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

С. И. Кашкевич, А. А. Толстиков

СБОРНИК ОЛИМПИАДНЫХ ЗАДАЧ ПО ИНФОРМАТИКЕ

Практикум для студентов
факультета прикладной математики и информатики

В пяти частях

Часть 2

МИНСК
2018

УДК 004(075.3)
ББК 32.81к2
К31

Рекомендовано советом
факультета прикладной математики и информатики
27 ноября 2018 г., протокол № 3

Р е ц е н з е н т
заведующий кафедрой дискретной математики и алгоритмики,
доктор физико-математических наук, профессор *В. М. Котов*

Кашкевич, С. И.

К31 Сборник олимпиадных задач по информатике : практикум. В 5 ч. Ч. 2 / С. И. Кашкевич, А. А. Толстиков. – Минск : БГУ, 2018. – 55 с.

Практикум содержит условия, а также разборы решений задач седьмого и восьмого школьных командных чемпионатов по программированию, проведенных в г. Минске в 2014–2015 годах.

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

УДК 004(075.3)
ББК 32.81к2

Предисловие

Школьные олимпиады по информатике проводятся уже достаточно давно. Каждый год они дают возможность школьникам получить льготы при поступлении в высшие учебные заведения, а в будущем – и возможность получить престижную работу, так как все больше различных ИТ-компаний на собеседованиях предлагают решить различные алгоритмические задачи. Именно на таких задачах и сфокусированы данные соревнования, а вместе с ними и данное пособие. Кроме этого, авторы считают, что решение олимпиадных задач по информатике позволяет развивать логическое мышление и находить оптимальные решения даже в самых повседневных ситуациях.

Авторы предлагаемого вашему вниманию практикума в течение многих лет составляют задачи для школьных и студенческих олимпиад различного уровня. Естественно, возникло желание предложить часть из накопленного опыта на суд общественности, издав сборники олимпиадных задач с их разборами. В 2012 году началась работа над изданием таких сборников. Три части, названные «Задачи школьных олимпиад по информатике», вышли в 2013 - 2014 годах. В 2016 году вышла первая часть практикума «Сборник олимпиадных задач по информатике».

Эти книги можно скачать из электронной библиотеки БГУ по следующим адресам:

elib.bsu.by/handle/123456789/41655 («Задачи школьных олимпиад по информатике», часть I)

elib.bsu.by/handle/123456789/48379 (часть II)

elib.bsu.by/handle/123456789/104667 (часть III)

elib.bsu.by/handle/123456789/147857 («Сборник олимпиадных задач по информатике», часть I)

Во второй части сборника авторы собрали условия заданий VII и VIII командных чемпионатов школьников по программированию, проведенных под эгидой факультета прикладной математики и информатики БГУ в 2014 – 2015 годах (всего – 23 задачи), а также показали свои подходы к решению этих задач. Однако мы советуем читателям не спешить читать разбор заданий. Постарайтесь сначала попробовать придумать математическую модель для задачи, где это возможно, найти алгоритм решения задачи, подумать над деталями реализации и даже реализовать свое решение на известном Вам языке программирования. Не забывайте обдумать различные крайние случаи для решения, так очень частыми

ошибками являются упущенные из вида случаи граничных значений параметров.

Тесты к задачам находятся по адресу:
https://yadi.sk/d/l2-Tu3tenh1BWg

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 А.С. Керножицкого, который внимательно прочёл рукопись и сделал 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.

Условия задач

Седьмой чемпионат (2014 год)

Чемпионат состоялся 28 сентября 2014 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 восемь команд.

Авторы идей задач – С.И. Кашкевич (задачи A – B, E, I), С.А. Соболь (задачи C, F – H), В.М. Котов (задачи D, J). Кроме них, в работе над задачами принимали участие А.А. Толстиков, П. А. Иржавский и В.П. Керус.

Задача A: Лошадью ходи, лошадью...

Имя входного файла: chevaux.in
Имя выходного файла: chevaux.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

На шахматной доске, состоящей из M строк и N столбцов, размещены два шахматных коня — белый и чёрный. Каждый конь располагается в одной клетке, но возможна ситуация, когда в одной и той же клетке находятся оба коня.

Кони делают ходы по очереди в соответствии с правилами движения шахматного коня (первым ходит белый конь). Целью игры является как можно более быстрое размещение обоих коней в одной клетке.

Сможете ли вы рассчитать количество ходов, необходимое для завершения игры, исходя из начального расположения фигур?

Формат входных данных. Первая строка файла содержит величины M и N ($2 \leqslant M, N \leqslant 1000$). Во второй и третьей строке записаны координаты клеток, в которых находится соответственно белый и чёрный конь. Первая координата находится в границах от 1 до M, вторая — в границах от 1 до N.

Формат выходных данных. Выведите единственное число — количество ходов, необходимое для завершения игры. Если кони никогда не смогут быть помещены в одну клетку, выведите −1.

Пример входных и выходных данных

8 10 2 4 7 9	4

Пояснение. Конь может пойти на любое поле доски, если она располагается на другом конце русской буквы Г (то есть вначале конь перемещается на два поля по горизонтали или по вертикали, а затем на одну клетку перпендикулярно первоначальному направлению). Выходить за границы доски, естественно, нельзя.

Задача B: Ров вокруг замка

Имя входного файла: ditch.in
Имя выходного файла: ditch.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Владелец старинного замка принимает в нём много туристов. Чтобы ещё больше увеличить их число, он решил восстановить давно засыпанный ров вокруг стен замка, доверху заполнить его водой и запустить в неё красивых рыб. Архитектор, которому поручена эта работа, убедил хозяина, что ров будет особенно красив, если его ширина будет одинаковой и равной A, а ближний к замку край рва будет отстоять от стен на одинаковое расстояние B.

Поскольку стоимость строительства рва зависит от площади поверхности воды, владелец попросил вас рассчитать эту площадь.

План замка представляет собой строго выпуклый многоугольник, а перепадами высот на месте строительства можно пренебречь.

Формат входных данных. Первая строка содержит три целых числа: N — количество вершин многоугольника, образующего замок ($3 \leqslant N \leqslant 1000$), а также A и B ($1 \leqslant A, B \leqslant 100$). Каждая из последующих N строк содержит два целых числа, не превосходящих по модулю 10000 — координаты очередной вершины многоугольника. Вершины перечисляются в порядке их обхода против часовой стрелки. Гарантируется, что многоугольник строго выпуклый.

Формат выходных данных. Выведите единственное число — площадь поверхности воды, которой будет заполнен ров, абсолютная или относительная погрешность вашего ответа не должна превышать 10^{-4}.

Пример входных и выходных данных

4 5 2 0 0 4 0 7 4 3 3	228.20040098

Задача C: Забор

Имя входного файла: fence.in
Имя выходного файла: fence.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Завод занимается выпуском стальных листов с отверстиями (см. рис. 1), из которых затем собираются ограждения, устанавливаемые вдоль дорог и вокруг стройплощадок.

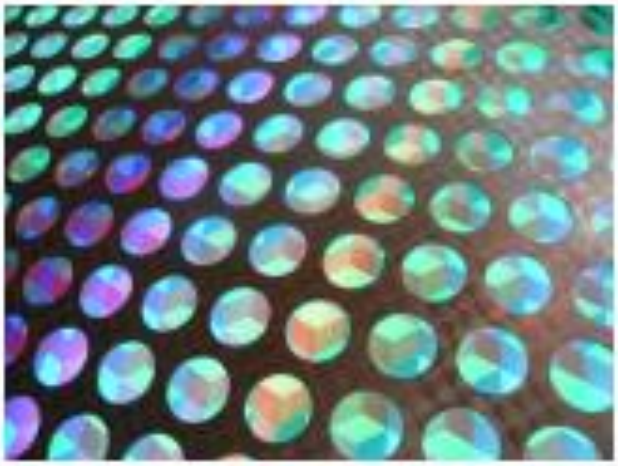

Рис. 1

Прямоугольная стальная перфорированная полоска имеет по k круглых отверстий радиуса r в каждом горизонтальном ряду. Отверстия отстоят друг от друга на расстояние $2d$. Расстояние от крайних отверстий до ближайшей границы полосы равно d. Таким образом, металлическая полоса имеет ширину $2k(d + r)$. Случай $k = 2$ показан на рисунке 2.

Процесс производства забора устроен следующим образом. Изначально имеется цельная полоса стали длиной L и шириной $2k(d + r)$. В ней вырезаются отверстия в соответствии с указанными правилами. Круглые куски стали, получившиеся после вырезания отверстий, и неиспользованный нижний остаток стального листа (показан на рисунке

штриховкой) сплавляются заново, чтобы получить вновь сплошную полосу прежней ширины. Затем опять делаются отверстия, лист отрезается, остатки идут на переплавку... Процесс повторяется до тех пор, пока очередная полоска не окажется столь короткой, что в ней нельзя будет сделать ряд из k окружностей с положенными отступами от края. Все листы стали в процессе производства имеют одинаковую однородную толщину.

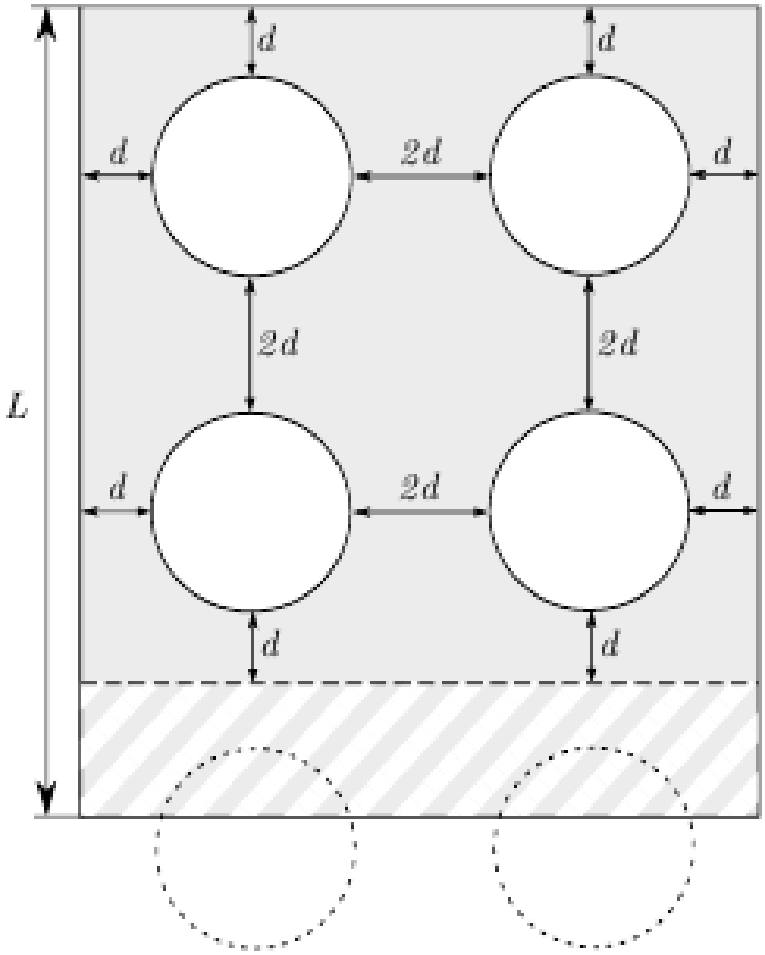

Рис. 2

Обозначим через C (d, r, k, L) общее число вырезанных в процессе производства отверстий при заданной величине промежутка d, заданном радиусе r, числе отверстий в горизонтальном ряду k и длине исходной сплошной полосы L. Аналитикам на заводе для оценки эффективности производства и выбора оптимальных параметров конвейера необходимо посчитать значения C для всех целых d от $d_{\min}$ до $d_{\max}$ и для всех целых r от $r_{\min}$ до $r_{\max}$. Осуществите эти расчёты и выведите на выход сумму

$$\sum_{r=r_{min}}^{r_{max}} \sum_{d=d_{min}}^{d_{max}} C(d,r,k,l)$$

то есть суммарное число отверстий для всевозможных вариантов организации производственного процесса при фиксированных k и L.

Формат входных данных. В единственной строке входного файла записаны шесть целых чисел: $r_{\min}$ и $r_{\max}$ ($1 \leqslant r_{\min} \leqslant r_{\max} \leqslant 10000$), $d_{\min}$ и $d_{\max}$ ($1 \leqslant d_{\min} \leqslant d_{\max} \leqslant 1000$), k ($1 \leqslant k \leqslant 100$) и L ($1 \leqslant L \leqslant 10^9$). Числа разделены одиночными пробелами.

Формат выходных данных. Выведите в выходной файл единственное число — искомую сумму.

Пример входных и выходных данных

4 4 3 3 2 14	2

Задача D: Вложение денег

Имя входного файла: investment.in
Имя выходного файла: investment.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

У вас неожиданно появились свободные деньги, и вы решили их выгодно вложить. Увы, за свою жизнь вы убедились, что для ведения собственного бизнеса у вас нет способностей, и вы решили вложить деньги в надёжные инвестиционные проекты.

Вы рассмотрели представленные заявки и выбрали N проектов, в которые можно вложить свои деньги без риска их потерять. Объём инвестиций в i-й проект ($1 \leqslant i \leqslant N$) составляет D_i, а доход, который вы рассчитываете получить после реализации этого проекта, равен T_i. Максимальный объём ваших инвестиций равен M.

К сожалению, все выбранные проекты начинаются одновременно, так что доход, полученный от одного проекта, инвестировать в другой нельзя... Кроме того, инвестировать в один проект более одного раза запрещается.

Определите максимальный доход, который вы сможете получить после реализации всех проектов, в которые вы вложили деньги.

Формат входных данных. Первая строка содержит величины N и M ($1 \leqslant N \leqslant 1000$, $1 \leqslant M \leqslant 10^9$). Далее следуют N строк, описывающих каж-

дый проект и содержащих величины D_i и T_i ($1 \leqslant D_i \leqslant 10^6$, $D_i < T_i \leqslant D_i + 100$).

Формат выходных данных. Выведите единственное число — максимальную сумму полученного вами дохода (с учётом сделанных инвестиций).

Примеры входных и выходных данных

3 10 7 11 4 6 5 8	15
3 10 11 20 13 14 20 21	10

Задача E: Попрыгунчик

Имя входного файла: jumper.in
Имя выходного файла: jumper.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Это школа Соломона Пляра,
Школа бальных танцев, вам говорят.
Две шаги налево, две шаги направо,
Шаг вперёд и две — назад!

Ваня любит придумывать новые и новые игры на клетчатых полях. Вот и сейчас он придумал ещё одну игру и предлагает вам оценить её качество.

«Попрыгунчик» — игра для одного игрока. Игровое поле — матрица размерами $M \times N$, где левая верхняя клетка имеет координаты (1,1), а правая нижняя — координаты (M, N).

Положению игрока соответствует одна фишка, занимающая одну клетку и характеризующая своей *стоимостью* — целым неотрицательным числом. Из клетки с координатами (i, j) фишка может совершить одно из следующих *перемещений*:

- *переход* в одну из клеток с координатами ($i + 1$, j), ($i + 1$, $j + 1$), (i, $j + 1$). Стоимость перехода равна a_{ij};

• *прыжок* в одну из клеток с координатами $(i + 2, j)$, $(i, j + 2)$. Стоимость прыжка равна b_{ij}.

Перемещение фишки не может быть выполнено, если его стоимость превышает стоимость фишки, либо если результат перемещения находится за пределами игрового поля. После перемещения в клетку с координатами (u,v) стоимость фишки уменьшается на величину стоимости перемещения, а затем увеличивается на величину t_{uv} — бонус за перемещение в соответствующую клетку.

В начале игры фишка стоит в клетке с координатами (1,1) и имеет стоимость Q. Определите максимальную стоимость фишки, стоящей в клетке с координатами (M, N) при соблюдении всех правил, указанных выше.

Формат входных данных. Первая строка файла содержит значения M, N, Q ($1 \leqslant M, N \leqslant 10000$, $1 \leqslant M \times N \leqslant 100000$, $1 \leqslant Q \leqslant 1000$). Далее следуют $M \times N$ строк, каждая из которых содержит три целых числа a_{ij}, b_{ij}, t_{ij} ($0 \leqslant a_{ij}, b_{ij}, t_{ij} \leqslant 1000$). Значения для индексов i, j записаны в строке с номером $N \cdot (i - 1) + j + 1$. Гарантируется, что $t_{11} = 0$, а также величины a_{ij} и b_{ij} равны нулю, если из соответствующих клеток нельзя совершить перехода либо прыжка.

Формат выходных данных. Выведите единственное число — максимальную стоимость фишки, стоящей в клетке с координатами (M, N). Если достичь клетки с этими координатами невозможно, выведите −1.

Примеры входных и выходных данных

4 3 10 1 2 0 4 6 20 3 3 5 5 4 8 7 6 1 5 7 4 3 2 1 1 0 2 2 0 5 4 6 20 10 0 10 0 0 5	35

3 4 8 10 20 0 4 6 20 3 3 5 5 4 8 7 6 1 2 0 5 4 0 20 10 0 10 4 2 8 6 5 3 1 0 4 0 0 5	-1

Задача F: Таблетки

Имя входного файла: pills.in
Имя выходного файла: pills.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Доктор прописал Васе принимать по полтаблетки определённого лекарства каждый день в течение $2N$ дней.

Вася купил в аптеке пузырёк с N одинаковыми таблетками. В первый день Вася извлёк из пузырька одну таблетку и разломал её пополам. Одну из получившихся частей таблетки он принял, а вторую забросил обратно в пузырёк.

В каждый из последующих дней Вася вытряхивал из пузырька случайно или целую таблетку, или половину. Если ему попадалась уже отломанная часть, он принимал её. Если же из пузырька высыпалась целая таблетка, Вася делил её пополам, выпивал назначенную ему дозу, а оставшуюся половину возвращал в пузырёк, чтобы выпить потом.

Сколько существует способов для Васи пройти назначенный доктором курс лечения? Можно представить один способ (последовательность элементов, которые доставал Вася из пузырька) как строку из $2N$ символов, i-й из которых — **A** (в i-й день Васе попалась целая таблетка) или **B** (в i-й день Васе попалась половина таблетки). Способы различны, когда соответствующие строки различны.

Формат входных данных. Во входном файле задано одно целое число N — количество целых таблеток в пузырьке ($1 \leqslant N \leqslant 30$).

Формат выходных данных. Выведите единственное число – количество способов пропить $2N$-дневный курс по полтаблетки в день.

Примеры входных и выходных данных

2	2
3	5

Пояснение: Во втором примере искомые способы соответствуют строкам **AAABBB**, **AABABB**, **AABBAB**, **ABAABB** и **ABABAB**.

Задача G: Манхэттенский почтальон

Имя входного файла: postman.in
Имя выходного файла: postman.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Дорожная сеть города представляет собой прямоугольную сетку размера $W \times H$ с единичным шагом: имеется W улиц, ведущих с севера на юг и пронумерованных в порядке с запада на восток натуральными числами от 1 до W, а также H улиц, которые идут в перпендикулярном направлении (с запада на восток) и нумеруются числами от 1 до H от самой северной до самой южной улицы. Таким образом, в городе есть $W \cdot H$ перекрёстков, и каждый перекрёсток однозначно соответствует точке на плоскости с парой координат (x, y). Каждый квартал города представляет собой квадрат со стороной 1.

Почтальону необходимо доставить N посылок для N адресатов, которые располагаются на N различных перекрёстках. Посылки довольно большие и тяжёлые, поэтому почтальон не может нести более одной посылки одновременно, и вынужден после вручения каждой посылки возвращаться за новой посылкой в почтовое отделение. Почтальон ходит только по дорогам. Когда последняя посылка будет доставлена получателю, возвращаться на почту уже не нужно: рабочий день почтальона заканчивается, и он отправляется по своим делам. Порядок доставки посылок может быть произвольным.

Определите, где должно располагаться почтовое отделение, и в каком порядке следует разносить N посылок, чтобы суммарное расстояние, которое проходит почтальон, было минимально.

Формат входных данных. В первой строке заданы два целых числа W и H — размеры города ($1 \leqslant W, H \leqslant 10^9$). Во второй строке задано число N — количество пунктов назначения ($1 \leqslant N \leqslant 10^5$). В последующих N

строках заданы по два целых числа x_i и y_i через пробел — координаты перекрёстков, к которым нужно доставить посылки ($1 \leqslant x_i \leqslant W$ и $1 \leqslant y_i \leqslant H$).

Формат выходных данных. В первой строке выведите одно число — минимальное расстояние, которое необходимо пройти работнику почты. Во второй строке выведите через пробел два числа x и y ($1 \leqslant x \leqslant W$ и $1 \leqslant y \leqslant H$) — место, где следует разместить почтовое отделение. Если мест, обеспечивающих оптимальное суммарное расстояние, несколько, выведите лексикографически минимальное (с минимальным x, а при равенстве x — с минимальным y).

Примеры входных и выходных данных

3 3 1 2 2	0 2 2
5 4 3 1 1 3 4 5 3	10 3 3
2 1 2 1 1 2 1	1 1 1

Пояснение. Во втором примере почтальон стартует из точки (3,3), отправляется сначала ко второму адресату в точку (3,4), проходя при этом расстояние 1, вручает посылку, затем возвращается на почту в (3, 3), проходя вновь расстояние 1. Далее почтальон направляется к третьему адресату в пункт (5,3), перемещаясь на расстояние 2, отдаёт посылку и возвращается обратно в (3,3), пройдя расстояние 2. Наконец, почтальон отвозит последнюю посылку из отделения (3,3) адресату (1,1), перемещаясь на расстояние 4. Таким образом, суммарный пройденный путь почтальона: 1 + 1 + 2 + 2 + 4 = 10.

В третьем примере не важно, где можно разместить почту: в точке (1,1) или в точке (2,1). Выбирается лексикографически минимальная пара чисел.

Задача H: Последовательные суммы

Имя входного файла: sums.in
Имя выходного файла: sums.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Сумма p последовательных целых положительных чисел иногда равна сумме следующих q последовательных положительных чисел. Например,

$9 + 10 + 11 + 12 = 13 + 14 + 15$ ($p = 4$, $q = 3$),

$4 + 5 + 6 + 7 + 8 = 9 + 10 + 11$ ($p = 5$, $q = 3$).

Для заданного числа q найдите, сколько существует подходящих p.

Формат входных данных. Во входном файле записано одно целое число q ($1 \leqslant q \leqslant 10^{14}$).

Формат выходных данных. Выведите одно число — количество подходящих значений p.

Пример входных и выходных данных

5	3

Задача I: Лабиринт с телепортами

Имя входного файла: teleport.in
Имя выходного файла: teleport.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 64 мебибайта

Лабиринт представляет собой совокупность из N комнат, некоторые из которых соединены M двунаправленными коридорами. Каждая пара комнат может быть соединена не более чем одним коридором. Вход в лабиринт находится в комнате с номером 1, а выход — в комнате с номером N. В некоторых комнатах находятся телепорты, перебрасывающие вошедшего в эту комнату в другое место лабиринта, минуя коридоры. В одной комнате расположено не более одного телепорта.

Телепорты срабатывают «через раз», иными словами, только одно из двух последовательных вхождений в комнату с телепортом приводит к его запуску. При этом одни телепорты срабатывают при нечетном вхождении (первом, третьем и т.д.), другие — при четном (втором, четвертом и т.д.).

Если при входе в лабиринт срабатывает телепорт, находящийся в комнате с номером 1, вы сразу же перемещаетесь по телепорту. Если при входе в комнату N срабатывает телепорт, вы перемещаетесь по нему, не успевая выйти из лабиринта.

Сможете ли вы пройти от входа к выходу лабиринта, используя его особенности? Рассчитайте длину кратчайшего из таких путей (под длиной пути понимается количество пройденных коридоров).

Формат входных данных. В первой строке записаны числа N и M ($2 \leqslant N \leqslant 100$, $1 \leqslant M \leqslant 200$). Далее следуют M строк, описывающих коридоры лабиринта и содержащих номера комнат, которые соединяет соответствующий коридор. В следующей строке содержится величина K — количество телепортов ($1 \leqslant K \leqslant 10$). Наконец, каждая из последующих K строк описывает один телепорт и содержит номера начальной и конечной комнат и число 1 или 2 — признак того, срабатывает ли телепорт при нечетном либо четном вхождении в комнату. Гарантируется, что у всех телепортов начальные комнаты различны.

Формат выходных данных. Выведите единственное число — длину кратчайшего пути от входа к выходу лабиринта. Если пройти лабиринт, по вашему мнению, невозможно, выведите −1.

Пример входных и выходных данных

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

Задача J: Раскраска тетраэдров

Имя входного файла: tetrahedron.in

Имя выходного файла: tetrahedron.out

Ограничения по времени: 2 секунды

Ограничения по памяти: 64 мебибайта

В вашем распоряжении имеется K бочонков с краской различного цвета, и вы хотите раскрасить краской из этих бочонков некоторое количество правильных тетраэдров. Каждая грань тетраэдра должна быть закрашена краской одного цвета, а для закраски тетраэдра можно использовать от одного до четырёх различных цветов (см. рисунок 3).

Рис. 3

Сколько существует различных вариантов раскраски четырёх граней одного тетраэдра? Раскраски двух тетраэдров считаются совпадающими, если после определённого поворота один из них полностью совпадает с другим.

Формат входных данных. Входной файл содержит единственное число K ($1 \leqslant K \leqslant 10^4$).

Формат выходных данных. Выведите требуемое количество вариантов раскраски.

Пример входных и выходных данных

2	5

Восьмой чемпионат (2015 год)

Чемпионат состоялся 17 октября 2015 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 идей задач – С.И. Кашкевич (задачи A – B, G – H), А.А. Толстиков (задачи D – E), С. А. Соболь (задачи C, F, I – L). Кроме них, в работе над задачами принимал участие А. Колесов.

Задача A: Путь ядра

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

Продольный разрез прямого жёлоба имеет вид, показанный на рисунке 4. Точки **A**, **B**, **C** и **D** имеют координаты соответственно (0, y), (x_1, 0), (x_2, 0), (x_3, y).

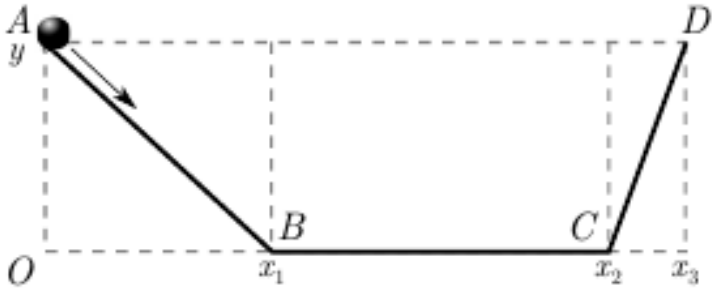

Рис. 4

В нулевой момент времени пушечное ядро находится в точке A и начинает скатываться в жёлоб. По участку **AB** ядро движется с ускорением a_1, по участку **BC** — с замедлением a_2, по участку **CD** — с замедлением a_3.

Если ядро останавливается на горизонтальном участке жёлоба (включая точки **B** и **C**), дальше оно не двигается. Если ядро останавливается на подъёме (не в верхней точке этого подъёма), оно сразу же начинает скатываться в обратную сторону. Ускорение на спуске **DC** численно равно a_3, а замедление на подъёме **BA** — a_1. Наконец, если ядро достигает точек A или D, оно вылетает из жёлоба.

Рассчитайте время от начала скатывания ядра до его полной остановки или до вылета из жёлоба.

Формат входных данных. Первая строка входных данных содержит величины x_1, x_2, x_3, y (целые положительные числа, не превосходящие 1000, $x_1 < x_2 < x_3$). Во второй строке записаны три целые величины a_1, a_2, $a3$ ($0 < a_1, a_3 \leqslant 100$, $0 \leqslant a_2 \leqslant 100$, $a_1 > a_2$, $a_3 > a_2$).

Формат выходных данных. Выведите единственное число — искомое значение времени, рассчитанное с относительной или абсолютной погрешностью не более 10^{-6}.

Примеры входных и выходных данных

10 50 70 20 5 3 5	7.975193500
10 50 70 20 5 0 5	17.312712690

20 60 70 20 25 5 7	3.422848811

Задача B: Ожерелье из монет

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

Петя нашёл баночку с утерянными кем-то советскими 15-копеечными монетами. Количество монет в баночке равно N. Теперь эти монеты — его новая игрушка!

Однажды Петя решил сложить из имеющихся монет ожерелье так, чтобы каждая монета касалась только двух соседних, а центры монет образовывали правильный N-угольник. И это ему удалось!

Но, когда Петя похвастался перед отцом своими успехами, тот попросил мальчика найти максимальное расстояние между точками, находящимися на поверхности монет (в том числе и на их краю). Увы, Петя не знает решения этой задачи... Помогите ему!

Формат входных данных. В первой строке входных данных записано число N ($3 \leqslant N \leqslant 10000$), а во второй — диаметр одной монеты (целое положительное число, не превосходящее 1000).

Формат выходных данных. Выведите единственное число — ответ на задачу. Ответ будет принят, если относительная или абсолютная погрешность не будет превышать 10^{-9}.

Примеры входных и выходных данных

3 6	12.0000000000
7 1	3.2469796037

Задача C: Набор номера

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

В крупных международных IT-компаниях для связи офисов, расположенных в разных городах, применяются системы видеотелефонии. В офисах организуют переговорные комнаты, оборудованные телевизорами и видеокамерами, при этом за каждой комнатой закреплён некоторый телефонный номер. Система видеотелефонии позволяет установить связь между двумя абонентами или организовать конференцию, в которой могут участвовать несколько абонентов. Если не вдаваться в подробности, звонок по видеотелефону с точки зрения пользователя осуществляется так же, как по обычному телефону: нужно набрать номер телефона вызываемого абонента на пульте дистанционного управления и нажать кнопку вызова. Трудность заключается в том, что на «эргономичных» пультах производства одной широко известной в узких кругах фирмы нет цифровой клавиатуры, а есть только клавиши управления курсором и бесполезная кнопка выключения микрофона (см. рисунок 5).

Рис. 5

Номер телефона приходится набирать нетривиальным способом. Изначально на экране телевизора отображается ряд из тринадцати символов: цифры от 0 до 5, точка, цифры от 6 до 9, звёздочка * и решётка #. Курсор установлен посередине — на символе точки.

0	1	2	3	4	5	.	6	7	8	9	*	#

Для набора номера можно использовать три кнопки пульта дистанционного управления. Нажатием кнопок «Влево» и «Вправо» можно сдвигать курсор на одну позицию в горизонтальном направлении. Последовательность символов зациклена, то есть левее самой левой клетки следует самая правая и правее крайней правой клетки идёт самая левая. При нажатии кнопки «Ввод» текущий символ, на который указывает курсор, добавляется к номеру.

Вам необходимо определить, за какое минимальное число нажатий клавиш пульта дистанционного управления можно набрать заданный номер.

Формат входных данных. В первой строке содержится число T — количество тестовых примеров ($2 \leqslant T \leqslant 10$). В последующих T строках

записаны непустые строки — номера, которые нужно набрать на видеотелефоне, по одному в строке. Длина каждого номера не превосходит 3000, а в его состав входят только символы, показанные в таблице, приведённой выше. В начале набора каждого номера курсор находится в исходном положении.

Формат выходных данных. Выведите T строк — минимальное количество нажатий на кнопки пульта дистанционного управления, необходимое для набора каждого номера.

Пример входных и выходных данных

3 1 123 1808	6 10 22

Задача D: Делимость Фибоначчи

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

Рассмотрим последовательность Фибоначчи

$$F_0 = 0,$$

$$F_1 = 1,$$

$$F_{n+1} = F_n + F_{n-1},\ n \geqslant 1$$

которая имеет вид

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, . . .

Сколько членов этой последовательности из K первых делятся на M?

Формат входных данных. Во входных данных задаются два целых числа K ($1 \leqslant K \leqslant 10^{18}$) и M ($1 \leqslant M \leqslant 10^6$).

Формат выходных данных. Выведите одно число – ответ на задачу.

Пример входных и выходных данных

10 2	4

Задача E: Делимость Фибоначчи (сложная версия)

Имя входного файла: fibonacci2.in
Имя выходного файла: fibonacci2.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Рассмотрим последовательность Фибоначчи

$$F_0 = 0,$$

$$F_1 = 1,$$

$$F_{n+1} = F_n + F_{n-1},\ n \geqslant 1$$

которая имеет вид

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, . . .

Сколько членов этой последовательности из K первых делятся на M?

Формат входных данных. В первой строке входных данных задается число тестовых примеров T ($1 \leqslant T \leqslant 100$). Далее в каждой из T строк содержится по два целых числа K ($1 \leqslant K \leqslant 10^{18}$) и M ($1 \leqslant M \leqslant 10^9$).

Формат выходных данных. Выведите T чисел — количество членов последовательности из K первых, которые делятся на M. Числа следует выводить по одному в строку.

Пример входных и выходных данных

2 100 2 1000000000000000 987654321	34 172125453293

Задача F: Хэллоуин

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

Через пару недель Васю ждут жаркие выходные, потому что приближается Хэллоуин! Вася хочет посетить как можно больше праздничных костюмированных вечеринок. Вечеринки, как правило, тематические, и нужно приходить на них в соответствующем облачении. Напри-

мер, направляясь на вечеринку фанатов комиксов, Вася выберет костюм Супермена, а на закрытую вечеринку своих коллег по цеху, спортивных программистов, Вася придёт в костюме бродячего торговца — коммивояжёра.

Поскольку Вася за ночь с 31 октября на 1 ноября хочет успеть сходить на несколько вечеринок и при этом быть в соответствующих костюмах, ему придётся переодеваться несколько раз. Чтобы было проще, Вася может надеть сразу несколько костюмов один на другой (например, костюм коммивояжёра поверх костюма Супермена). Перед каждой вечеринкой Вася может снять с себя несколько костюмов или надеть новый. Так, если форма коммивояжёра надета сверху костюма Супермена и Вася идёт на тусовку супергероев, он может либо снять костюм коммивояжёра, либо надеть наверх новый, ещё один, костюм Супермена.

К сожалению, Вася не любит надевать ношеные вещи вновь без предварительной стирки, поэтому после снятия формы коммивояжёра Вася, если ему вновь понадобится такой вид, возьмёт новую форму. За раз Вася может снять любое число костюмов, но, естественно, если он снимает k костюмов, то это будут те k костюмов, которые надеты последними (если костюм **A** надет перед костюмом **B**, то, чтобы снять **A**, надо предварительно снять **B**).

Зная, какие вечеринки запланированы в ночь Хэллоуина, определите минимальное число костюмов, которые нужно приготовить Васе.

Формат входных данных. В первой строке записаны целые числа N ($1 \leqslant N \leqslant 100$) — количество вечеринок — и M ($1 \leqslant M \leqslant 100$) — количество различных типов костюмов. Во второй строке записано N целых чисел c_1, ..., c_N, где c_i ($1 \leqslant c_i \leqslant M$) — тип костюма, который подходит к вечеринке i ($1 \leqslant i \leqslant N$). Вася посещает вечеринки в порядке 1, 2, . . . , N.

Формат выходных данных. Выведите единственное число – минимальное количество костюмов.

Примеры входных и выходных данных

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

Задача G: Ленивый поварёнок

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

После окончания поварских курсов Гриша устроился на работу в элитный ресторан. Но, естественно, на первых порах сложных заданий ему не поручают...

Вот и сегодня Гриша должен приготовить N видов свежевыжатых соков, которые заказали организаторы намеченного на вечер банкета. Для приготовления соков требуется K различных овощей и фруктов, причём в состав одного сока может входить от 1 до K компонентов. Объём чаши соковыжималки, которой будет пользоваться Гриша, а также количество исходных материалов достаточны для того, чтобы приготовить весь объём требуемого сока за один раз.

По технологии, чашу соковыжималки надо помыть после приготовления каждого сока. Но Грише очень не нравится эта операция... Он заметил, что если в состав следующего сока входят все компоненты, требуемые для изготовления предыдущего, то чашу можно и не мыть. Так, после приготовления яблочного сока можно обойтись без мытья чаши перед приготовлением яблочно-ананасового сока.

Помогите Грише и рассчитайте минимальное количество операций мытья чаши, которые потребуются для приготовления всех соков. Естественно, после выполнения работы чаша должна быть чистой!

Формат входных данных. В первой строке записываются целые величины N и K ($1 \leqslant N, K \leqslant 300$). Далее следуют N строк, каждая из которых описывает рецепт изготовления одного сока. Первое число этой строки m ($1 \leqslant m \leqslant K$) — количество компонентов сока. Далее следуют m различных целых чисел, каждое из которых находится в интервале от 1 до K включительно — номера компонентов, которые входят в состав соответствующего сока. Гарантируется, что набор компонентов различен для различных соков.

Формат выходных данных. Выведите единственное число — минимальное количество операций по мытью чаши.

Примеры входных и выходных данных

3 2	2

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

Задача H: Серии пенальти

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

Групповые турниры являются одним из этапов крупнейших футбольных соревнований. Суть группового турнира заключается в следующем: несколько команд формируют группу, участники которой играют по принципу «каждый с каждым» в один или два круга. На следующий этап проходят команды, набравшие по итогам игр в группе наибольшее количество очков. Так, в Лиге Чемпионов формируется восемь групп по четыре команды в каждой. По результатам группового турнира по две лучших команды из каждой группы продолжают борьбу в раунде плей-офф этого турнира, а команды, занявшие третьи места в группах, играют в Лиге Европы.

Проблема заключается в том, что не всегда можно определить победителей в группе только исходя из набранных в групповом турнире очков. Конечно, есть и дополнительные показатели, которые можно применить (разность забитых и пропущенных мячей, голы, забитые на поле соперника, и т. д.), однако и они не всегда срабатывают. Что делать, например, если все матчи в группе закончились с одинаковым счётом 1:1?

Одно из возможных предложений для разрешения этой коллизии состоит в реализации последовательных серий пенальти до выявления победителя. Пусть нам необходимо отобрать из N команд группы, имеющих одинаковые результаты по итогам группового турнира, Q лучших. Серия пенальти состоит из $N \cdot (N - 1)$ одиннадцатиметровых ударов, где каждая пара команд пробивает по одному удару в ворота друг друга. Рассмотрим случай, когда игрок команды **A** бьёт пенальти в ворота команды **B**. Если пенальти забит, одно очко получает команда **A**, в против-

ном случае — команда **B**. После окончания серии все команды ранжируются по убыванию набранных в этой серии очков. Если команды, занявшие Q-е и $(Q + 1)$-е места, набрали различное количество очков, коллизия считается разрешённой. В противном случае назначается новая серия для команд, набравших то же количество очков, что и команда, оказавшаяся на Q-м месте.

Авторы этого предложения нуждаются в его статистических исследованиях. В частности, они хотят узнать, насколько часто одной серии пенальти будет достаточно для определения команд, выходящих в следующий этап.

Помогите авторам, определив, достаточно ли будет проведения единственной серии по её результатам!

Формат входных данных. В первой строке записаны величины N и Q ($1 \leqslant Q < N \leqslant 100$). Далее следуют $N \cdot (N - 1)$ строк с результатами выполнения одного одиннадцатиметрового удара. Каждая такая строка содержит три числа: номер команды, игрок которой бьёт пенальти, номер команды, в ворота которой выполняется удар, и результат удара (1 — пенальти забит, 0 — нет). Нумерация команд начинается с единицы.

Формат выходных данных. Если проведения одной серии достаточно, первая строка должна содержать слово `yes`. Во второй строке записываются Q чисел — номера команд, прошедших на следующий этап турнира (в порядке возрастания этих номеров).

Если потребуются дополнительные серии, первая строка должна содержать слово `no`. Во второй строке записываются величины N_1 и Q_1 — количество команд в последующей серии и количество команд, которые необходимо отобрать для следующего этапа соревнований. Наконец, в третьей строке указываются N_1 чисел — номера команд, для которых будет проведена новая серия пенальти (в порядке возрастания этих номеров).

Примеры входных и выходных данных

4 2 1 2 1 2 1 1 1 3 0 3 1 0 1 4 0 4 1 1 3 4 0	yes 3 4

4 3 0 2 4 1 4 2 1 2 3 0 3 2 1	
4 2 1 2 1 2 1 1 1 3 0 3 1 0 1 4 0 4 1 1 3 4 0 4 3 0 2 4 1 4 2 1 2 3 1 3 2 1	no 2 1 2 3

Задача I: Авиабилеты

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

Цены на авиабилеты существенно колеблются неделю за неделей, и их непредсказуемость является основным источником разочарования для путешественников. Когда по мере приближения даты вылета цены падают, пассажиры жалеют, что купили билеты слишком рано. Другие путешественники, наоборот, начинают переживать, что не приобрели билет заранее, когда цена вдруг подскакивает. В итоге все огорчаются, довольны только сами авиакомпании.

Наверняка есть какая-то причина для этого безумия. Получается, что авиакомпании устанавливают цены на билеты динамически, основываясь на том, сколько остаётся свободных мест и как скоро состоится вылет. Например, если с начала продажи билетов салон быстро заполнился, цена на оставшиеся места будет держаться на высоком уровне вплоть до момента, когда до рейса останется пара недель. Цена может упасть, чтобы все билеты оказались проданы. Так или иначе, авиакомпании хотят увеличить свои прибыли от каждого полёта.

Вы работаете в авиакомпании «Байтландавиа», и ваша задача — установить цены на билеты по неделям. Авиакомпания собрала и проанализировала исторические данные, поэтому у вас есть надёжные оценки количества мест, которые будут проданы по определенной цене билета за определенное количество недель до полёта. Вам необходимо установить цену билета на каждую неделю, чтобы максимизировать выручку авиакомпании. Можно считать, что в каждую неделю при конкретной цене будет продано столько билетов, сколько прогнозируется, за исключением случая, когда свободных мест не хватает. В таком случае все оставшиеся места будут проданы.

Отметим, что повышение цен не обязательно означает, что будет продано меньше билетов. Более высокие цены могут иногда увеличить продажи, поскольку путешественники могут быть обеспокоены тем, что цены вскоре поднимутся ещё выше.

Формат входных данных. В первой строке записаны два целых числа — число оставшихся мест N ($0 < N \leqslant 300$) и число недель до вылета W ($0 \leqslant W \leqslant 52$). Следующие $W + 1$ строк содержат оценки для W недель, $W - 1$ недель, ..., и, наконец, одной недели перед вылетом. Каждая строка начинается с числа K ($1 \leqslant K \leqslant 100$) — количества различных цен для соответствующей недели. Далее следуют K чисел $0 < p_1 < p_2 < ... < p_K < 1000$, задающих цены в долларах. Далее в строке записаны ещё K чисел $s_1, s_2, ..., s_K$ (где $0 \leqslant s_i \leqslant N$), которые обозначают число билетов, продаваемых за определённую цену.

Формат выходных данных. В первой строке выведите максимальную общую выручку в долларах, которую может обеспечить авиакомпания от продаж, начиная с текущей недели до момента вылета. Во второй строке выведите $W + 1$ чисел — по какой цене нужно продавать в каждую из недель.

Примеры входных и выходных данных

10 1 1 100 8 2 200 300 7 9	1400 100 300
50 2 1 437 47 3 357 803 830 13 45 46 1 611 14	23029 437 830 611

Задача J: Радиационная безопасность

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

Атомные электростанции (АЭС) — это и благо, и беда современной цивилизации. С эксплуатацией АЭС связаны некоторые риски, но это по-прежнему один из самых дешёвых способов получения электроэнергии в развитых странах мира. В этой задаче мы будем рассматривать ситуацию в регионе, где две атомные электростанции находятся недалеко друг от друга.

Представим, что Земля плоская и на её поверхности задана двумерная прямоугольная система координат для определения положения точек. Пусть координаты двух атомных электростанций (a_x,a_y) и (b_x,b_y). В районе расположения АЭС экспертами определено N точек для возможного размещения автоматических пунктов радиационного контроля, которые будут осуществлять круглосуточный мониторинг радиационного фона. Оптимально было бы построить все N пунктов контроля, чтобы держать под наблюдением как можно большую площадь и оперативно реагировать на аварийный рост радиоактивности, но, к сожалению, бюджет проматомнадзора не резиновый...

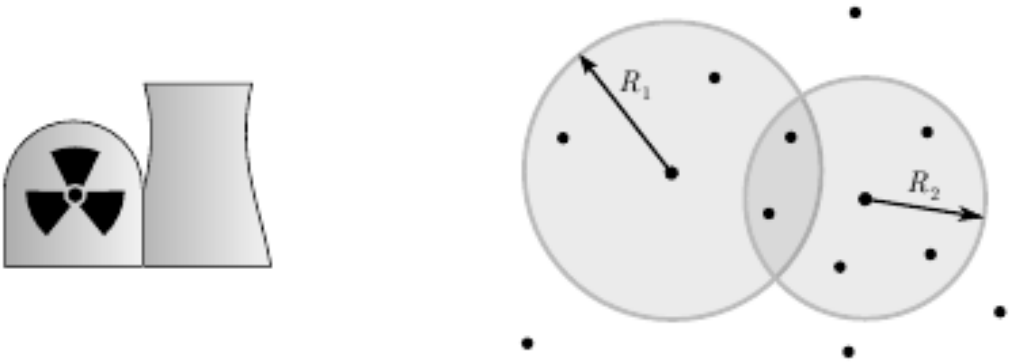

Рис. 6

В соответствии с действующим законодательством вокруг первой АЭС устанавливается зона наблюдения радиуса R_1, поэтому руководство станции финансирует строительство всех пунктов контроля, лежащих на расстоянии не более R_1 от станции. Аналогично, радиус зоны наблюдения около второй АЭС равен R_2, и строительством пунктов контроля, по-

падающих в этот радиус, по закону должна заниматься администрация второй электростанции. Однако, поскольку станции расположены рядом, некоторые пункты контроля из N указанных могут попасть сразу в две зоны наблюдения. Естественно, нет смысла в этих пунктах устанавливать два одинаковых комплекта дозиметрического оборудования. Рисунок 6 иллюстрирует эту ситуацию.

Однако деньги уже выделены и должны быть освоены, поэтому решено построить дополнительно столько пунктов контроля вне зон наблюдения, сколько пунктов попало одновременно в обе зоны наблюдения (если такие пункты, конечно, есть среди N запланированных, иначе оставшиеся деньги придётся осваивать другим способом).

Определите, сколько пунктов радиационного контроля из запланированных N так и не будет построено при разных значениях радиусов зон наблюдения, несмотря на рациональное использование бюджета.

Формат входных данных. В первой строке записано целое число N ($1 \leqslant N \leqslant 200000$) — количество спроектированных пунктов радиационного контроля. В последующих N строках записано по два целых числа x_i, y_i ($0 \leqslant x_i, y_i \leqslant 20000$) — координаты точки, где может быть расположен i-й пункт контроля. Гарантируется, что все точки различны. В следующей строке записано пять целых чисел: координаты двух АЭС a_x, a_y, b_x, b_y ($0 \leqslant a_x, a_y, b_x, b_y \leqslant 20\ 000$) и число запросов Q ($1 \leqslant Q \leqslant 20\ 000$). Точки расположения АЭС различны и не совпадают с точками установки пунктов контроля. Далее идут Q строк, в каждой записано по два целых числа R_1 и R_2, задающих радиусы зон наблюдения ($0 < R_1, R_2 \leqslant 13000$) вокруг первой и второй АЭС.

Формат выходных данных. Выведите Q строк. Для каждой пары радиусов зон наблюдения R_1, R_2 выведите одно число — количество пунктов дозиметрического контроля из N запланированных, которые так и не будут построены из-за недостатка финансирования.

Пример входных и выходных данных

11 95 75 27 6 93 5 124 13 34 49 65 61 81 49	2 2

77 33 110 50 91 22 110 25 57 42 97 36 2 31 25 25 25	

Задача K: Риск

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

«Риск» — настольная игра, местом действия в которой служит карта мира. Границы делят мир на регионы. Каждый регион находится под контролем игрока (это либо вы, либо один из ваших соперников). В каждом регионе, который вы контролируете, базируются ваши вооружённые силы: одна или несколько армий.

В течение хода вы можете изменять положение своих войск. Армия может либо остаться на месте, либо переместиться в смежный регион, также находящийся под вашим контролем. Движения выполняются одно за другим в любом порядке на ваше усмотрение. Однако в любой момент каждый регион должен содержать хотя бы одну армию.

Из стратегических соображений важно переместить свои армии в регионы, которые граничат с регионами соперников, и минимизировать число армий на внутренних регионах, которые окружены лишь регионами под вашим контролем. Ваша задача — добиться того, чтобы ваш слабейший рубеж, т. е. граничный регион с минимальным числом армий, был как можно более сильным. Какое наибольшее число армий можно разместить там?

Формат входных данных. В первой строке записано целое число N ($1 \leqslant N \leqslant 100$) — количество регионов в игре. Во второй строке записано N целых чисел a_1, a_2, ..., a_N ($0 \leqslant a_i \leqslant 100$), задающих число армий в каждом регионе. Если i-е число равно нулю, то регион i контролируется вашим соперником. Далее следуют N строк по N символов, каждый из которых Y или N. Если регион с номером i граничит с регионом j, то на j-м месте i-й строки стоит символ Y, иначе N. Отношение смежности регионов симметричное. Также гарантируется, что i-й символ i-й строки всегда N. Гарантируется, что под вашим контролем находится хотя бы один

регион, равно как и под контролем соперников есть хотя бы один регион. Более того, хотя бы один из ваших регионов граничит с регионом соперника.

Формат выходных данных. Выведите одно число — наибольшее число армий на вашем слабейшем рубеже.

Примеры входных и выходных данных

3 1 1 0 NYN YNY NYN	1
7 7 3 3 2 0 0 5 NYNNNNN YNYYNNN NYNYYNN NYYNYNN NNYYNNN NNNNNNY NNNNNYN	4

Задача L: Треугольники и четырёхугольник

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

Маленькому Евклиду всего два года, а он уже любит геометрию. Сейчас он занят тем, что пытается из двух бумажных треугольников составить определённый бумажный четырёхугольник. Фигуры можно двигать, вращать в плоскости и переворачивать. Треугольники не должны накладываться друг на друга.

Формат входных данных. Входные данные содержат несколько отдельных тестов.

В первой строке записано целое число T — количество тестов ($1 \leqslant T \leqslant 500$). Каждый тест описывается десятью строками, каждая строка содержит координаты x и y некоторой точки на плоскости. Вначале три строки задают координаты вершин первого треугольника. Следую-

щие три строки содержат координаты вершин второго треугольника. Наконец, следующие четыре строки задают координаты вершин четырёхугольника, заданные в порядке обхода по часовой стрелке или против часовой стрелки.

Гарантируется, что все фигуры имеют положительную площадь, четырёхугольник не имеет самопересечений и самокасаний, все его вершины попарно различны. Координаты являются целыми числами и по модулю не превосходят 15000.

Формат выходных данных. Выведите T строк. Для каждого теста выведите `Yes`, если из двух треугольников можно составить четырёхугольник, или `No` в противном случае.

Пример входных и выходных данных

2 0 0 0 1 1 0 0 0 0 1 1 0 -1 0 0 0 1 0 0 1 0 0 -1 1 1 0 0 0 0 1 1 0 -1 0 0 0 1 0 0 1	Yes No

Разбор задач

Седьмой чемпионат (2014 год)

Как и в прошлые годы, набор задач для командного чемпионата и студенческой олимпиады ФПМИ, проходившей в это же время, частично совпал (на студенческой олимпиаде вместо задачи E была предложена более сложная).

На наш взгляд, набор задач для школьного чемпионата получился удачным. Были решены все задачи, кроме задачи G. Победители командного чемпионата решили 8 задач из 10.

Как и в 2013 году, на материалах базового уровня интернет-олимпиад, проводимых в ИТМО (г. Санкт-Петербург), прошёл отборочный тур. Это позволило отсеять заведомо нулевые команды. Действительно, если 39 команд не смогли решить ни одной из шести простых задач – зачем их допускать к основному туру? В дальнейшем практика проведения отборочных туров станет традиционной для командных чемпионатов…

А теперь – о грустном… В 2014 году руководство Комитета по образованию Мингорисполкома сделало первый шаг к коммерциализации этих соревнований. Де-факто командный чемпионат г. Минска стал соревнованием, на которое съезжаются команды со всей Беларуси. Поэтому Комитетом по образованию было выставлено требование: взыскивать организационный взнос для команд, не представляющих г. Минск (в том числе и для участия только в отборочном туре). Это, на наш взгляд, привело к тому, что иногородние команды задумывались: стоит ли платить деньги, если нет шансов пройти на основной тур? В то же время минские команды подавали заявки, как и ранее. Неудивительно, что большинство отсеявшихся на отборе команд представляли г. Минск (за исключением команд Лицея БГУ, который формально Комитету по образованию не подчиняется).

В соревновании приняло участие 80 команд в отборочном и 26 команд в основном туре, 21 команда решила хотя бы одну задачу. Как и в предыдущих книгах, в таблице 1 показана оценка сложности задач с точки зрения их составителей. Обратите внимание на слабую корреляцию этих оценок с количеством принятых решений ☺.

Разбор выполнили С.И. Кашкевич (задачи A – B, E - F), А. А. Толстиков (задачи C – D, G – J).

Таблица 1

Задача	Оценка сложности	Количество решений
Лошадью ходи, лошадью…	2.33	21
Ров вокруг замка	1.63	9
Забор	2.88	1
Вложение денег	2.00	5
Попрыгунчик	2.13	19
Таблетки	1.50	15
Манхэттенский почтальон	3.33	0
Последовательные суммы	3.83	3
Лабиринт с телепортами	3.00	4
Раскраска тетраэдров	3.83	9

Задача A: Лошадью ходи, лошадью…

Прежде всего определяем, не находятся ли оба коня в одной клетке (в этом случае, очевидно, ответ равен нулю, и никаких дальнейших действий выполнять не надо).

В противном случае строим две матрицы W и B размерности $M \times N$ каждая. Их элементы равны минимальному количеству ходов, за которые конь белого или чёрного цвета достигнет соответствующей клетки доски (для этого используется механизм поиска в ширину). После этого для каждой клетки (i, j) рассчитываем величину r_{ij} – минимальное время для встречи коней в этой клетке. При расчётах рассматриваем четыре случая:

- $W_{ij} = B_{ij}$ или $W_{ij} = B_{ij} + 1$ (кони встречаются в этой клетке во время первого их захода в неё). В этом случае $r_{ij} = W_{ij} + B_{ij}$;
- $W_{ij} > B_{ij}$ и $W_{ij} - B_{ij}$ чётно (для повторной встречи чёрного коня в этой клетке белому коню придётся войти в неё дважды). В этом случае $r_{ij} = 2 * W_{ij}$;
- $W_{ij} > B_{ij}$ и $W_{ij} - B_{ij}$ нечётно. В этом случае $r_{ij} = 2 * W_{ij} - 1$;
- $B_{ij} > W_{ij}$ и $B_{ij} - W_{ij}$ чётно (для повторной встречи белого коня в этой клетке чёрному коню придётся войти в неё дважды). В этом случае $r_{ij} = 2 * B_{ijj}$.

В остальных случаях кони встретиться в клетке (i, j) не могут.

Ответом на задачу будет величина $\min\limits_{i,j} r_{ij}$.

Ответ -1 возможен лишь в случае, когда величины M или N меньше трёх.

А.С. Керножицкий предложил другое, гораздо более простое решение этой задачи: будем считать, что чёрный конь не движется, а белый идет в ту клетку, где изначально находится другая фигура. Тогда ответом будет полученное расстояние между двумя клетками…

Задача B: Ров вокруг замка

Задача, которую авторы отнесли к разряду утешительных, но, увы, нелюбимая школьниками геометрия…

Пусть периметр многоугольника с планом замка, рассчитанный по координатам вершин, равен P.

Ров состоит из прямоугольных участков, параллельных стенам замка, и закруглений вокруг его углов. Ширина прямоугольного участка равна A, а суммарная длина совпадает с периметром P. Закругление – это часть кругового кольца с внешним радиусом $A + B$ и внутренним B, лежащая внутри угла UOV (см. рисунок 7).

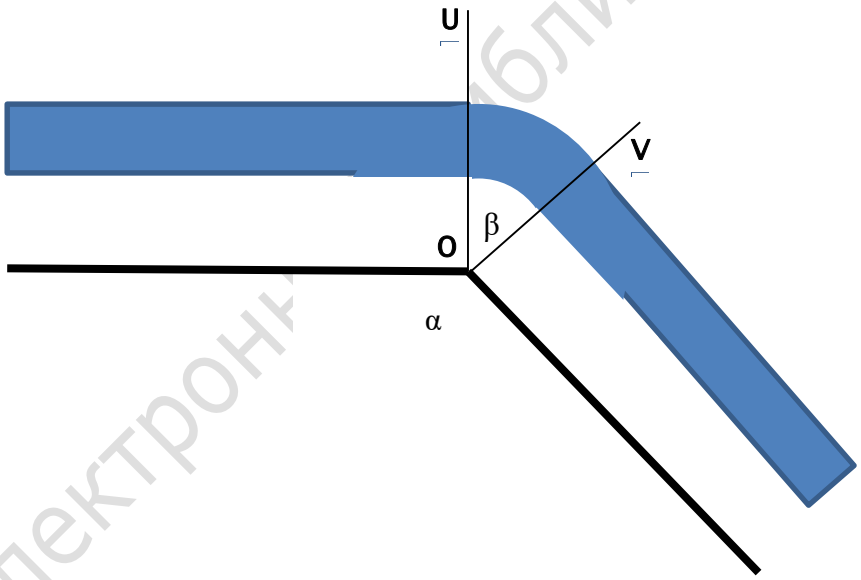

Рис. 7

Величина этого угла β равна, очевидно, π-α, где α – соответствующий внутренний угол многоугольника. Известно, что сумма внутренних углов выпуклого многоугольника с N вершинами равна $\pi \cdot (N - 2)$. Тогда сумма внешних углов β равна 2π, и все закругления складываются в полное кольцо.

Итак, искомая площадь рассчитывается по формуле

$$PA + \pi(A + B)^2 - \pi B^2.$$

Задача C: Забор

Для начала можно предложить достаточно медленное решение, но которое с небольшим изменением показывает процесс, который происходит при производстве забора.

Отметим, что ширина полоски $W = 2(d + r)k$, а, следовательно, площадь полоски изначально равна $S = WL = 2(d + r)kL$. Легко понять, что если в какой-то момент площадь оставшейся полоски $S < Sf = 2(d + r)W$, то больше уже вырезать перфорированного забора не получится. Будем отрезать от полоски по одному ряду кругов, а все остальное будем отправлять на переплавку, очевидно, что ответ для такого подхода не изменится, а вот процесс стал проще.

Еще заметим, что для вырезания одного ряда кругов мы тратим следующую площадь полоски $So = 2(d + r)W - k\pi r^2$, здесь площадь вырезанных кругов возвращается на переплавку.

А теперь заметим, что процесс не зависит от того, в какой момент мы отправили материал на переплавку, а только от дополнительного ограничения на последнюю вырезанную полоску.

Собирая все элементы решения, получаем, что

$\mathrm{C}(d, r, k, L) = \left(\left\lfloor \frac{S - Sf}{So} \right\rfloor + 1\right)k$, если $S \geq Sf$, иначе $\mathrm{C}(d, r, k, L) = 0$.

Задача D: Вложение денег

Данная задача является примером классической задачи упаковки рюкзака. Необходимо собрать набор предметов (проектов, в которые предполагается вложиться) наибольшей суммарной стоимости (получить наибольший доход).

Классический подход к решению этой задачи предполагает составление рекуррентного соотношения $F(W, i)$ – наибольшая суммарная ценность предметов, которую можно получить, выбрав предметы суммарным весом W из первых i вариантов предметов.

В нашем случае такой метод не подходит, т.к. разрешенный суммарный вес слишком велик – до 1 миллиарда.

Однако можно заметить, что возможная прибыль любого из проектов $T_i - D_i \leq 100$. В этой задаче предполагается «перевернуть» задачу о рюкзаке. Составим рекуррентное соотношение, которое позволит минимизировать суммарные вложения, для достижения суммарной прибыли P: $G(P, i) = \min(G(P, i - 1),\ G(P - (T_i - D_i), i - 1) + D_i)$.

Первая часть нашего соотношения $G(P, i-1)$ фактически обозначает «в i-й проект не инвестируем», а вторая часть $G(P-(T_i-D_i), i-1)+D_i$ означает «в -й проект инвестируем». Для инициализации значений динамического программирования можно использовать $G(P, i)$=∞ и $G(0,0)=0$, т.е. получить нулевую прибыль можно и без инвестиций.

Ответом на задачу будет максимальное значение $M+P$ при условии, что $G(P, N) \leq M$.

Задача E: Попрыгунчик

Задача решается с использованием механизмов динамического программирования.

Строим двумерную матрицу R размером $M \times N$, где $R(u, v)$ – максимальная стоимость фишки при её попадании в клетку (u, v). Изначально $R(1,1) = Q$, остальные значения равны -1, что соответствует недоступным клеткам.

Попасть в клетку (u, v) можно не более чем из пяти клеток: $(u\text{-}1, v)$, $(u\text{-}1, v\text{-}1)$, $(u, v\text{-}1)$ с использованием перехода и $(u\text{-}2, v)$, $(u, v\text{-}2)$ с использованием прыжка. Некоторые из этих клеток могут находиться за пределами игрового поля или быть недоступными; исключаем их из дальнейшего рассмотрения. Кроме того, исключаем из рассмотрения клетки (i, j), стоимость перемещения из которых в клетку $(u. v)$ больше, чем $R(i, j)$. В результате получаем множество клеток D, из которых может быть выполнено перемещение в клетку $(u. v)$. Тогда величина $R(u, v)$ рассчитывается по формуле

$$R(u, v) = t_{uv} + \max_{(i,j)\in D} C(i, j, u, v)$$

где $C(i, j, u, v)$ – стоимость перемещения (прыжка или перехода) из клетки (i, j) в клетку (u, v).

Ответом на задачу будет значение $R(M, N)$.

Задача F: Таблетки

Ещё одна задача на динамическое программирование…

Введём величину $F(a, b)$ – количество способов опустошить пузырёк, в котором находится a целых таблеток и b половинок ($0 \leq a, b \leq N$, $a + b \leq N$). Тогда, очевидно, ответом на задачу будет вычисленная величина $F(N, 0)$.

Выведем рекуррентные соотношения для $F(a, b)$.

1) $F(0, 0)=0$ (пузырёк уже пуст, достать из него ничего нельзя);
2) $F(0, b)=1$ для $b > 0$ (в пузырьке остались только половинки таблеток, и порядок их извлечения не имеет значения);

3) *F(a, 0)= F(a* - 1, 1*)* для $a > 0$ (в пузырьке остались только целые таблетки, и мы должны достать и разломать любую из них);
4) *F(a, b) = F(a* - 1, b + 1*)* + *F(a,* b – 1*)* для $a > 0$, $b > 0$ (мы можем достать либо целую таблетку и разломать её, либо половинку).

В качестве примера рассмотрим вычисление величины *F(a, b)* для $N = 4$.

b \ a	0	1	2	3	4
0	**0**	**1**	**2**	**5**	**14**
1	**1**	**2**	**5**	**14**	
2	**1**	**3**	**9**		
3	**1**	**4**			
4	**1**				

Задача G: Манхэттенский почтальон

Для начала переформулируем условие задачи. Задан набор точек (x_i, y_i), необходимо найти такую точку (X, Y) (координаты почтового отделения) и индекс f (номер последнего дома), что следующая сумма принимает минимальное значение:

$$|x_f - X| + |y_f - Y| + 2\sum_{\substack{j=1 \\ j \neq f}}^{N} (|x_j - X| + |y_j - Y|).$$

Лемма. Для множества чисел $a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_{2K-1}$ минимум выражения $\sum_{i=1}^{2K-1} |a_i - P|$ достигается при $P = a_K$.

Для доказательства леммы достаточно предположить, что P имеет другое значение, тогда при изменении этого значения на единицу в сторону a_{K-1} значение нашей суммы будет строго уменьшаться.

Воспользуемся этой леммой для решения нашей задачи. Предположим, что мы зафиксировали номер f дома, куда будет доставлена последняя посылка, тогда для выбора значений X и Y можно воспользоваться результатом леммы независимо, т.е. отдельно найти оптимальное значение X (среди $x_1, x_1, x_2, x_2, \ldots, x_{f-1}, x_{f-1}, x_f, x_{f+1}, x_{f+1}, \ldots, x_N, x_N$) и Y (среди аналогичной последовательности чисел).

Если проделывать действия, описанные в алгоритме выше, для каждого значения f отдельно, то программа будет делать квадратичное количество операций (если координаты отсортировать один раз предварительно) или квадратично-логарифмическое (если каждый раз сортировать новые последовательности длины $2N - 1$). Это слишком медленно, алгоритм следует еще ускорить.

Первый способ ускорить алгоритм заключается в том, что последовательности можно хранить в сбалансированном бинарном дереве или дереве отрезков. Тогда проверить индекс финального дома можно будет за логарифмическое время. Но авторы решили в разбор включить более элегантное решение, которое не требует знаний сложных структур данных, а что особенно полезно в решении соревнований на время, имеет очень простую реализацию.

Если внимательно посмотреть на последовательность, в которой мы хотим найти медианный (стоит в середине отсортированной последовательности) элемент, то, оказывается, он может принимать только два значения, если N четно, и только одно значение, если N нечетно.

Поэтому наше решение будет заключаться в том, чтобы попробовать одну или четыре позиции для почтового отделения. После этого легко найти сумму расстояний до всех домов. Легко видеть, что в этом случае для того, чтобы выбрать наилучший последний дом, из которого почтальон не будет возвращаться в отделение, достаточно взять самый отдаленный от почтового отделения.

Таким образом, нам необходимо отдельно отсортировать координаты домов, выбрать значения средних элементов в этих последовательностях (взять по два, если количество элементов четное), и попробовать скомбинировать эти значения. Для полученных координат почтовых отделений найти сумму расстояний и наилучшей финальный дом.

Время работы предложенного алгоритма линейно-логарифмическое.

К сожалению, никому не удалось решить задачу на соревновании.

Задача H: Последовательные суммы

В решении этой задачи нам потребуется функция количества делителей числа N из теории чисел: $\tau(N) = \tau({p_1}^{a_1}{p_2}^{a_2} \dots {p_k}^{a_k}) = (a_1 + 1)(a_2 + 1) \dots (a_k + 1)$.

Сумма p последовательных чисел $X, X + 1, \dots, X + (p - 1)$ равна сумме q последовательных следующих чисел $X + p, X + p + 1, \dots, X + (p + q - 1)$. Следовательно, $p > q$. Будем считать, что $p = q + k$, тогда имеем:

$$\frac{(q + k)(X + X + q + k - 1)}{2} = \frac{q(X + p + X + p + q - 1)}{2}.$$

Определим $2X$ из этого выражения, получим

$$2X = k - 1 - \frac{2q^2}{k}.$$

В левой части равенства стоит некоторое четное натуральное число, следовательно, в правой части также должно быть четное натуральное число. Следовательно, k является делителем $2q^2$, причем $k - 1 > \frac{2q^2}{k}$ и $k - 1 - \frac{2q^2}{k}$ четное число. Легко видеть, что одно из чисел k и $\frac{2q^2}{k}$ должно быть нечетным, а второе четным, причем, какое из них четное, значения не имеет.

Дальше внимательный читатель сможет собрать все вместе и заметить, что количество подходящих значений k равно количеству нечетных делителей $2q^2$ (примечание, один раз это число делится на 2 из-за положительности числа X, а потом еще умножается на 2 из-за того, что четным может быть либо k, либо $\frac{2q^2}{k}$).

Таким образом, нам необходимо разложить число q на простые множители, убрать простое 2, если оно есть в разложении. После этого достаточно подсчитать количество нечетных делителей q^2 по формуле из первого абзаца.

Задача I: Лабиринт с телепортами

Попробуем свести данную задачу к какой-нибудь классической задаче. Первым делом отметим, что лабиринт можно описать положением, где сейчас находится персонаж, которому надо выйти из лабиринта, и состоянием телепортов, которые, возможно, уже срабатывали.

Заметим, что описать состояние K телепортов можно битовой маской длины K – i-й бит будет равен 1, если -й телепорт сработает при следующем попадании в комнату, которая является его началом (к сожалению, в условии задачи эти комнаты никакого опознавательного идентификатора не имеют).

Таким образом, чтобы понять, где находится персонаж и какое сейчас состояние телепортов, достаточно $N \times 2^K$ – состояний, это будут вершины нашего графа. Для определения ребер графа посмотрим, что с персонажем может происходить:

- Он находится в комнате X, где сейчас сработает телепорт. В этом случае необходимо переместить персонажа в комнату, где эффект телепорта заканчивается и изменить состояние маски, описывающей телепорты (убрать один единичный бит). Обратите внимание, что телепорт срабатывает мгновенно, поэтому длина этого ребра будет равна 0.
- Персонаж находится в комнате Y, где не будет срабатывать телепорт. Он просто переходит по одному из коридоров, при этом не забываем изменить состояние телепорта, который начинается в комнате Y,

если таковой существует. Так как персонаж сам перемещается по коридору, то длина такого ребра будет равна 1.

Отметим, что общее число ребер в получившемся графе равно $O(M \times 2^K)$.

Таким образом, мы получили классическую задачу на 0-1 BFS, которая решается с помощью модифицированного обхода в ширину, только структуру данных «очередь» в этом случае следует заменить на «дек».

Общая трудоемкость решения – $O((N + M) \times 2^K)$.

Задача J: Раскраска тетраэдров

Решить эту задачу можно аккуратным разбором случаев.

Для начала подсчитаем, сколькими способами можно раскрасить тетраэдр в 1 цвет. Очевидно, что это число равно K, т.е. нам нужно только выбрать цвет, в который тетраэдр будет покрашен.

Попробуем подсчитать, сколькими способами можно раскрасить тетраэдр в 2 цвета. Для начала следует зафиксировать цвета, это можно сделать $C_K^2 = \frac{K(K-1)}{2}$ способами. Если мы поставим наш тетраэдр на грань, покрашенную в цвет 1, но у нас будут видны 3 грани, которые с точностью до поворота можно покрасить в 2 цвета (цвет 2 использовать обязательно!) тремя способами – фактически важно только количество граней, которые мы покрасим в цвет 2.

Небольшая хитрость: разбирая внимательно тест из условия, можно было количество раскрасок в 2 цвета получить из этой информации!

Далее необходимо разобраться с покраской в 3 цвета. Следует зафиксировать цвета, это можно сделать $C_K^3 = \frac{K(K-1)(K-2)}{6}$ способами. Вновь поставим тетраэдр на грань, покрашенную в цвет 1. Среди оставшихся граней обязательно будут грани покрашенные в цвета 2 и 3, а значит, только один из этих цветов будет иметь две покрашенные грани. Легко видеть, что это задает все возможные покраски в 3 цвета.

Немного более интересен случай покраски в 4 цвета. Следует зафиксировать цвета, это можно сделать $C_K^4 = \frac{K(K-1)(K-2)(K-3)}{24}$ способами. В каждых из этих цветов будет покрашено по одной грани, однако раскраску можно сделать двумя способами, которые нельзя получить один из одного с помощью поворота тетраэдра. Для того, чтобы представить эти две раскраски, нужно поставить тетраэдр на грань, покрашенную в цвет 1. После этого можно повернуть его таким образом, что грань, окрашенная в цвет 2, будет смотреть на вас. Далее, в зависимости от взаимного расположения граней с цветами 3 и 4 получаем две различные

раскраски (цвет 3 относительно взгляда может быть на левой или правой грани).

Собираем все вместе и получаем итоговую формулу:

$$C_K^1 + 3C_K^2 + 3C_K^3 + 2C_K^4.$$

Восьмой чемпионат (2015 год)

Как и в предыдущие годы, задачи чемпионата частично пересекались с задачами студенческой олимпиады БГУ (вместо задачи D студентам была предложена более сложная задача).

В отличие от прошлого года, сказать, что набор задач для школьного чемпионата получился удачным, сложно… Две задачи не были решены вообще (хотя попытки их выполнения были). Победители командного чемпионата решили 8 задач из 12.

В 2015 году Комитетом по образованию Мингорисполкома организация чемпионата была поручена ГУО «Минский городской институт развития образования». Это, на наш взгляд, понизило статус соревнования. Кроме того, организационный взнос было необходимо уплатить всем командам, участвовавшим в отборочном туре, независимо от региона. Это, по сравнению с предыдущим годом, резко уменьшило число заявившихся, но плохо подготовленных команд. Тем не менее, соревнование привлекло школьные команды со всей Беларуси.

Квота для иногородних команд, действовавшая на предыдущем чемпионате, была отменена (платят все, значит, участвуют все отобравшиеся).

В отборочном соревновании приняло участие 33 команды, в основном – 29, 26 из которых решили хотя бы одну задачу.

Как и ранее, в таблице 2 показана оценка сложности задач с точки зрения их составителей. Заметьте, что корреляция между оценкой сложности с точки зрения авторов и количеством решений заметно увеличилась по сравнению с 2014 годом (за исключением задач A и B)!

Разбор выполнили С.И. Кашкевич (задачи A – C, H), А.А. Толстиков (задача D –G, I – L).

Таблица 2

Задача	Оценка сложности	Количество решений
Путь ядра	2.50	1
Ожерелье из монет	1.50	18
Набор номера	1.00	26
Делимость Фибоначчи	2.5	13

Делимость Фибоначчи (сложная версия)	4.5	0
Хэллоуин	2.33	2
Ленивый поварёнок	3.33	1
Серии пенальти	1.25	20
Авиабилеты	2.17	12
Радиационная безопасность	2.00	4
Риск	3.50	2
Треугольники и четырёхугольник	3.75	0

Задача A: Путь ядра

Для авторов было неожиданностью то, что только одна команда решила эту несложную в принципе задачу. А ведь задача предлагалась на гораздо более низком уровне – для районных олимпиад г. Минска 2013 года! Хотя свою роль сыграло и различие в форматах соревнований ACM и IOI – думается, что много участников районной олимпиады набрало бы частичные баллы.

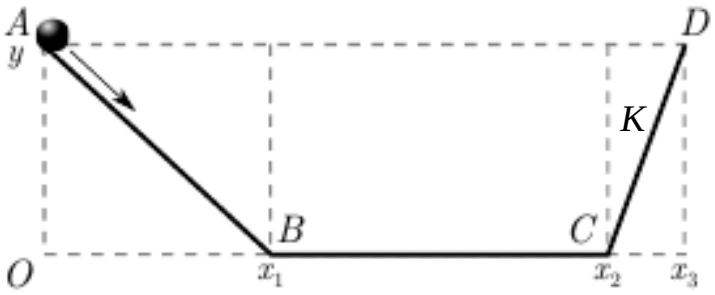

Рис. 8

На рисунке 8 показан разрез желоба, описанный в условии задачи.

Первым шагом для решения является вычисление времени t_0 спуска на участке **AB** и скорости v_0 в точке **B**. Эта информация понадобится нам при дальнейших вычислениях.

Вначале рассмотрим случай $a_2 = 0$ (именно некорректное рассмотрение этого случая привело к неверному результату во многих решениях). В этом случае при начальном спуске скорость ядра в точках **B** и **C** равна v_0. Теперь, если эта скорость достаточна для вылета ядра в точке **D**, задача решается «в одно действие». Если же ядро останавливается на участке **CD,** не достигая точки **D**, оно скатывается обратно в точку **C**, достигая её на скорости v_0, с этой же скоростью доходит до точки **B** и поднимается к точке **A**, достигая её со скоростью 0, но тем не менее выкатываясь из жёлоба.

При $a_2 > 0$ необходимо последовательно рассмотреть следующие случаи:

a) разгона, полученного на спуске, недостаточно для достижения точки **C**, и нам необходимо найти время t_1 до остановки ядра на участке **BC**. Результатом в этом случае, очевидно, будет $t_0 + t_1$;

b) ядро разогналось достаточно, чтобы достичь точки **C**. В этом случае необходимо рассчитать время t_1 прохождения участка **BC** и скорость v_1 в точке **C**. После этого переходим к рассмотрению следующих случаев:

c) скорости v_1 достаточно, чтобы достичь точки **D** и вылететь из жёлоба. Рассчитываем время t_2 подъёма на участке **CD**. Результатом в этом случае, очевидно, будет $t_0 + t_1 + t_2$;

d) в противном случае находим точку **K**, в которой ядро останавливается. Рассчитываем время t_2 подъёма до точки **K**. Запоминаем величину $t_0 + t_1 + t_2$ в качестве времени, потраченного на первый проход и повторяем вычисления, предполагая, что спуск начинается в точке **K**. После нескольких итераций мы придём, в конце концов, к случаю a).

Вывод соответствующих формул авторы предоставляют читателю.

Задача B: Ожерелье из монет

Ещё одна недооценённая авторами задача. Опять геометрия...

Задача сводится к построению правильного N-угольника, длина стороны которого равна диаметру монеты. Обозначим этот диаметр через D.

Угол между двумя соседними вершинами многоугольника и его центром равен $\alpha = \frac{2\pi}{N}$, а расстояние от центра многоугольника до любой из его вершин равно $d = \frac{D}{2\sin{\alpha}/2}$. Теперь рассмотрим три случая:

1) $N = 3$. В этом случае искомое расстояние равно удвоенному диаметру монеты.

2) N чётное. В этом случае искомое расстояние равно $2d + D$.

3) N нечётное и большее 3. Рассчитываем расстояние от вершины 1 до вершины с номером $k = N/2-1$ (эта величина равна $\sqrt{2d^2 * (1 - \cos 2\pi \frac{k}{N})}$) и добавляем к полученному результату диаметр монеты.

Задача C: Набор номера

А эту задачу должны были сделать все (увы, трём командам задача оказалась не под силу...)

Записываем строковую константу, порядок символов в которой соответствует порядку символов на пульте:

```
S1 := '012345.6789*#';
```

После этого задаём начальную позицию курсора, равную 7 (согласно условию, вначале курсор находится в позиции точки). После этого для каждого символа в исходном номере, начиная со второго, вычисляем смещение относительно предыдущего символа. Если модуль смещения больше 6, пытаемся двигаться в другую сторону. Вот и всё!

Задача D: Делимость Фибоначчи

Для решения этой задачи полезны следующие наблюдения:

- Фактически нам нужно найти количество нулей в последовательности $G_i = F_i \ (mod\ M)$.
- Читатели с хорошей математической подготовкой могут доказать, а любители эвристик заметить (напечатав несколько последовательностей), что для любого значения M существует значение P, что все $G_{jP} = 0$, и только они. Например, если $M = 5$, то G_i: 0 1 1 2 3 0 3 3 1 4 0 4 4 3 2 0 2 2 4 1 0 1 ... легко видеть, что $P = 5$.

  Если $M = 6$, то G_i: 0 1 1 2 3 5 2 1 3 4 1 5 0 ... в этом случае $P = 12$.
- Найдя минимальное значение $P > 0$, что $G_P = 0$, мы сразу можем выписать ответ на задачу $\left\lceil \frac{K}{P} \right\rceil$.

В рамках участия в олимпиадах по программированию достаточно догадаться до такого решения, что сделали большинство участников, и не доказывать, что требуемое значение P достаточно мало (не превосходит периода последовательности G, который не превосходит $6M$). Либо знать известный факт про числа Фибоначчи – период Пизано. Авторы предлагают читателю найти соответствующую информацию в литературе или сети Интернет, например, в Википедии.

Задача E: Делимость Фибоначчи (сложная версия)

Для решения этой задачи только интуиции уже недостаточно.

Первый метод больше подходит для очень сильных в математике участников, или тех, кто интересовался свойствами периода Пизано (и что очень важно, хорошо в них разобрался).

По разложению числа M на простые множители мы можем построить число Q, которое будет гарантированно кратно P, но, скорее всего, не будет ему равно. После этого можно постепенно проверять делители Q,

пока не найдем минимальный подходящий. Тут и далее нам пригодится быстрый способ вычисления чисел Фибоначчи:

- Воспользуется соотношением

$G_{n+m} = G_{n-1}G_m + G_nG_{m+1}\ (mod\ M).$ $(*)$

- Реализуем рекурсивное вычисление деления пополам ($m = n$ или $m = n - 1$) с запоминанием, например, в контейнере std::set.

Авторы сознательно не излагают все выкладки данного подхода, поскольку их несложно найти в литературе. К тому же, он не являлся самым ожидаемым от участника.

Второй метод является более «программистским», он основан на так называемой идее маленького и большого шага. Тут нам вновь понадобится быстрый способ вычисления числа Фибоначчи с большим индексом.

Отметим достаточно очевидный факт, но очень полезный в решении задачи. Если $G_x = G_y$ и $G_{x+1} = G_{y+1}$, то для любого значения $z \geq 0$ $G_{x+z} = G_{y+z}$. Доказательство этого факта можно провести методом математической индукции.

Для решения задачи будем искать первое повторение пары $(0,1) = (G_0, G_1)$. Среди будущих соседних элементов последовательности $(0,1)$ встретится не позднее чем через $6M$ шагов.

Предположим, что пара $(0,1)$ повторно встретится через Z шагов. Будем искать Z в следующем виде: $Z = aX + b$, где выполнены следующие условия $X \sim \sqrt{6M}$ и $0 \leq b < X$ $\left(\text{тогда } 0 \leq a < \frac{6M}{X}\right)$. Для нахождения разложения нам нужно «запомнить» (записать в подходящую структуру данных) X первых пар (G_i, G_{i+1}). После этого нужно начать двигаться из пары (G_0, G_1) с шагов в X переходов сразу, для этого мы можем использовать соотношение (*). Через $(a + 1)$ шаг мы попадем в одну из «запомненных» пар соседних элементов, отсюда сможет найти и подходящее значение b. Обратите внимание, что перепрыгнуть сразу все запомненные пары мы не можем, так как их количество равно размеру прыжка.

Оба предложенных решения работают за время $O\left(\sqrt{M}logM\right)$.

Задача F: Хэллоуин

Данная задача является очень хорошим упражнением на использование техники динамического программирования.

Заметим, если один и тот же костюм использовался на вечеринках a и b, и при этом не использовался между данными вечеринками, то мы можем считать, что Вася оптимально использовал некоторые новые костюмы между этими вечеринками, а перед вечеринкой b снял все эти но-

вые костюмы. Вторым интересным замечанием является такой факт: если Вася пришел на вечеринку в подходящем костюме, то нет смысла одевать такой же новый.

Пусть $F(a, b)$ – оптимальное количество новых костюмов, которые потребуются Васе, чтобы посетить вечеринки с номерами от a до b включительно, и потом все эти костюмы снять. Очевидно, что ответом на задачу будет величина $F(1, N)$.

Составим рекуррентное соотношение для $F(a, b)$:

$$F(a,b) = \begin{cases} 0, \text{если } a > b, \\ 1, \text{если } a = b, \\ \min\{1 + F(a+1, b), 1 + F(a+1, s-1) + F(s, b) | c_a = c_s\}. \end{cases}$$

Первая часть нашего соотношения означает «Если интервал событий пустой, то и костюмы не понадобятся», это соотношение может быть полезно при вычислении третьего случая. Вторая часть – «Если интервал событий содержит одну вечеринку, и нам необходимо снять костюм в конце этого интервала, то необходимо использовать ровно один костюм». Третья часть – «Если мы повторно используем костюм с первой вечеринки впервые на вечеринке с номером s $(a < s \leq b)$, то нам необходимо оптимально распределить костюмы на вечеринках с $a + 1$ до $s - 1$ (не снимая костюм с вечеринки с номером a). После этого на вечеринке с номером s у нас уже подходящий костюм! Если же мы не используем костюм с вечеринки с номером a повторно, то нам следует оптимально составить план переодеваний на вечеринках с номерами от $a + 1$ до b.»

Т.к. в соотношении для вычисления $F(a, b)$ длины интервалов всегда сокращаются, то это соотношение можно вычислить одним из двух способов: постепенно увеличивать длину отрезка $[a, b]$ либо воспользоваться рекурсивным подходом с мемоизацией.

Трудоемкость предложенного решения – $O(N^3)$.

Задача G: Ленивый поварёнок

Для того чтобы какой-то сок a можно было приготовить перед соком b, необходимо выполнение только свойства на компоненты сока.

Заметим, что перед приготовлением сока a нужно выбрать только подходящий сок $p(a)$, который можно приготовить перед ним. Отметим, что все значения $p(i)$ должны быть различны. Чем больше подходящих значений $p(a)$ мы сможем построить, тем меньше раз потребуется мыть чашу.

Описанные выше ограничения соответствуют решению задачи о наибольшем паросочетании. Для каждого сока следует сделать две вер-

шины: одну в левой доле графа, а вторую – в правой. Дугу (a, b) из левой доли проведем в вершину b в правой доле, если все компоненты сока a входят в сок b (т.е. сразу после приготовления сока a можно приготовить сок b без мытья чаши).

В построенном двудольном графе необходимо найти наибольшее паросочетание. Сделать это можно базовым алгоритмом построения дополняющих путей или более эффективным алгоритмом Хопкрофта-Карпа.

Задача H: Серии пенальти

Задача «на реализацию», не требующая знания каких-либо специальных алгоритмов. Единственное, что требуется от участников – умение аккуратно работать с двумерными массивами.

Храним результаты выполнения пенальти в двумерном массиве размеров $N \times N$. После чтения входных данных определяем количество очков P_i, добытых каждой командой, и сортируем команды по этому показателю. Затем проверяем значения P_Q и P_{Q+1}. Если они не равны, коллизия разрешена, и нам следует вывести номера первых Q команд.

В противном случае находим команды с одинаковым значением P_Q, проходя по массиву результатов вверх и вниз от индекса Q. Тем самым мы найдём требуемые в условии величины N_1 и Q_1. Номера команд, для которых будет проведена новая серия пенальти, будут находиться в просмотренном подинтервале массива результатов.

Задача I: Авиабилеты

Еще одна задача-упражнение на динамическое программирование.

В этом случае достаточно использовать стандартную технику «упаковки рюкзака»: необходимо продать не более N билетов за $W + 1$ недель, причем в каждую из недель можно выбрать только один из возможных вариантов цены на билеты.

Составим рекуррентное соотношение для величины $F(w, n)$ – максимальной прибыли за первые w недель продажи, если непроданными остаются n билетов:

$$F(w, n) = \max\{ F(w - 1, n - \min(s_i, n)) + \min(s_i, n) \cdot p_i\}.$$

Соотношение просто максимизирует прибыть от продажи билетов на неделе w. Второй параметр фиксирует количество оставшихся билетов, а выражение $\min(s_i, n)$ – количество проданных билетов на текущей неделе по цене p_i. Некоторым дополнительным упражнением является восстановление цен, по которым стоит продавать билеты на каждой из недель. Для решения этого упражнения можно запомнить, какая цена

была зафиксирована на соответствующей неделе, чтобы получить максимальное значение $F(w, n)$.

В завершение отметим, что оптимальное значение после продажи билетов на протяжении всех $W + 1$ недель не обязательно находится в значении $F(W + 1, 0)$, потому как может оказаться выгодным не продать все билеты. Оптимальное значение нужно искать среди всех $F(W + 1, n)$.

Задача J: Радиационная безопасность

Нужно признать, что это задание в большей степени на внимательное чтение условия. Процитируем очень важную часть условия: «… Естественно, нет смысла в этих пунктах устанавливать два одинаковых комплекта дозиметрического оборудования. …решено построить дополнительно столько пунктов контроля вне зон наблюдения, сколько пунктов попало одновременно в обе зоны наблюдения...»

Таким образом, нужно найти количество точек, которые попадают в круг определенного радиуса относительно двух заданных центров. Это очень просто сделать, отсортировав квадраты расстояний (в этом случае не потребуется вещественная арифметика, которая всегда вынуждает проделывать определенные манипуляции с окрестностями вещественных значений – так называемое «не забыть про эпсилон…»). Когда расстояния до пунктов от обеих АЭС отсортированы, то ответ на запрос делается двумя тривиальными двоичными поисками.

Таким образом, описанное решение имеет трудоемкость $O(NlogN + QlogN)$.

Задача K: Риск

Подход к решению этой задачи немного похож на решение задачи «Ленивый поварёнок». Нам необходимо определить, куда будем перемещать армии, чтобы выполнялись условия задачи.

Для начала заметим, что ответ на задачу можно искать методом «двоичного поиска по ответу». Это значит, что мы можем попробовать составить процедуру проверки гипотезы «А правда, что ответ не меньше X?», и метод проверки будет отвечать на подобный вопрос «Да/Нет». Действительно, если мы показали, что для какого-то значения X верно предположение гипотезы, то и для всех меньших значений гипотеза также верна, если для какого-то значения Y гипотеза неверна, но она не будет верна и для всех больших значений Y'. Здесь и далее под ответом нужно понимать количество армий в слабейшем граничном регионе игрока.

Теперь сформулируем принцип работы процедуры проверки для значения X. Для того, чтобы отслеживать перемещение, необходимо составить сеть, в которой пропускные способности дуг будут задавать возможность перемещения армий:

- для каждого региона создадим по две вершины в сети, первая будет задавать начальное состояние региона, а вторая - финальное состояние региона;
- из источника в каждую из вершин первого типа проведем дугу с пропускной способностью, равной количеству армий в данном регионе;
- между вершинами, соответствующими одному региону, проведем дугу пропускной способности «бесконечность» – фактически поток по этой дуге будет означать, что армии остались на месте;
- между вершиной первого типа одного региона и вершиной второго типа другого региона проведем дугу с пропускной способностью «бесконечность», если эти регионы являются соседними – фактически поток по этой дуге будет означать, что армия переместилась из региона в соседний;
- для каждого граничного региона необходимо создать дугу из второй его вершины в сток с пропускной способностью X – поток по этой дуге будет задавать требование на необходимость сделать все граничные регионы достаточно защищенными;
- для каждой региона, не являющегося граничным, из второй его вершины создадим дугу в сток с пропускной способностью 1 – фактически потом по этой дуге будет задавать дополнительное требование в задаче, что во всех регионах в после перемещения должна быть хотя бы одна армия игрока.

Для положительного ответа на поставленную гипотезу необходимо, чтобы после нахождения максимального потока в построенной сети все дуги, ведущие в сток, были полностью насыщенными.

Оставим небольшое упражнение читателю. Можно показать, что перемещение армий можно совершать в такой последовательности, что в любой момент времени в каждом из регионов игрока будет как минимум одна армия.

Дополнительно отметим, что, наверное, самым полезным и эффективным алгоритмом поиска максимального потока в сетях (в применимости к олимпиадам по программированию) авторы считают алгоритм Диница. Если он еще не изучен, то самое время почитать его описание и решить несколько задач с его помощью.

Задача L: Треугольники и четырёхугольник

Для решения этой задачи нужно определить, в какой именно четырехугольник могут сложиться два треугольника. Здесь возможны несколько случаев, показанных на рисунках 9 - 12:

- треугольник – две стороны совпадают, а две становятся продолжением одна другой;

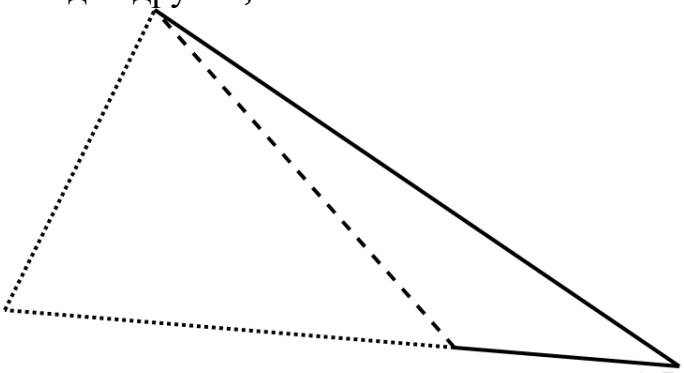

Рис. 9

- выпуклый четырехугольник – две стороны треугольников становятся диагональю;

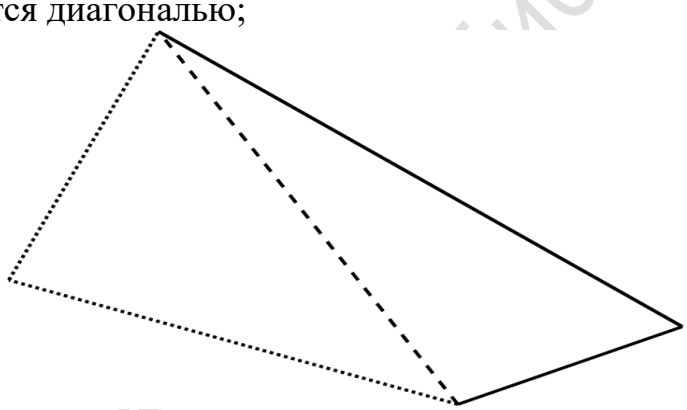

Рис. 10

- невыпуклый четырехугольник – две стороны треугольников становятся диагональю;

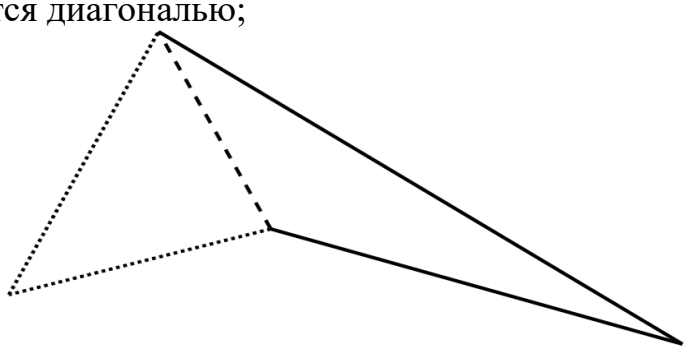

Рис. 11

- невыпуклый четырехугольник – две стороны треугольников становятся стороной четырехугольника, а другая пара сторон образует еще 2 стороны и диагональ.

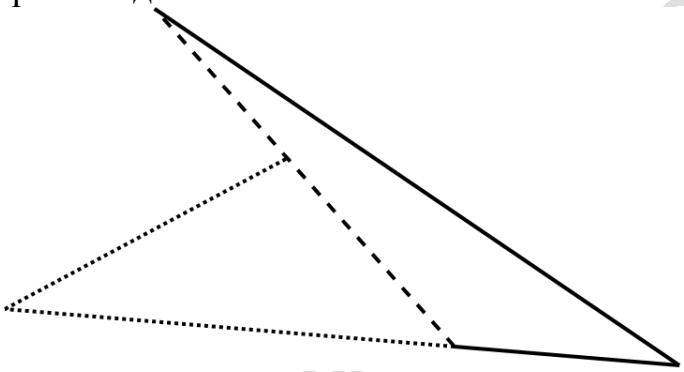

Рис. 12

Но даже после рассмотрения всех случаев расположения треугольников, формирующих четырехугольник, еще нужно очень аккуратно реализовать всю необходимую геометрию на плоскости. Это уже технические детали.

Авторы считают, что самой большой сложностью этой задачи является рассмотрение всех четырех способов расположения треугольников.

Содержание

ПРЕДИСЛОВИЕ ... 3

УСЛОВИЯ ЗАДАЧ ... 5

Седьмой чемпионат (2014 год) ... 5

Задача A: Лошадью ходи, лошадью... ... 5

Задача B: Ров вокруг замка ... 6

Задача C: Забор ... 7

Задача D: Вложение денег ... 9

Задача E: Попрыгунчик ... 10

Задача F: Таблетки ... 12

Задача G: Манхэттенский почтальон ... 13

Задача H: Последовательные суммы ... 15

Задача I: Лабиринт с телепортами ... 15

Задача J: Раскраска тетраэдров ... 16

Восьмой чемпионат (2015 год) ... 17

Задача A: Путь ядра ... 17

Задача B: Ожерелье из монет ... 19

Задача C: Набор номера ... 19

Задача D: Делимость Фибоначчи ... 21

Задача E: Делимость Фибоначчи (сложная версия) ... 22

Задача F: Хэллоуин ... 22

Задача G: Ленивый поварёнок ... 24

Задача H: Серии пенальти ... 25

Задача I: Авиабилеты ... 27

Задача J: Радиационная безопасность ... 29

Задача K: Риск ... 31

Задача L: Треугольники и четырёхугольник ... 32

РАЗБОР ЗАДАЧ 34
Седьмой чемпионат (2014 год) 34
Задача A: Лошадью ходи, лошадью... 35
Задача B: Ров вокруг замка 36
Задача C: Забор 37
Задача D: Вложение денег 37
Задача E: Попрыгунчик 38
Задача F: Таблетки 38
Задача G: Манхэттенский почтальон 39
Задача H: Последовательные суммы 40
Задача I: Лабиринт с телепортами 41
Задача J: Раскраска тетраэдров 42
Восьмой чемпионат (2015 год) 43
Задача A: Путь ядра 44
Задача B: Ожерелье из монет 45
Задача C: Набор номера 45
Задача D: Делимость Фибоначчи 46
Задача E: Делимость Фибоначчи (сложная версия) 46
Задача F: Хэллоуин 47
Задача G: Ленивый поварёнок 48
Задача H: Серии пенальти 49
Задача I: Авиабилеты 49
Задача J: Радиационная безопасность 50
Задача K: Риск 50
Задача L: Треугольники и четырёхугольник 52

Учебное издание

Кашкевич Сергей Иванович
Толстиков Алексей Александрович

СБОРНИК ОЛИМПИАДНЫХ ЗАДАЧ ПО ИНФОРМАТИКЕ

Практикум для студентов факультета прикладной математики и информатики

В пяти частях

Часть 2

В авторской редакции

Ответственный за выпуск *А. А. Толстиков*

Подписано в печать 04.12.2018. Формат 60×84/16. Бумага офсетная.
Усл. печ. л. 3.25. Уч.-изд. л. 2.9. 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.