РАЗБОР ЗАДАЧ
Четвертьфинальных соревнований Северного подрегиона
Северо-Восточного европейского региона
Командного студенческого чемпионата мира по программированию 2010-11

Задача A. Alien Communication Masterclass

В данной задаче требовалось вывести арифметическое равенство, верное в системах счисления с основаниями a_1, a_2, …, a_n и неверное в системах счисления с (отличными от a_i) основаниями b_1, b_2, …, b_m. Для составления равенства можно было использовать сложение, вычитание, умножение, целочисленные константы, а так же круглые скобки для группировки операций.

Рассмотрим случай, когда одно из a_i равно двум. Для того, что бы арифметическое выражение было корректно в двоичной системе счисления, оно должно использовать только цифры «0» и «1».

Рассмотрим числа, которые можно составить из этих цифр, например 0, 1, 10. Заметим, что у чисел 0 и 1 значения не зависят от основания системы счисления, в то время как число 10 в k-ичной системе счисления равно k. Таким образом, выражение вида

$$10\underbrace{-1-1-\cdots-1}_{k \text{ вычитаний}},$$

равно нулю системе счисления с основанием k и только в ней. Таким образом, равенство вида

$$10\underbrace{-1-1-\cdots-1}_{k \text{ вычитаний}}=0,$$

является решением задачи для $n = 1$. (10−1−1) × (10−1−1−1) = 0

Для решения задачи в целом, заметим, что произведение нескольких чисел равно нулю тогда, и только тогда, когда, по меньшей мере одно из них равно нулю. Таким образом, мы можем скомбинировать решения для систем счисления с различными основаниями, перемножив их. Итоговый ответ имеет вид

$$(10\underbrace{-1-1-\cdots-1}_{a_1 \text{ вычитаний}})\times(10\underbrace{-1-1-\cdots-1}_{a_2 \text{ вычитаний}})\times\cdots\times(10\underbrace{-1-1-\cdots-1}_{a_n \text{ вычитаний}})=0,$$

где перемножаются n скобок, в первой из которых единица вычитается a_1 раз, во второй — a_2 раз, и так далее.

Отметим, что данное построенное равенство верно только в системах счисления с основаниями a_1, a_2, …, a_n, и, следовательно, неверно в системах счисления с основаниями b_1, b_2, …, b_m.

Приведенное решение имеет сложность по памяти и по времени $O(tn^2)$, где t — максимальное основание системы счисления.

Задача B. Bug2

В данной задаче требовалось определить длину пути, проходимого роботом, реализующим алгоритм Bug2. Этот алгоритм предназначен для нахождения на плоскости пути из точки S в точку F при наличии препятствий. Алгоритм Bug2 формулируется следующим образом:

1. Робот двигается по прямой в направлении точки F пока не будет выполнено одно из следующих условий:
   - Достигнута точка F. В этом случае алгоритм успешно завершается.
   - Достигнуто некоторое препятствие. В этом случае осуществляется переход ко второму шагу алгоритма.
2. Обозначим текущую точку через H. Робот двигается вдоль границы препятствия, обходя его в направлении по часовой стрелке. Движения осуществляется до наступления одного из следующих событий:
   - Достигнута точка F. В этом случае алгоритм успешно завершается.

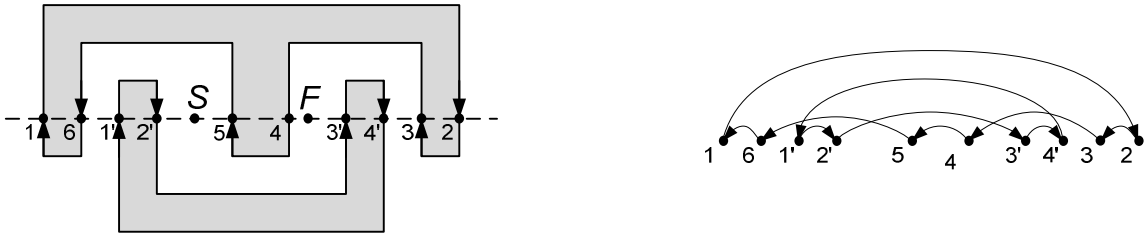
- Достигнута точка *H*. В этом случае алгоритм завершается с ошибкой «Точка *F* недостижима».
- Достигнута такая точка *L*, лежащая на прямой *ST*, что расстояние между точками *L* и *F* меньше чем расстояние между точками *H* и *F*, и, кроме того, возможно движение по прямой в направлении точки *F*. В этом случае осуществляется переход к первому шагу алгоритма.

В задаче гарантировалось, что точка *F* достижима из точки *S*.

Решение этой задачи состоит из двух частей. Вначале требуется построить граф точек, лежащих на прямой *SF* и границе какого-либо препятствия. Во второй части необходимо подсчитать длину пути, проходимого роботом.

Найдем все точки пересечения границ препятствий с прямой *SF*. Отметим, что это проще, чем нахождением точек пересечения с отрезком *SF*, а появляющиеся при этом точки пересечения вне отрезка, хоть и не интересны, но и не мешают последующему решению.

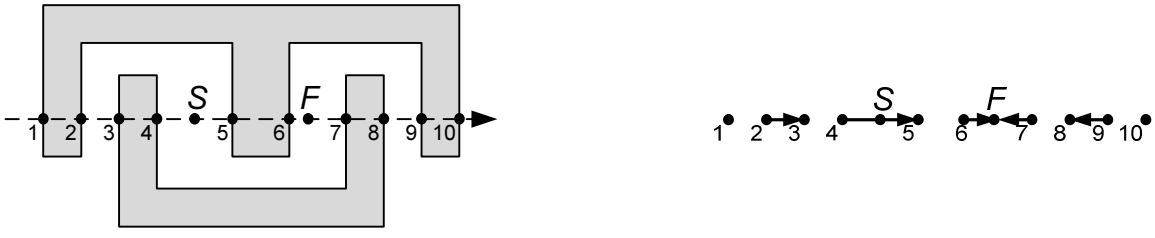
Будем искать точки пересечения в порядке обхода каждого препятствия по часовой стрелке. Например, на рис. 1а цифрами помечены точки пересечения, найденные при обходе препятствий. В процессе обхода будем подсчитывать расстояние между соседними с точки зрения обхода точками пересечения. Таким образом, получим граф точек пересечения и «поверхностных» переходов между ними (рис. 1б).



а б

Рис 1. Точки пересечения в порядке обхода препятствий

Рассмотрим найденные точки пересечения в порядке их расположения на прямой *SF*. За положительное направление примем направление вектора *SF* (рис. 2а). Отметим, что точки между точкой с чётным номером и следующей за ней всегда можно пройти на прямую. При этом такой «сокращенный» проход будет всегда направлен к точке *F*. Для интервала, содержащего точку *F* вместо одного «сокращенного» прохода добавим два, непосредственно ведущих в *F*. Для точки S так же добавим переход, ведущий в ближайшую большую точку (она будет иметь нечетный номер). В итоге получим граф «сокращенных» проходов, приведенный на рис. 2б.



а б

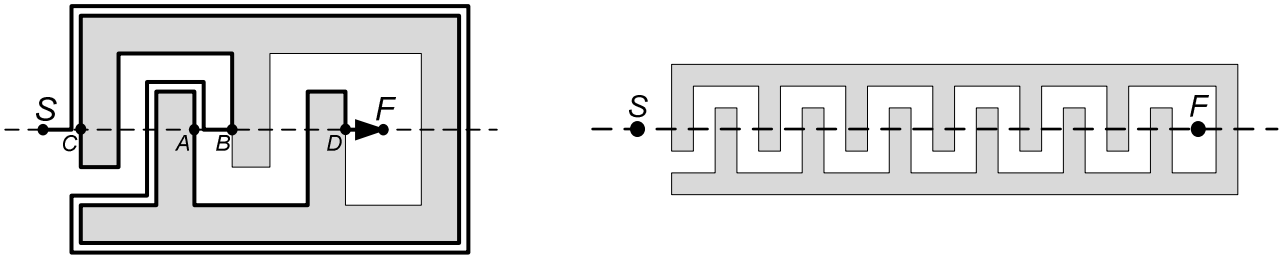
Рис. 2. Точки пересечения в порядке от *S* к *F*

Рассмотрим работу алгоритма с применением двух построенных графов. Шаг один заключается в том, что мы используем «сокращенный» переход, ведущий из текущей точки. Шаг два сводится к перемещению в графе «поверхностных» переходов пока не станет возможным «сокращенный» переход. Таким образом, решение задачи свелось к нахождению пути с применением построенных графов.

Отметим, что состояние алгоритма на втором шаге зависит не только от текущей точки, но и от расстояния *HF*. Таким образом, возможно, что в процессе работы алгоритма проход по

одному участку будет осуществляться несколько раз, например, как схематично показано на рис. 3а. При этом в первый раз используется «сокращенный» переход из точки A в точку B, так как $|AF| < |CF|$. Во второй же раз этот переход не производится, так как $|AF| > |BF|$.

Увеличивая число зигзагов, как показано на рис. 3б. можно получить путь, состоящий из $O(n^2)$ переходов, где n — число точек пересечения, которое может достигать суммарного числа отрезков в препятствиях. Таким образом, непосредственный подсчет пути в графе недостаточно эффективен.



а б

Рис. 3. Многократное прохождение по стороне препятствия

Для решения этой проблемы отметим, что ни один «сокращенный» переход не может быть использован дважды, а всего таких переходов $O(n^2)$. Если бы могли эффективно перемещаться между точками «сокращенного» перехода, то это позволило бы найти решение за линейное время. Для этого будем «сжимать» пути между соседними точками, в которых использован «сокращенный переход», то есть считать, что он состоит из одного «сжатого» перехода, а не нескольких «поверхностных». При этом длина «сжатого» перехода будет равна сумму длин сжатых переходов. Далее разрешим аналогично сжимать и другие «сжатые» переходы.

В примере, приведенном на рис. 3а. последовательно будут сжаты путь между парами точек: C и A, B и C, далее использован «сжатый» переход от точки C к точке A, который войдет в «сжатый» переход из точки B в точку D.

Отметим, что в процессе сжатия путей мы заменяем каждую последовательность путей на один путь, при этом далее используются только «сжатые» переходы. При этом каждый «поверхностный» и «сокращенный» переходы будут использованы не более двух раз (один раз при движении «по поверхности» и один раз по «сокращенному» переходу). Таким образом, сжатие путей позволяет найти искомую длину пути в графе за $O(m)$.

Построенное решение имеет сложность по времени $O(m \log m)$ и $O(m)$ по памяти, где m — суммарное число отрезков, ограничивающих все препятствия. Логарифм в оценке времени получается за счёт сортировки точек пересечения для построения графа «сокращенных» переходов.

Задача C. Commuting Functions

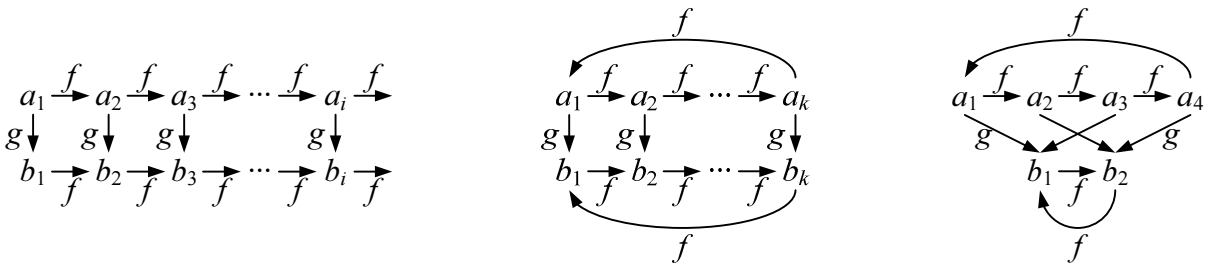
В данной задаче требовалось по заданной биекции f из X в X ($X = \{1, 2, \ldots, n\}$) построить функцию g, коммутирующую с функцией f, то есть для любого x из X выполняется: $f(g(x)) = g(f(x))$. Среди таких функций требовалось выбрать ту, для которой список $[f(1), f(2), \ldots, f(n)]$ минимален в лексикографическом порядке.

Возьмем некоторый x из X и рассмотрим последовательность $a_1 = x$, $a_2 = f(a_1)$, $a_3 = f(a_2)$, …. Эту последовательность можно продолжать бесконечно, а так как число элементов в множестве X конечно, то данная последовательность будет с какого-то места периодической.

Пусть $a_1, a_1, \ldots, a_p$ — предпериод, $a_{p+1}, a_{p+2}, \ldots, a_{p+k}$ — период и при этом $a_p \neq a_{p+k}$. По свойству периода $a_{p+k+1} = a_{p+1}$, то есть $f(a_{p+k}) = a_{p+1} = f(a_p)$. Так как функция f является биекцией, то существует единственный элемент y из X, такой что $f(y) = a_{p+1}$, следовательно, $a_p = y = a_{p+k}$. Полученное противоречие не возникает только при $p = 0$. Следовательно, последовательность a_i не содержит предпериода.

Таким образом, посредством последовательного применения функции f множество X разбивается на набор циклов вида a_1, $a_2 = f(a_1)$, $a_3 = f(f(a_2))$, …, $a_k = f^{k-1}(a)$, где k — длина цикла.

Рассмотрим последовательность $b_i = g(a_i) = g(f^{i-1}(a_1))$, так как g коммутирует с f то $b_i = f^{i-1}(g(a_1))$. Таким образом, b_i с одной стороны получается однократным применением функции g к a_i, а с другой стороны, $(i - 1)$-кратным применением функции f к b_1 (рис. 1а).



А б в

Рис 1. Последовательности a_i и b_i

Так как k — длина цикла, то $a_{k+1} = a_1$, следовательно, $b_{k+1} = g(a_{k+1}) = g(a_1) = b_1$, то есть последовательность b_i так же имеет период k (рис. 2б). Отметим, что этого не следует, что k — минимальная длина цикла в последовательности b_i, эта последовательность может иметь цикл и меньшей длины l, если k кратно l (рис. 2в).

Таким образом, решение задачи свелось к отображению циклов, порождаемых функцией f в другие циклы этой же функции. При этом длина отображаемого цикла должна быть кратной длине того цикла, в которой производится отображение.

Теперь можно описать решение:

1. Вначале множество неотображенных вершин $A = X$.
2. Возьмем $a = \min A$ — минимальный элемент множества A.
3. Найдем соответствующий ему цикл и его длину k.
4. Найдем циклы, длина которых кратна k и среди всех таких циклов выберем минимальный элемент b.
5. Построим последовательности a_i и b_i, начиная с a и b соответственно, и положим $g(a_i) = f(b_i)$.
6. Удалим из A все a_i.
7. Если A не пусто, перейти к шагу 2.

Отметим, каждый шаг алгоритма исполняется не более n раз. Покажем, что суммарное число действий на каждом шаге так же можно сделать $O(n)$. Для этого будем хранить характеристический вектор множества A и n чисел m_i — минимальный возможный элемент цикла длиной i.

На первом шаге помечаем, что в A содержатся все элементы и все m_i не определены. Это требует $O(n)$ действий.

На втором шаге будем хранить j — номер минимального нерассмотренного элемента, тогда можно будет не рассматривать элементы меньшие j, что позволит найти все минимумы за суммарно линейное время.

На третьем шаге мы рассмотрим каждый цикл не более одного раза. Суммарная длина всех циклов n, следовательно, на это потребуется $O(n)$ времени.

На четвертом шаге, если m_k не определено, то положим $m_k = a$. Далее, рассмотрим все возможные длины циклов, меньшие либо равные k, среди них выберем те, которые делят k, после чего найдем минимальный элемент с использованием m_i. Таким образом, будет выполнено $O(k)$ действий на каждый цикл и $O(n)$ в сумме.

Шаги пять и шесть так же требуют $O(k)$ действий на каждый цикл и $O(n)$ в сумме.

Таким образом, построенное решение имеет сложность $O(n)$ по времени и по памяти.

Задача D. Defense of a Kingdom

В данной задаче требовалось найти максимальный по площади прямоугольник, клетки которого не защищен арбалетными башнями. Каждая арбалетная башня защищает все клетки в одной с ней горизонтали и вертикали.

Дополним заданную расстановку башен дополнительными башнями с координатами (0, 0) и ($w + 1$, $h + 1$), где w и h — ширина и высота королевства. Это позволит искать требуемый прямоугольник только в промежутках между башнями.

Отсортируем башни по возрастанию координаты x и рассмотрим разности $c_i = x_i - x_{i-1}$ — расстояния между соседними башнями по горизонтали. Затем отсортируем башни по возрастания координаты y и найдем $r_i = y_i - y_{i-1}$ — расстояния между соседними башнями по вертикали.

Пусть искомый прямоугольник имеет размеры $a \times b$ клеток, тогда $a \le \max c_i$, $r \le \max r_i$, но в королевстве есть прямоугольник $\max c_i \times \max r_i$, таким образом, он и является ответом.

За счет сортировки предложенное решение имеет сложность $O(n \log n)$ по времени и $O(n)$ по памяти.

Задача E. Explicit Formula

В данной задаче требовалось подсчитать значение булевой функции f от десяти переменных. При этом сообщалось, что эта функция определяет четность числа пар и троек аргументов, среди которых как минимум один истинен.

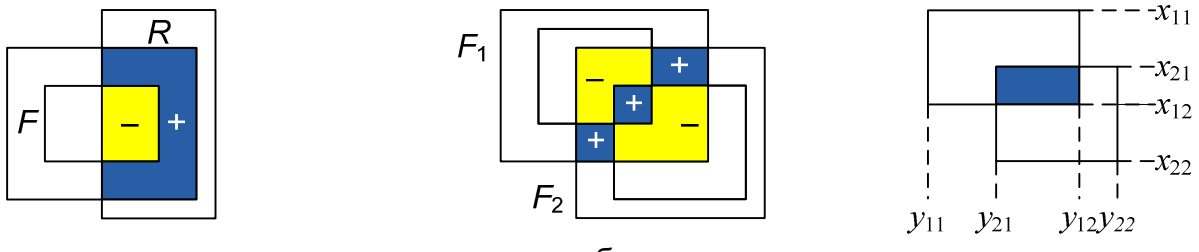
Данную задачу можно было решать большим числом различных способов:

1. Вбить формулу, приведенную в условии задачи в решение. Та как формула довольно длинна, то это занимает немало времени, при большой вероятности ошибки.
2. Сгенерировать формулу, приведенную в условии задачи и вставить ее в решение. Так как формула имеет несложную структуру, то это легко сделать.
3. Проэмулировать вычисления по формуле, перебирая пары и тройки переменных вложенными циклами.
4. Заметить, что формула все переменные входят в формулу симметрично. Следовательно, ответ зависит только от того, сколько средин них равно 0. Далее ответ модно подсчитать по формуле $(q(10) - q(z)) \bmod 2$, где $q(n) = n^2 / 2 + n^3 / 6$ — число способов выбрать пары и тройки из n элементов, z — число переменных, равных нулю.
5. В явной форме посчитать ответ для всех возможных вариантов числа нулевых входных переменных (от 0 до 10) и выявить, что ответ «0» только при числе нулей равном 2, 6 или 10.

Задача F. Frames

В данной задаче требовалось найти такое положения двух прямоугольных рамок, при котором площадь их пересечения максимальна.

Заметим, что площадь пересечения рамки F и прямоугольника R можно определить, как разность площадей пересечения внешнего (F_o) и внутреннего (F_i) прямоугольников рамки с прямоугольником R. (рис 1а.). Это можно записать как $F \cap R = F_o \cap R - F_i \cap R$.



а б в

Рис 1. Пересечения

Представив вторую рамку как разность внутреннего и внешнего прямоугольников, получим: $F_1 \cap F_2 = F_1 \cap F_{2o} - F_1 \cap F_{2i} = F_{1o} \cap F_{2o} - F_{1i} \cap F_{2o} - F_{1o} \cap F_{2i} + F_{1i} \cap F_{2i}$. (рис. 1б). Таким образом, задача подсчета площади пересечения рамок свелась к задаче подсчета площадь пересечения прямоугольников.

Пусть даны два прямоугольника $R_1(x_{11}, y_{11}, x_{12}, y_{12})$ и $R_1(x_{21}, y_{21}, x_{22}, y_{22})$, где $x_{i1} < x_{i2}$ и $y_{i1} < y_{i2}$. Точки (x, y), входящие в пересечение прямоугольников R_1 и R_2 удовлетворяют

следующим неравенствам: $\max(x_{11}, x_{21}) \le x \le \min(x_{12}, x_{22})$ и $\max(y_{11}, y_{21}) \le y \le \min(y_{12}, y_{22})$ (рис. 1в). Таким образом, если $\max(x_{11}, x_{21}) < \min(x_{12}, x_{22})$ и $\max(y_{11}, y_{21}) < \min(y_{12}, y_{22})$, то площадь пересечения прямоугольников R_1 и R_2 равна

$$|R_1 \cap R_2| = (\min(x_{12}, x_{22}) - \max(x_{11}, x_{21})) \times (\min(y_{12}, y_{22}) - \max(y_{11}, y_{21})).$$

Рассмотрим сдвиг прямоугольника R_2 относительно R_1 вдоль оси *Ox* на расстояние *d*, таким образом, что в процессе ни какая пара координат из $x_{11}, x_{21}, x_{12}, x_{22}$ не совпадает («хороший» сдвиг). В этом случае второй множитель не изменяется, а первый либо не изменяется, либо изменяется на *d*. В обоих случаях, изменение площади пересечения будет пропорционально *d*. Аналогично, при «хорошем» сдвиге второго прямоугольника вдоль оси *Oy* изменение площади пересечения будет пропорционально сдвигу.

Рассмотрим некоторое положение двух рамок и небольшой сдвиг второй рамки вдоль оси *Ox*, такой, что порядок координат, вершин прямоугольников, определяющих рамки сохранится (то есть, в процессе сдвига ни какие две прямые, содержащие стороны прямоугольников не совпадают). Выразив площадь пересечения рамок через площади пересечения прямоугольников, получим, что при таком сдвиге изменение площади в случае рамок также пропорционально сдвигу (как сумма величин, каждая из которых пропорциональна сдвигу). Таким образом, при «хорошем» сдвиге рамки площадь зависит линейно от координаты. Так как у линейной функции нет экстремумов, то выгодно сдвигать рамку в каком-то направлении, пока две стороны прямоугольников, определяющих рамки не совпадут.

Аналогичное рассуждение, примененное к сдвигам по оси *Oy*,позволяет сделать вывод, что максимальная площадь пересечения рамок достигается при одном из сдвигов, для которых стороны некоторых пар прямоугольников (возможно разных, для разных осей), задающих рамки совпадут как по *x*, так и по *y*.

Таким образом, для нахождения ответа достаточно перебрать все сдвиги, при которых какие-то пары координат рамок совпадают (как по *x*, так и по *y*) и найти среди них сдвиг, дающий максимальную площадь.

Приведенное решение не зависит от характеристик рамок и имеет сложность O(1) как по времени, так и по памяти.

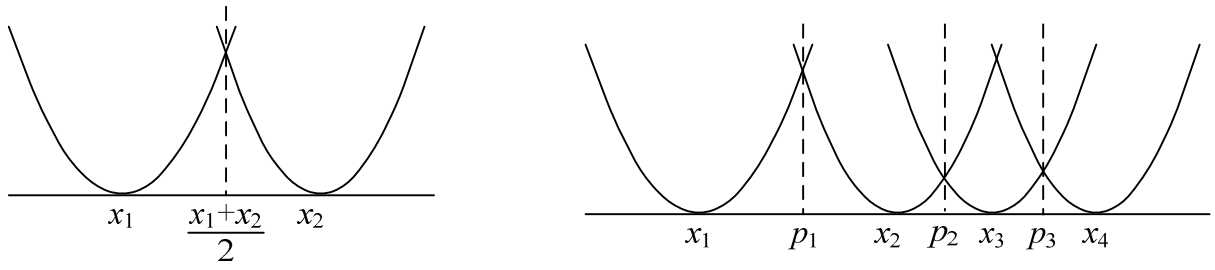
Задача G. Gadgets Factory

В данной задаче требовалось найти такое местоположение фабрики, чтобы сумма квадратов расстояний до фабрик, производящие необходимые детали была минимальна. При этом все фабрики расположены на координатной оси *Ox*, и может быть более одной фабрики производящей некоторую деталь.

Вначале решим задачу в случае, если каждую необходимую деталь производит ровно одна фабрика. Пусть эти фабрики имеют координаты $x_1, x_2, \ldots, x_n$, где n — число необходимых деталей. Тогда сумма квадратов расстояний от них до точки x будет равна $\sum(x - x_i)^2$, здесь и далее суммирование производится по i от 1 до n. Для нахождения оптимального x продифференцируем сумму и приравняем ее нулю: $(\sum(x - x_i)^2)' = 2\sum(x - x_i) = 0$. Раскрыв скобки и решив полученное уравнение, получим $x = \sum x_i / n$. Отметим, что сумма квадратов расстояний при оптимальном x будет равна

$$\sum(x - x_i)^2 = \sum(x^2 - 2xx_i + x_i^2) = nx^2 - 2x\sum x_i + \sum x_i^2 =$$
$$= (\sum x_i)^2 / n - 2(\sum x_i)^2 / n + \sum x_i^2 = \sum x_i^2 - (\sum x_i)^2 / n$$

Перейдем к рассмотрению случая, когда несколько фабрик производят деталь. Пусть некоторая деталь производится на заводах в точках x_1 и x_2 ($x_1 < x_2$) , в таком случае, если новая фабрика будет построена в точке $x < (x_1 + x_2) / 2$, то выгодно использовать в качестве поставщика завод в точке x_1, а иначе — завод в точке x_2 (рис. 1а). Аналогично, если фабрик больше двух, то в диапазоне $(-\infty, (x_1 + x_2) / 2)$ выгодно использовать завод в точке x_1, в диапазоне $((x_2 + x_3) / 2, (x_1 + x_2) / 2)$ —завод в точке x_2 и так далее. Таким образом, k заводов, производящих один товар разбивают плоскость на k полос $k - 1$ граничной точкой вида $p_i = (x_{i+1} + x_i) / 2$ (рис. 1б).



а б

Рис 1. Полосы

Рассмотрим оптимальное решение. Оно попадает в одну из таких полос для каждого типа детали. Таким образом, рассмотрев все варианты пересечения полос можно найти правильный ответ. Для этого отсортируем все граничные точки в порядке возрастания их координаты и будем рассматривать полосы в этом порядке. При этом потребуется рассмотреть O(m) полос, где m — суммарное число всех фабрик.

Вычислим для каждой полосы оптимизируемую сумму $S = \sum x_i^2 - (\sum x_i)^2 / n$ (где в качестве x_i берется координата соответствующей фабрика i-го типа). Выбрав среди таких сумм минимальную, получим искомую полосу и соответствующее оптимальное расположение фабрики. Данное решение требует вычисления сумм из n слагаемых для каждой полосы, и, таким образом, имеет сложность по времени O(nm), что недостаточно эффективно.

Для повышения эффективности пересчета оптимизируемой суммы представим её в виде $S = R - T^2 / n$, где $R = \sum x_i^2$, $T = \sum x_i$. Рассмотрим переход в соседнюю полосу, при котором положение ближайшей фабрики, производящей i-й ресурс меняется с x_i на x_i', тогда мы можем посчитать новые значения $R' = R - x_i^2 + x_i'^2$ и $T' = T - x_i + x_i'$. Таким образом. Новое значение оптимизируемой функции может быть посчитано за константное время, что дает рассмотрение всех полос за O(m). Начальные значения R и T для самой левой полосы считаются непосредственно.

Отметим, что, во-первых, оптимальным решением для текущей полосы является $x = T / n$, а во-вторых, умножив оптимизируемую функцию на n можно считать её в целых числах.

Предложенное решение имеет сложность O($m \log m$) по времени и O(m) по памяти, где логарифм появляется из-за сортировки граничных точек полос.

Задача H. Horrible Truth

В данной задаче требовалось составить как можно более длинный сценарий телесериала, в рамках которого n персонажей узнают ужасную правду. В сериале допустимы серии трех типов:

1. Персонаж A узнает правду.
2. Персонаж A узнает, что персонаж B знает правду.
3. Персонаж A узнает, что персонаж B не знает правду.

При этом в сценарии две серии одного типа не могут идти подряд.

Подчитаем теоретически возможное число разных серий. Серий первого типа может быть по одной на каждого персонажа. Серий второго и третьего типов — по одной на каждую упорядоченную пару персонажей. Таким образом, общее число различных серий равно $n + 2n (n - 1)$.

Не уменьшая общности, положим, что персонажи будут узнавать праву в порядке своих номеров.

Рассмотрим, какие серии могут идти до первой серии первого типа. Это не могут быть серии второго типа, так как ни один персонаж еще не знает правды. Следовательно — это серии третьего типа, причем не более одной. Аналогично после последней серии первого типа может быть не более одной серии, причем второго типа. Так как только один персонаж сможет узнать, что первый персонаж не знал правды, и только один персонаж может узнать, что последний персонаж узнал правду, то по сравнению с теоретическим подсчетом мы потеряли два раза по $(n - 2)$ серии.

Будем составлять сценарий так, чтобы до того, как i-й персонаж (кроме первого) узнал истину, все остальные персонаж узнали, что он не знает истины. И все персонаж знали про $(i - 1)$-го персонажа, что он знает истину. Для этого потребуются по $(n - 2)$ события второго и третьего типов, которые можно чередовать следующим образом:

- 1-й персонаж узнает, что i-й персонаж не знает правды;
- 1-й персонаж узнает, что $(i - 1)$-й персонаж знает правду;
- 2-й персонаж узнает, что i-й персонаж не знает правды;
- 2-й персонаж узнает, что $(i - 1)$-й персонаж знает правду;
- …
- $(i - 2)$-й персонаж узнает, что i-й персонаж не знает правды;
- $(i - 2)$-й персонаж узнает, что $(i - 1)$-й персонаж знает правду;
- $(i - 1)$-й персонаж узнает, что i-й персонаж не знает правды;
- i-й персонаж узнает, что $(i - 1)$-й персонаж знает правду;
- $(i + 1)$-й персонаж узнает, что i-й персонаж не знает правды;
- $(i + 2)$-й персонаж узнает, что $(i - 1)$-й персонаж знает правду;
- …
- n-й персонаж узнает, что i-й персонаж не знает правды;
- n-й персонаж узнает, что $(i - 1)$-й персонаж знает правду;

В таком блоке содержатся $(2n - 2)$ серии — по две на каждого персонажа, кроме случаев, когда персонаж должен был бы узнать сам о себе.

Итого мы можем построить сценарий из начальной серии, заключительной серии, n серий первого типа и $(n - 1)$ блока по $(2n - 2)$ серий второго и третьего типов. В сумме получаем $2 + n + (n - 1) \times (2n - 2) = 2 + n + 2n(n - 1) - 2(n - 1)$ серий, что отличается от теоретической оценки на $2(n - 2)$ серии, «потерянные» до первого и после последнего события первого типа. Таким образом, полученный сценарий содержим максимально возможное число серий.

Предложенное решение имеет сложность $O(n^2)$ по времени и $O(1)$ по памяти.

Задача I. Ideal Contest

В этой задаче требовалось по таблице результатов соревнования определить его характеристики:

- «Тщетность»: (число команд, не решивших ни одной задачи) / T.
- «Переупрощение»: (число команд решивших все задачи) / T.
- «Равномерность»: (число «пропусков» в списке числа решенных задач) / P.
- «Нерешаемость»: (число задач, которые не решила ни одна команда) / P.
- «Нестабильность» задачи i: (число команд, не решивших задачу i, предшествующих последней команде решившей задачу i) / T.

Здесь T — число команд, а P — число задач.

Проще всего было решать эту задачу следующим образом. Вначале найти строку, состоящую только из знаков «-». Над ней будет находиться заголовок таблицы, а под ней — тело таблицы. Посчитав количество слов в заголовке таблицы и вычтя из него четыре («`Team`», «=», «`Time`», «`Rank`») найдем P.

Строки таблицы проще всего разбирать справа налево. Тогда последние три столбца соответствуют «=», «`Time`», «`Rank`» , а последующие P столбцов содержат информацию о решенных задачах. При этом задача решена тогда и только тогда, когда соответствующая ячейка содержит символ «+».

После разбора таблицы результатов легко подсчитать требуемые характеристики.

Задача J. Journey

В данной задаче требовалось найти максимальную длину пути, по которой Иван Сусанин может отправить армию Речи Посполитой, если армия использует две отдельных карты для дневных и ночных переходов. При этом расстояние до цели после перехода по соответствующей карте должно уменьшаться.

Для того чтобы определить, какие переходы разрешены требуется найти кратчайшие расстояния от всех деревень до цели отдельно для дневной и ночной карт. Это можно сделать алгоритмом Дейкстры. После этого Иван Сусанин может порекомендовать переход из деревни a в деревню b тогда и только тогда, когда на соответствующей карте $d_a > d_b$, где d_i — найденные минимальные расстояния.

Для нахождения ответа «раздвоим» каждую деревню на «ночную» и «дневную» и разрешим совершать переход из «ночной» деревни a в «дневную» деревню b, если он разрешен по ночной карте. Аналогично поступим с дневной картой и переходами из «дневных» деревень в «ночные». Если в построенном графе содержится цикл, достижимый из исходной деревни, но Иван Сусанин может водить армию бесконечно долго. Если такого цикла нет, то достижимая часть графа является направленным ациклическим графом, в котором длинна самого длинного пути может быть посчитана при помощи динамического программирования на топологически отсортированном графе. При этом надо учитывать, что армия должна посылаться по самой длинной из возможных дорог между деревнями.

При реализации алгоритма за $O(n^2)$ предложенное решение будет иметь сложность $O(n^2)$ по времени и по памяти.

Задача K. Kitchen Robot

В данной задаче требовалось найти длину минимального пути работа в прямоугольнике, который позволит вынести ему из прямоугольника все бутылки.

Построим граф, вершинами которого будут начальная точка, точки, в которых находятся бутылки и край стола. Из начальной точки ребра напрямую идут в точки с бутылками. К краю стола ведут ребра от каждой бутылки до ближайшего края, что соответствует обработке последней бутылки.

Между парой бутылок робот должен побывать на крае стола. Пусть нам надо найти кратчайший путь, проходящий через край стола между точками A и B (рис. 1а) Отразим стол относительно его сторон, построив четыре варианта пути (рис. 1б), которые при «сворачивании» обратно (рис. 1в) дадут оптимальные пути до соответствующих сторон. Таким образом, среди них достаточно выбрать кратчайший.

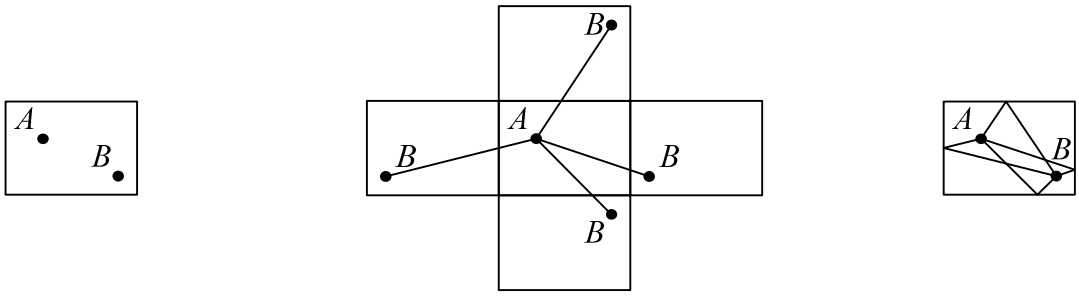


Рис 1. Перемещение между бутылками

После построения графа в нём необходимо найти кратчайший гамильтонов (посещающий каждую вершину) путь, из начальной точки к краю стола. Таким образом. Мы свели исходную задачу к задаче коммивояжёра.

Задачу коммивояжера при заданных ограничениях можно решать методом динамического программирования на подмножествах. Будем для каждой пары (текущая вершина, множество еще непосещёных вершин) хранить длину кратчайшего пути, посещающего все эти вершины. Тогда каждый раз у нас есть выбор, взять одну из еще непосещённых вершин за, следующую, что позволяет организовать подсчет по увеличению размера множества непосещённых вершин.

В качестве баз динамического программирование нужно использовать сведения о том, что если не посещена ровна одна бутылка, то после ее посещения нужно напрямую идти к краю стола.

Предложенное решение имеет сложность по времени и по памяти $O(2^n)$, где n — число бутылок.

Список литературы

1. Фихтенгольц Г. М. Курс дифференциального и интегрального исчисления. Том 1. М.: Физматлит, 2001, 616 с.
2. Кормен Т. Х. , Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ. Первое издание, раздел 22.2. М.: МЦНМО. 2001. 960 с.