1.67	20
Фонари у дороги – 2	2.18	2
Палиндромы	3.67	2
Прогрессия	1.0	25
Серии	3.50	0
По спирали	2.0	12

Задача A: Я буду долго гнать велосипед…

Относительно несложная задача, в которой, тем не менее, нужно показать умение использовать идеи теории графов, хотя в условии задачи графы явно не просматриваются.

Прежде всего определяем, на какой из дорожек находятся точки *A* и *B*. Затем находим точки пересечения велодорожек (таких точек может быть от нуля до четырёх).

Теперь необходимо рассмотреть два частных случая:

- если точки *A* и *B* находятся на разных дорожках и эти дорожки не пересекаются, ответом будет -1;
- если точки *A* и *B* совпадают, ответом будет 0.

Для решения задачи в общем случае строим взвешенный неориентированный граф, вершинами которого будут точки пересечения велодорожек, а также точки *A* и *B*. Рёбра графа соответствуют участкам одной велодорожки, которые соединяют две различные вершины, а веса этих рёбер равны наименьшему расстоянию, которое надо проехать по дорожке. Если точки *A* и *B* лежат на одной дорожке, добавляем в граф ребро, соединяющее соответствующие вершины.

из электронной библиотеки ТГУ

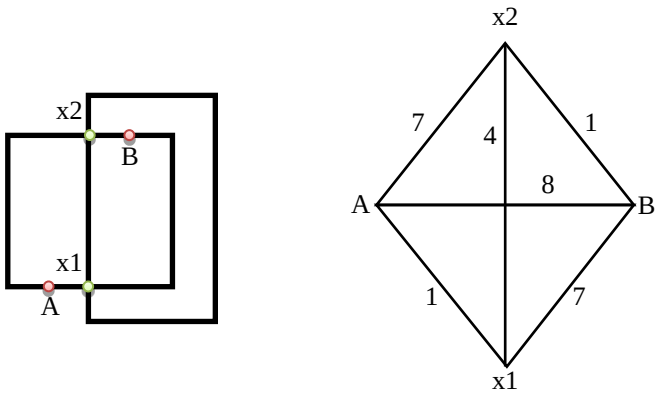

Рисунок 3. Расположение велодорожек и соответствующий ему граф

Теперь задача сводится к поиску кратчайшего пути между вершинами *A* и *B* в построенном графе.

На рисунке 3 изображена схема велодорожек и построенный на её основе граф для теста

```
0 4 4 0
2 5 5 -1
1 0
3 4
```

Задача B: И будет вам щасце!

Классическая задача о назначениях, только требуется найти максимум целевой функции, а не минимум.

Целесообразность включения этой задачи в соревнование состоит в том, что она изучается только на старших курсах вуза, а школьники должны сами придумать алгоритм её решения либо запрограммировать один из известных алгоритмов, если задачу о назначениях они рассматривали при подготовке к олимпиадам.

В наиболее общей форме задача формулируется следующим образом: имеется некоторое число работ и такое же число исполнителей. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы между исполнителями так, чтобы выполнить их с минимальными затратами.

Для решения этой задачи к настоящему времени разработано несколько алгоритмов, один из которых (т.н. *венгерский* алгоритм) будет изложен применительно к первому примеру из условия задачи.

3 2 4 2 1 2 2 2 5	Исходная матрица взаимных симпатий.
2 3 1 3 4 3 3 3 0	1. Переходим к задаче на минимум, заменяя S_{ij} на T - S_{ij}, где $T = \max_{i,j} S_{ij}$ Теперь матрицу можно трактовать как затраты на выполнение работ исполнителями.

Из электронной библиотеки БГУ

1 2 0 0 1 0 3 3 0	2. Вычитаем наименьший элемент строки из всех элементов этой строки. Повторяем эту операцию для всех строк. В результате в каждой строке должен появиться хотя бы один ноль. Если мы смогли распределить работы между исполнителями так, что суммарные затраты равны нулю, завершаем работу.
1 1 0 0 0 0 3 2 0	3. Выполняем аналогичные действия для столбцов. Выполнить распределение работ с нулевыми затратами по-прежнему нельзя.
1 1 **0** 0 0 ~~0~~ 3 2 ~~0~~	4. Находим любую строку с единственным нулём и фиксируем его (выделен жирным). Остальные нули столбца, в котором стоит фиксированное значение, зачёркиваем, они не будут рассматриваться на последующих шагах.
1 1 **0** **0** ~~0~~ ~~0~~ 3 2 ~~0~~	5. Выполняем аналогичные действия для столбцов.
1 1 **0** **0** 0 0 3 2 0	6. Проводим минимальное количество прямых, проходящих через все фиксированные нули.
0 0 0 0 0 1 2 1 0	7. Находим минимальный элемент в незачёркнутых клетках. Вычитаем его значение из всех клеток, через которые не проходят прямые, проведенные на предыдущем шаге. Добавляем это значение к клеткам, через которые проходят две прямые.
0 0 0 0 **0** 1 2 1 **0**	8. Решение найдено и выделено жирным. Если решение не найдено (например, на шаге 6 были проведены только вертикальные или горизонтальные линии), повторяем шаги 4 – 7.

Задача C: Фонари у дороги – 1

Одна из двух задач, которые должны были сделать практически все участники. Так, в принципе, и получилось...

Сортируем данные об освещённых участках по возрастанию значений a_i. Полагаем величину K (количество неосвещённых участков) равной 1, если $a_1 > 0$, и нулю в противном случае. Заведём две величины p и q – начало и конец текущего освещённого участка: $p = a_1$, $q = b_1$.

После этого просматриваем все освещённые участки, начиная со второго. Если $a_i > q$, увеличиваем K на единицу, накапливаем суммарную

из электронной библиотеки БГУ

длину всех освещённых участков и полагаем $p = a_i$, $q = b_i$. В противном случае, если $b_i > q$, полагаем $q = b_i$.

По завершении цикла вновь изменяем суммарную длину всех освещённых участков. Если $q \neq M$, то увеличиваем K.

Задача D: Фонари у дороги – 2

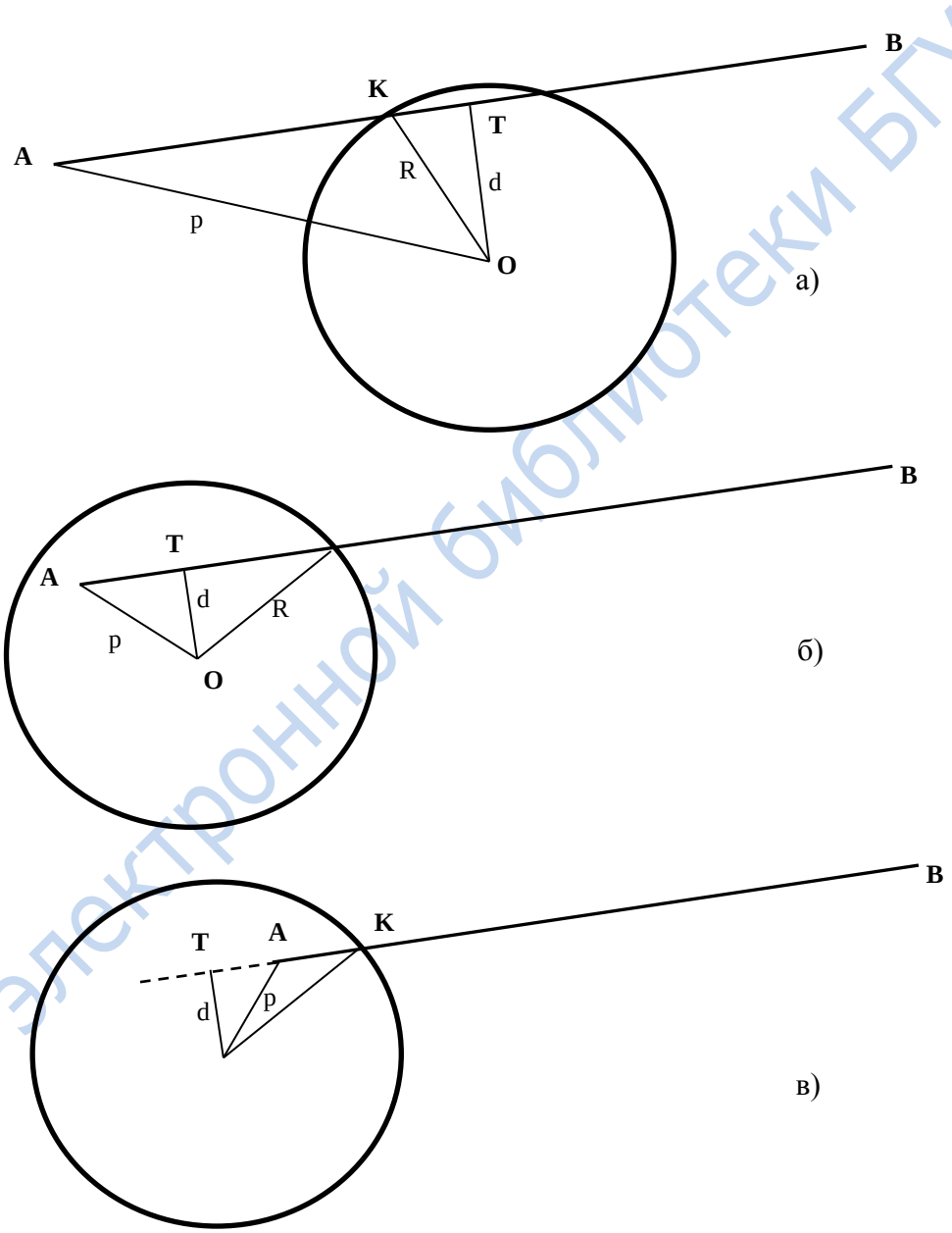

Рисунок 4. Взаимное расположение дороги и фонаря для задачи D

Усложнённый вариант предыдущей задачи. Однако в отличие от задач, разобранных во второй части книги («Расчёт сопротивления це-

из электронной библиотеки БГУ

пи», «Буква V»), нельзя сказать, что программа, решающая вторую задачу, автоматически решит и первую…

Решение состоит из двух этапов:

- определение значений $[l_i, r_i]$ для каждого фонаря;
- сведение задачи к предыдущей.

Рассмотрим идею решения первого этапа.

Для каждого фонаря рассчитываем три величины: расстояние от центра круга до концов отрезка дороги и до самой дороги. Обозначим эти величины соответственно через p, q, d.

Теперь рассматриваем четыре случая.

1) $d > R$. Фонарь не освещает дорогу, и в дальнейшем его можно исключить из рассмотрения.

2) $p > R > d$. Эта ситуация показана на рисунке 4а. Величина l_i, численно равная длине отрезка AK, рассчитывается из анализа прямоугольных треугольников OAT и OKT.

3) $R > p$, но проекция центра круга на участок дороги лежит на этом участке (см. рисунок 4б). В этом случае $l_i = 0$.

4) $R > p$, и проекция центра круга на участок дороги лежит вне этого участка (см. рисунок 4в). В этом случае $l_i = 0$, а значение r_i, численно равное длине отрезка AK, опять-таки вычисляется из анализа прямоугольных треугольников.

Для точки B проводятся аналогичные рассуждения.

Задача E: Палиндромы

Задача может быть сведена к следующей: проверить, является ли палиндромом строка, составленная из подмножества символов исходной строки S, и подсчитать количество таких палиндромов.

Задача решается с использованием механизма динамического программирования.

Пусть L – длина исходной строки, и a_i (i = 1, … , 26) – количество соответствующих букв латинского алфавита в ней. Положим M = L / 2, $b_i = a_i / 2$. Заметим, что в левой (правой) половине палиндрома частота i-го символа не может превосходить b_i, и при этом любая из двух половин может быть восстановлена из другой (разворотом). Следовательно, нам нужно посчитать, количество различных строк из символов исходной строки таких, что частоты символов не превосходят b_i, каждая из таких строк будет соответствовать какому-то палиндрому четной длины.

из электронной библиотеки БГУ

Введём функцию $F(k, t)$ – количество строк длины t, составленных из первых k букв латинского алфавита. Естественно, количество букв в этих строках не должно превосходить b_i.

Начальные значения функции F:

$F(0, 0) = 1$ (пустая строка считается палиндромом. По условию задачи такого быть не может, но это нужно для запуска рекуррентного соотношения);

$F(0, t) = 0$ для $t > 0$.

Теперь рекуррентное соотношение для расчёта $F(k, t)$ будет выглядеть так:

$$F(k,t) = \sum_{i=0}^{b_k} C_t^i \times F(k-1, t-i)$$

Таким образом, количество палиндромов чётной длины будет равно

$$R_1 = \sum_{t=1}^{M} F(26, t)$$

Для подсчета количества строк-палиндромов нечетной длины требуются дополнительные манипуляции. Для каждого символа, входящего в исходную строку S, пытаемся поставить его в середину палиндрома (изменяя тем самым величину a_i). После этого сводим задачу к рассмотрению предыдущего случая. Таким образом, число палиндромов нечётной длины будет равно

$$R_2 = \sum_{i=1}^{26} \sum_{t=1}^{M} G(26, t)$$

где функция G рассчитывается по формулам, описанным ранее для функции F, после изменения a_i. Ответом на задачу будет величина $R_1 + R_2$.

Не надо забывать о том, что результат должен быть рассчитан по модулю $10^9 + 2012$.

Задача F: Прогрессия

Самая простая, с точки зрения составителей, задача соревнования. Увы, три команды не смогли сделать и её…

Если $N = 2$, то ответ также равен двум. Поэтому будем рассматривать случай $N > 2$.

Изначально полагаем D равным $a_2 - a_1$ и запускаем цикл для $i = 3, \ldots, N$. Если $D = a_i - a_{i-1}$, увеличиваем счётчик длины текущей про-

из электронной библиотеки БГУ

грессии на единицу. В противном случае проверяем, не является ли текущая прогрессия максимально длинной, и изменяем величину *D*.

Аналогичную проверку на длину последней текущей прогрессии следует сделать и после завершения цикла.

Задача G: Серии

Эта задача также решается методом динамического программирования.

Заметим, несмотря на то, что в условии задачи нас просят найти вероятность некоторого события, несложно понять, что найти нужно значение отношения $F(n,r)/n!$, где F(n, r) – количество перестановок из n элементов, в которых ровно r серий.

Отдельно рассмотрим случай n=1. Тогда у нас есть всего одна перестановка, она состоит из одной серии.

Далее будем считать, что $n \geq 2$. Для того чтобы посчитать значение F(n,r), сначала выпишем следующие рекуррентные соотношения:

$$Inc(n,r,x) = \sum_{y=1}^{x-1} Dec(n-1,r-1,y) + \sum_{y=1}^{x-1} Inc(n-1,r,y)$$

$$Dec(n,r,x) = \sum_{y=x}^{n-1} Inc(n-1,r-1,y) + \sum_{y=x}^{n-1} Dec(n-1,r,y)$$

Inc(n,r,x) – количество перестановок из n чисел с r сериями и последним элементом, равным x, причем последняя из серий возрастающая, т.е. $\pi(n) > \pi(n-1)$. *Dec(n,r,x)* – количество перестановок из n чисел с r сериями и последним элементом, равным x, причем последняя из серий убывающая, т.е. $\pi(n) < \pi(n-1)$. Отметим, что *Inc(2,1,2)=1* и *Dec(2,1,1)=1*.

Далее требуется заметить, что суммы входящие в эти соотношения имеют одинаковые верхние (или нижние пределы), поэтому их можно переписать в следующем виде:

$$Inc(n,r,x+1) = Inc(n,r,x) + Dec(n-1,r-1,x) + Inc(n-1,r,x)$$

$$Dec(n,r,x) = Dec(n,r,x+1) + Inc(n-1,r-1,x) + Dec(n-1,r,x)$$

Эти соотношения можно вычислить за время $O(n^3)$, вычисляя их последовательно увеличивая n и r. Если значения *Inc(n,r,x)* и *Dec(n,r,x)* вычислены, то $F(n,r) = \sum_{x=1}^{x=n}(Inc(n,r,x) + Dec(n,r,x))$. Стоит отметить, что во избежание проблем с выходом за диапазон значений типа данных double лучше сразу считать *Inc(n,r,x)* и *Dec(n,r,x)*, делённые на n!. Использовать же длинную арифметику для точного вычисления этих значений нельзя, т.к. трудоемкость алгоритма в этом случае возрастет.

из электронной библиотеки БГУ

Задача H: По спирали

Разбиваем матрицу на четыре зоны, в зависимости от того, в каком направлении идёт заполнение этой матрицы. Один из вариантов такого разбиения представлен на рисунке 5. Для определения того, лежит ли элемент матрицы, например, в зоне I, можно проверять истинность выражения

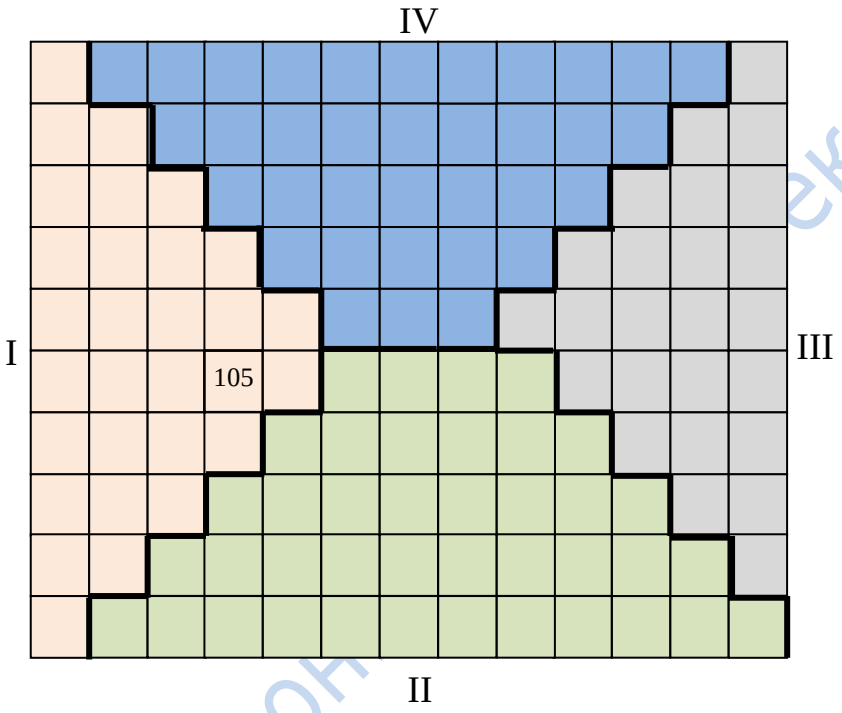

Рисунок 5. Разбиение матрицы размером 10×13 на зоны

$$\left(i < \left\lceil\frac{N}{2}\right\rceil\right) and \left(\left(\left(i < \left\lceil\frac{M}{2}\right\rceil\right) and\ (i \geq j)\right) or \left(\left(i \geq \left\lceil\frac{M}{2}\right\rceil\right) and (M - i + 1 \geq j)\right)\right)$$

Для других зон читателям предлагается вывести соответствующие формулы самостоятельно.

После того, как зона, в которой находится искомый элемент, определена, рассчитываем количество полных кругов, необходимое для заполнения матрицы до того, как начнётся последний (неполный) круг. Это количество численно равно расстоянию до границы зоны, в которой находится клетка (i, j), уменьшенному на единицу.

из электронной библиотеки БГУ

Первый круг состоит из $2 \times (M + N) - 4$ клеток, а количество клеток в каждом из последующих кругов будет отличаться от количества клеток в предыдущем круге на 8. Таким образом, количество клеток, содержащихся в T полных кругах, может быть рассчитано, как сумма арифметической прогрессии, и равно $2 \times T \times (M + N - 2 \times T)$.

Количество клеток, пройденных в последнем (неполном) круге до клетки (i, j), зависит от того, в какой зоне находится эта клетка. Для зоны I эта величина равна $i - j + 1$, соответствующие формулы для других зон предлагаем вывести самостоятельно.

Итак, значение элемента (6, 4) будет равно $6 \times (10 + 13 - 6) + 3 = 105$, что и показано на рисунке 5.

Шестой чемпионат (2013 год)

Как и в предыдущие годы, задачи чемпионата частично пересекались с задачами студенческой олимпиады БГУ, однако в этом году лишь одна задача C из командного чемпионата не была представлена на студенческих соревнованиях (вместо неё были предложены две гораздо более сложные задачи).

И вновь набор задач для школьного чемпионата вновь получился удачным (по мнению авторов, конечно). Сложные задачи G и J ожидаемо никто не решил, и даже ни одна команда не пыталась решить её. Увы, организаторы соревнования не рассчитывали на то, что очевидная по идее, но требующая скрупулёзной реализации задача H тоже не покорится ни одной из команд... Победители командного чемпионата решили 7 задач из 10.

Как и в 2012 году, было принято решение провести отборочный интернет-тур, задачи которого были взяты из задач командных чемпионатов г. Минска 2008 – 2009 годов. Причины проведения отбора объяснялись ранее. По результатам отбора на основной тур (с учётом квоты для иногородних команд) было приглашено 27 команд из 79 заявившихся. Одна команда не явилась на основной тур, но, как и на первом чемпионате, «никто не ушёл обиженным» - каждая команда, участвовавшая в основном туре, решила хотя бы одну задачу.

Следует сказать, что командный чемпионат, хотя формально и должен проводиться среди учащихся учреждений образования, подчинённых Комитету по образованию г. Минска, приобрёл общереспубликанский масштаб. Дипломы этого чемпионата получили представители всех регионов Беларуси.

Как и ранее, в таблице 8 показана оценка сложности задач с точки зрения их составителей.

из электронной библиотеки БГУ

Разбор выполнили С.И. Кашкевич (задачи A – F, H – I) , А.А. Толстиков (задачи G, H, J).

Таблица 8

Задача	Оценка сложности	Количество решений
«Балда»	2.67	10
Байкеры – 2	2.17	17
Что больше - 2?	2.0	9
Осторожно, злая собака!	1.0	26
Живая вода	1.83	17
Построить треугольник	2.5	13
Спички – не игрушка!	4.75	0
Испачканный номер	3.25	0
Подсчёт страниц	1.67	19
Бело-чёрное дерево	4.67	0

Задача A: «Балда»

Прежде всего проверяем, не совпадает ли полученное слово с начальным, записанным в средней строке (поскольку целью задачи является проверка соответствия придуманного слова правилам игры, такой вариант тоже надо иметь в виду!)

После находим положение первой буквы придуманного слова и запускаем поиск в глубину, пытаясь заново построить придуманное слово, соблюдая правила игры. Если это удалось сделать, и при этом одна из букв является последней записанной – выдаём положительный ответ. В противном случае ищем следующую позицию, в которой может находиться первая буква придуманного слова.

Задача B: Байкеры – 2

А вот и пример ситуации, когда сработала «плодотворная дебютная идея» (© О. Бендер) – при формулировке задачи авторы вспомнили старую «сказочку» и приспособили её для совершенно другой задачи…

Несложная задача, требующая, тем не менее, умения эффективно реализовывать алгоритм Дийкстры нахождения кратчайшего пути в графе. Найдем кратчайшие пути от вершины, соответствующей городу 1, до всех остальных вершин, используя веса a_i. После этого выполняем те же

из электронной библиотеки БГУ

действия, но для города N и весов b_i. Наконец, находим город, для которого сумма кратчайших весов будет минимальной.

Задача C: Что больше – 2?

Один из подходов к решению этой задачи заключается в том, что нам надо перевести число N_1 в систему счисления с основанием a_2, (либо наоборот, число N_2 перевести в систему счисления с основанием a_1).

Для перевода числа в систему счисления с основанием a_2 используется стандартный алгоритм деления этого числа на a_2 и выделения остатков. Однако деление должно выполняться в системе счисления с основанием a_1, что требует программной реализации этой операции.

После того, как оба числа переведены в систему счисления с одним основанием, сравниваем их, как строки. Более короткая строка соответствует меньшему числу, строки одинаковой длины сравниваются лексикографически.

Задача D: Осторожно, злая собака!

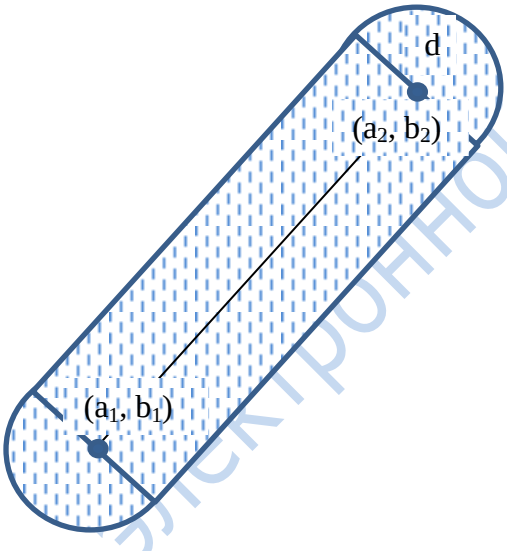

Рисунок 6. Участок земли, по которому может ходить собака

Самая простая задача соревнования. Не удивительно, что её решили все команды, прошедшие отбор и вышедшие на основной тур.

Как показано на рисунке 6, участок земли, по которому может ходить собака, представляет собой прямоугольник с длиной, равной расстоянию между кольями, и шириной $2d$, а также два полукруга радиуса d. Таким образом, площадь этого участка равна

$$\pi d^2 + d(\sqrt{(a2 - a1)^2 + (b2 - b1)^2}$$

Задача E: Живая вода

К сожалению, в условие этой задачи вкралась двусмысленность, на которую участники не указали во время проведения соревнования, но которая всплыла после его окончания. Авторы предполагали, что если бутылочку нельзя использовать полностью, то и покупать её не надо. Од-

ЭЛЕКТРОННОЙ БИБЛИОТЕКИ БГУ

нако условие можно понять и так, что бутылочку можно выпить не полностью, а оставшееся содержимое вылить на землю. Очевидная ошибка автора задачи ☹. Но, тем не менее, рассматриваем авторский подход…

Для решения этой задачи можно использовать идеи динамического программирования.

Обозначим через $F(u)$ минимальное количество денег, затраченное на покупку зелья, необходимого для того, чтобы довести здоровье персонажа до уровня u $(P \leq u \leq Q)$. Изначально

$$F(u) = \infty \text{ для } u > P, F(P) = 0$$

Затем выполняем проверку того, что покупка очередной бутылочки не превысит ресурсы героя и даст ему выигрыш в здоровье:

$$\text{если } F(u) \neq \infty \text{ и } F(u) + b \leq K, \text{то } F(u + a) = \min(F(u + a), F(u) + b)$$

Повторяем эту процедуру для больших, средних и маленьких бутылочек (именно в таком порядке!), а затем ещё 1000 раз (учитывая ограничения на количество денег у персонажа).

Задача F: Построить треугольник

В основе решения этой задачи лежит известная, но не изучаемая в рамках школьной программы формула для вычисления сторон треугольника по его медианам (предлагаем читателям вывести эту формулу самостоятельно). Пусть m_a, m_b, m_c – длины медиан, проведенных соответственно к сторонам A, B и C . Тогда

$$A = {}^{2}/_{3}\sqrt{2(m_b^2 + m_c^2) - m_a^2}$$

Аналогичные формулы строятся для расчёта B и C. Однако никаких ограничений на связь исходных данных между собой нет, поэтому треугольник может и не существовать (например, подкоренное выражение будет отрицательным, либо треугольник будет вырожденным). Поэтому в процессе вычислений надо осуществлять дополнительные проверки.

Задача G: Спички — не игрушка!

Задача оценивалась авторами как самая сложная, но в то же время, требующая только комбинаторных соображений, которые можно сделать без знаний университетской программы. Именно по этой причине она не была исключена из школьного набора задач.

из электронной библиотеки БГУ

Для начала вспомним, что в классической версии игры «Ним», упомянутой в условии задачи, первый игрок имеет выигрышную стратегию тогда и только тогда, когда «побитовое исключающее ИЛИ» размеров всех кучек камней не равно нулю. К примеру, для набора {1, 2, 3} выигрышной стратегии не существует, в том время как для набора {2, 3, 5} она есть.

Более того, нужно обратить внимание, что в каждой кучке может быть любое натуральное число спичек, в двоичной записи этого числа ровно *M* битов (возможно, есть лидирующие нули).

Пусть F(*n*) – количество способов выбрать n различных кучек и при этом «побитовое исключающее ИЛИ» размеров всех кучек камней ***не равно*** нулю. Пусть также G(*n*)=A(2^M-1,*n*)-F(*n*) – количество способов выбрать n различных кучек и при этом «побитовое исключающее ИЛИ» размеров всех кучек камней ***равно*** нулю (для краткости назовем такие наборы *плохими*), где $A(l,k)=l\cdot(l-1)\cdot\ldots\cdot(l-k+1)$ – количество размещений *k* объектов из *l*.

Тогда верно следующее соотношение:

$$G(n)=A\left(2^M-1,n-1\right)-G(n-1)-(n-1)\left(2^M-1-(n-2)\right)G(n-2).$$

Докажем корректность данного соотношения. Итак, для вычисления G(n) нам нужно выбрать n различных размеров кучек, для которых «побитовое исключающее ИЛИ» размеров всех кучек камней ***равно*** нулю.

Для начала следует заметить, что если набор *плохой*, то значение числа камней в любой из кучек равно «побитовому исключающему ИЛИ» всех остальных, а значит и последней кучки тоже. Поэтому, если мы возьмем корректный набор из *n*-1 кучки (их ровно A(2^M-1,*n*-1)), и размер *n*-ой кучки сделаем таким, что набор из всех *n* кучек будет *плохим*, то все такие наборы нужно посчитать (первое слагаемое).

Если размер *n*-ой кучки оказался равен нулю, а мы не можем делать пустые кучки, то набор из *n*-1 кучки *плохой* (их ровно G(*n*-1)), и мы зря посчитали такие наборы – вычитаем G(n-1).

Кроме того, у нас есть ограничение на различность всех размеров, если требуемый размер *n*-ой кучки уже использован в наборе, то такой набор мы также зря посчитали. Заметим, что пара совпадающих размеров у нас может быть только одна, т.к. предыдущие *n*-1 чисел уже различны. Без повторяющихся размеров мы имеем набор из *n*-2 различных

из электронной библиотеки БГУ

чисел, который является *плохим* (их ровно G(n-2)). Кроме того любой из этих наборов можно дополнить парой одинаковых чисел (любое из 2^M-1-(n-2)). Первое место, где элемент этой пары стоит, любое из n-1, а второе – последнее (n-ое). Таким образом, нам требуется еще вычесть произведение G(n-2)·(2^M-1-(n-2))·(n-1).

Для получения ответа на задачу воспользуется соотношением F(n)= A(2^M-1, n)-G(n). В заключение нужно добавить, что, так как ответ при ограничениях из условия задачи был бы очень большим, то все вычисления и вывод самого ответа требовалось проводить по некоторому модулю (1 000 000 007).

Задача H: Запачканный номер

Как уже было сказано ранее, эта задача требует тщательной реализации. Видимо, поэтому никто из школьных команд не смог рассмотреть все возможные случаи, хотя для студенческих команд эта задача нерешаемой не явилась. А ведь первоначально авторы планировали предложить эту задачу для второго этапа республиканской олимпиады школьников по информатике…

Рассмотрим два подхода к решению этой задачи.

Первый подход: вначале определяем, сколько символов содержится в коде региона. Конечно, если пробел распознался, то этот вопрос решается однозначно. Однако, если пробела среди распознанных символов нет, надо проверять дополнительные условия. Так, если длина номера больше, чем 8 символов, то длина кода региона не может быть равна единице. Этот же результат получится, если длина номера больше, чем четыре символа, но пятый символ – цифра.

Аналогичные рассуждения можно провести для длины кода региона в 2 и 3 символа. Однако однозначно определить позицию пробела невозможно. Поэтому отдельно рассматриваем случаи, когда код региона состоит из одного, двух и трёх символов. Во втором случае необходимо исключить из рассмотрения запрещённые комбинации.

Далее выполняем поиск серии (опять-таки с проверкой на запрещённые комбинации) и, наконец, выделяем собственно номер. После этого задача вычисления общего количества номеров не представляет особой сложности (хотя и требует аккуратного программирования).

Второй подход также содержит разбор различных случаев, но написание программы для этого решения оказывается более простым. Для начала разобьём все возможные номера машин на классы. В один класс попадут все номера с одинаковыми длинами кода региона (от 1 до

из электронной библиотеки БГУ

3 символов), серии (от 1 до 2 символов) и самого номера (от 1 до 4 символов).

Можем попробовать для каждого из 24 классов определить, сколько номеров из него подходит под запачканный. Для этого, во-первых, должна соответствовать длина номера (равняться сумме длин описанных выше блоков и еще один символ для пробела). Во-вторых, количество этих номеров должно быть равно произведению подходящих частей под каждый из 4 блоков (описанные выше и отдельный блок для пробела).

Можно сгенерировать отдельно все разрешенные комбинации прописных букв и все разрешенные комбинации цифр (лидирующие нули запрещены). А после этого посимвольно проверять, подходит ли часть номера под определенный символьный или числовой код.

Оценим общее число арифметических операций для предложенного решения. Возьмем самый большой для проверки класс: длина кода региона 3 символа (это нам дает 26^3=17 576), длина серии 2 символа (26^2=676, но нужно вычесть 5 запрещенных) и длина номера 4 символа (9 000 номеров без лидирующих нулей). На проверку всех этих вариантов нам понадобится не более 3·17 576+2·671+4·9 000=90 070 операций.

Так как мы хотим проверить в отдельности все 24 класса, то общее число операций не будет превосходить нескольких миллионов (даже если не использовать отсечение по суммарной длине блоков).

Задача I: Подсчёт страниц

Задача требует аккуратного разбора строки и выполнения нестандартной сортировки. Назовём *интервалом* последовательность подряд идущих страниц, которые необходимо напечатать. Очевидно, каждый интервал характеризуется номером первой и последней такой страницы документа.

В процессе разбора строки формируем массив интервалов. (Конечно же, при разборе строки следует выполнять проверку на все возможные ошибки!) После того, как массив интервалов сформирован, сортируем его по следующему критерию: интервал *A* считается меньшим, чем интервал *B*, если номер начальной страницы интервала *A* меньше номера начальной страницы другого интервала. Если номера начальных страниц этих интервалов совпадают, интервал *A* считается меньшим, чем интервал *B*, если номер его конечной страницы меньше номера конечной страницы другого интервала.

Сортируем массив интервалов по возрастанию в соответствии с приведенным критерием и выполняем последовательный просмотр эле-

из электронной библиотеки БГУ

ментов в отсортированном массиве. Пусть *A* и *B* – два соседних интервала, и *B* следует за *A*. Тогда следует рассмотреть три случая:

1) Интервалы *A* и *B* не пересекаются. Добавляем к счётчику числа страниц количество страниц в интервале A и переходим к рассмотрению интервала B;

2) Интервалы *A* и *B* частично пересекаются. Объединяем эти два интервала в один и находим интервал, следующий за объединённым;

3) Интервал *B* полностью входит в интервал *A*. Интервал *B* попросту игнорируется, и мы переходим к рассмотрению следующего интервала.

Конечно же, не забываем количество страниц и в последнем просмотренном интервале!

Задача J: Бело-чёрное дерево

Решение данной задачи предполагает, что читатель знаком с Heavy-light декомпозицией дерева. Если же это не так, то предлагаем ознакомиться с соответствующей теорией, например, здесь:

http://e-maxx.ru/algo/heavy_light.

Итак, разобьем наше дерево на непересекающиеся «тяжелые» пути. Каждый из этих путей можно поместить в отдельное дерево отрезков, которое будет позволять выполнять две операции:

a) перекраска ребер на отрезке;

b) определение количества белых ребер на этом же отрезке

Такая задача для дерева отрезков является стандартной и решается с помощью пометок вершин дерева количеством белых участков и меткой необходимости перекрасить все отрезки в поддереве. Читатель, знакомый с такой структурой данных, как декартово дерево, может заменить в этом решении дерево отрезков на нее.

Трудоемкость описанного решения $O(N + Q \cdot \log^2 N)$, что достаточно для того, чтобы уложиться в отведенные ограничения по времени.

из электронной библиотеки БГУ

Содержание

ПРЕДИСЛОВИЕ 3
УСЛОВИЯ ЗАДАЧ 4
Четвёртый чемпионат (2011 год) 4
Пятый чемпионат (2012 год) 14
Шестой чемпионат (2013 год) 22
РАЗБОР ЗАДАЧ 34
Четвёртый чемпионат (2011 год) 34
Пятый чемпионат (2012 год) 42
Шестой чемпионат (2013 год) 51

из электронной библиотеки БГУ

Учебное издание

Кашкевич Сергей Иванович
Толстиков Алексей Александрович

Задачи школьных олимпиад по информатике

Практикум для студентов факультета прикладной математики и информатики

В трёх частях

Часть 3

Задачи командных чемпионатов г. Минска по программированию (2011 – 2013 годы)

В авторской редакции

Ответственный за выпуск *С.И. Кашкевич*

Подписано в печать 05.11.2014. Формат 60×84/16. Бумага офсетная.
Усл. печ. л. 3,49. Уч.-изд. л. 3,0. Тираж 50 экз. Заказ

Белорусский государственный университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/270 от 03.04.2014..
Пр. Независимости, 4, 220030, Минск.

Отпечатано с оригинал-макета заказчика
на копировально-множительной технике
факультета прикладной математики и информатики
Белорусского государственного университета.
Пр. Независимости, 4, 220030, Минск.

из электронной библиотеки БГУ