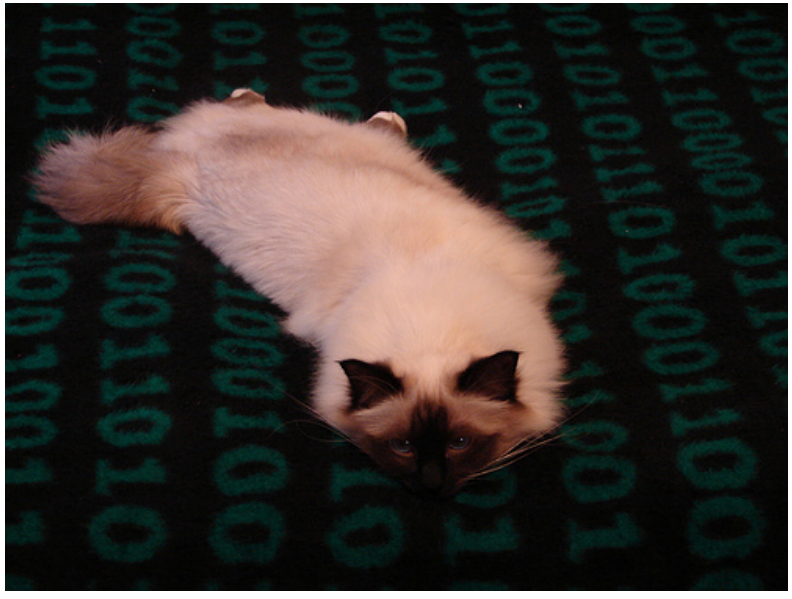
XIV командная олимпиада школьников по программированию

24 ноября 2013 года

Задача A. «Олег и двоичные последовательности»

Задача A. «Олег и двоичные последовательности»



- Идея задачи — Андрей Станкевич
- Подготовка тестов — Никита Иоффе
- Разбор задачи — Виталик Аксёнов

Постановка задачи

Дана Z-функция с пробелами для некоторой строки, состоящей из нулей и единиц.
Нужно было посчитать количество строк, удовлетворяющих данной функции.

Построение графа

- Для каждой пары позиций будем проводить ребро веса 1, если мы знаем, что символы в этих позициях различаются, и ребро веса 0, если мы знаем, что символы в этих позициях совпадают.
- Для нахождения данной информации посмотрим на Z-функцию.
- Рассмотрим символ на позиции i. Известно, что для любого $0 \leq l < Z[i]$: $s[i + l] = s[1 + l]$. А также, если $i + Z[i] < n$, то $s[i + Z[i]] \neq s[1 + Z[i]]$.
- Нужно сразу обработать условия невозможности: значение Z-функции в первом символе, и для любого $2 \leq i \leq n$: $i + Z[i] \leq n + 1$.

Построение графа

- Для каждой пары позиций будем проводить ребро веса 1, если мы знаем, что символы в этих позициях различаются, и ребро веса 0, если мы знаем, что символы в этих позициях совпадают.
- Для нахождения данной информации посмотрим на Z-функцию.
- Рассмотрим символ на позиции i. Известно, что для любого $0 \leq l < Z[i]$: $s[i + l] = s[1 + l]$. А также, если $i + Z[i] < n$, то $s[i + Z[i]] \neq s[1 + Z[i]]$.
- Нужно сразу обработать условия невозможности: значение Z-функции в первом символе, и для любого $2 \leq i \leq n$: $i + Z[i] \leq n + 1$.

Построение графа

- Для каждой пары позиций будем проводить ребро веса 1, если мы знаем, что символы в этих позициях различаются, и ребро веса 0, если мы знаем, что символы в этих позициях совпадают.
- Для нахождения данной информации посмотрим на Z-функцию.
- Рассмотрим символ на позиции i. Известно, что для любого $0 \leq l < Z[i]$: $s[i + l] = s[1 + l]$. А также, если $i + Z[i] < n$, то $s[i + Z[i]] \neq s[1 + Z[i]]$.
- Нужно сразу обработать условия невозможности: значение Z-функции в первом символе, и для любого $2 \leq i \leq n$: $i + Z[i] \leq n + 1$.

Подсчёт ответа

- Запустим *dfs* для каждой непосещённой ещё позиции.
- Пусть количество запусков *dfs* равно k, то ответ будет равен 2^k.
- Научимся отличать неверные данные. Это можно определять тем же *dfs*, что и для подсчёта ответа.
- Для этого будем считать от вершины запуска чётность расстояния до остальных вершин.
- Если мы в *dfs* идём в вершину не с той чётностью, которую мы предполагаем, то значит, данные не верны.

Подсчёт ответа

- Запустим *dfs* для каждой непосещённой ещё позиции.
- Пусть количество запусков *dfs* равно k, то ответ будет равен 2^k.
- Научимся отличать неверные данные. Это можно определять тем же *dfs*, что и для подсчёта ответа.
- Для этого будем считать от вершины запуска чётность расстояния до остальных вершин.
- Если мы в *dfs* идём в вершину не с той чётностью, которую мы предполагаем, то значит, данные не верны.

Задача B. «Город Че»

Задача B. «Город Че»

- Идея задачи — Елена Андреева
- Подготовка тестов — Демид Кучеренко
- Разбор задачи — Демид Кучеренко

Постановка задачи

Дан отсортированный массив a_i из n элементов. Необходимо было найти количество пар элементов массива, разность между которыми больше r.

Интуитивное решение

- Для каждого элемента *left* будем находить границу *right* такую, что $a[right] - a[left] \leqslant r$, а $a[right + 1] - a[left] > r$.
- Тогда для элемента *left* все элементы с $rigth + 1$ по n удалены на расстояние большее, чем r.
- Добавим к ответу $n - right$ и перейдем к следующему элементу (сдвинем *left*).
- Такое решение работает за $O(n^2)$ и не укладывается в ограничение по времени.

Интуитивное решение

- Для каждого элемента *left* будем находить границу *right* такую, что $a[right] - a[left] \leqslant r$, а $a[right + 1] - a[left] > r$.
- Тогда для элемента *left* все элементы с $rigth + 1$ по n удалены на расстояние большее, чем r.
- Добавим к ответу $n - right$ и перейдем к следующему элементу (сдвинем *left*).
- Такое решение работает за $O(n^2)$ и не укладывается в ограничение по времени.

Интуитивное решение

- Для каждого элемента *left* будем находить границу *right* такую, что $a[right] - a[left] \leqslant r$, а $a[right + 1] - a[left] > r$.
- Тогда для элемента *left* все элементы с $rigth + 1$ по n удалены на расстояние большее, чем r.
- Добавим к ответу $n - right$ и перейдем к следующему элементу (сдвинем *left*).
- Такое решение работает за $O(n^2)$ и не укладывается в ограничение по времени.

Правильное решение

- Для каждого *left* не будем заново за линию находить *right*, а воспользуемся старым значением.
- Очевидно, что если $a[right] - a[left - 1] \leqslant r$, то и $a[right] - a[left] \leqslant r$ (поскольку массив отсотрирован по возрастанию).
- Таким образом для того, чтобы найти новую границу *right* нужно сдвинуть вправо старую.
- Будем сдвигать вправо, пока не «перескочим» r, и пересчитывем ответ.

Правильное решение

- Для каждого *left* не будем заново за линию находить *right*, а воспользуемся старым значением.
- Очевидно, что если $a[right] - a[left - 1] \leqslant r$, то и $a[right] - a[left] \leqslant r$ (поскольку массив отсотрирован по возрастанию).
- Таким образом для того, чтобы найти новую границу *right* нужно сдвинуть вправо старую.
- Будем сдвигать вправо, пока не «перескочим» r, и пересчитывем ответ.

Правильное решение

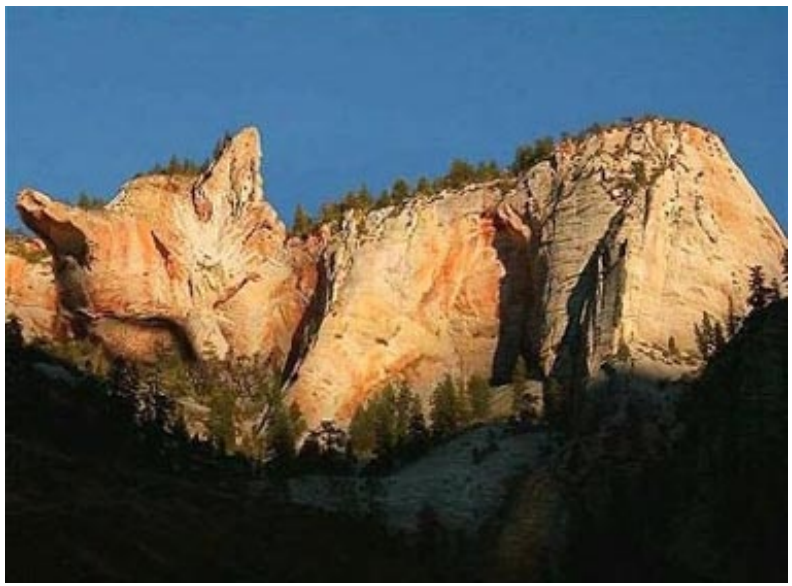
- Каждый элемент мы будем проходить два раза.
- Первый — когда на нем будет граница *right*, а второй — когда граница *left*.
- Таким образом решение работает за $O(n)$.
- Нужно было не забыть про 64-битный тип данных, поскольку ответ может быть порядка $O(n^2)$.

Правильное решение

- Каждый элемент мы будем проходить два раза.
- Первый — когда на нем будет граница *right*, а второй — когда граница *left*.
- Таким образом решение работает за $O(n)$.
- Нужно было не забыть про 64-битный тип данных, поскольку ответ может быть порядка $O(n^2)$.

Задача C. «Гномы и Одинокая гора»

Задача C. «Гномы и Одинокая гора»



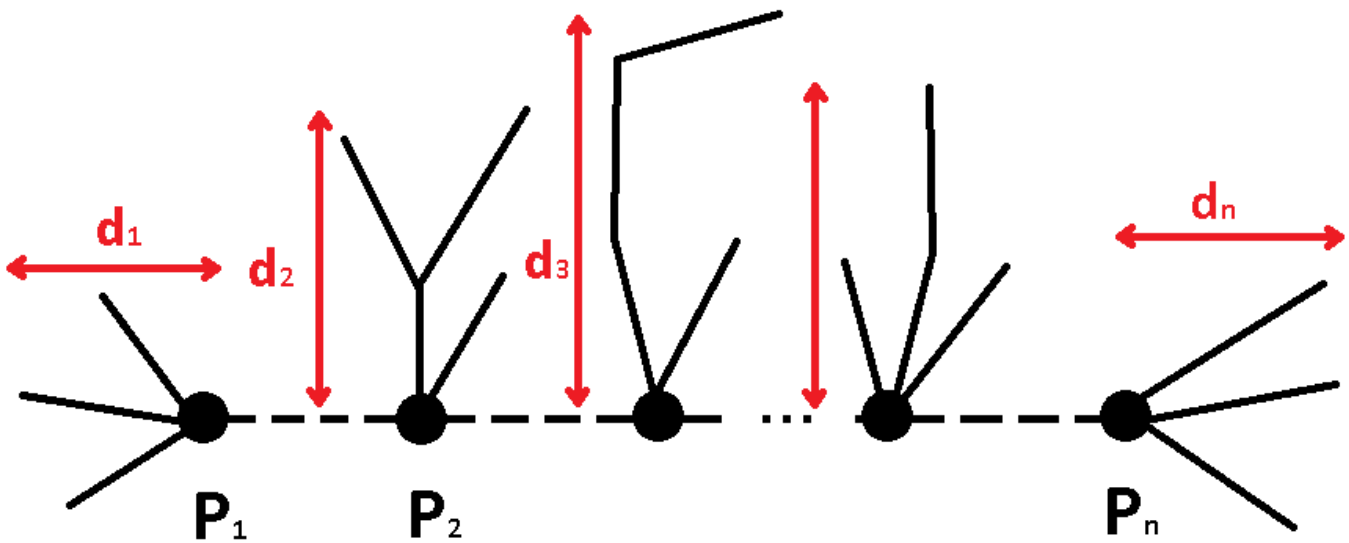
- Идея задачи — Нияз Нигматуллин
- Подготовка тестов — Нияз Нигматуллин
- Разбор задачи — Виталик Аксёнов

Постановка задачи

Дано дерево на n вершинах и две вершины u и v. Нужно найти такое максимальное d, что из u и из v существуют два неперсекающихся пути длины d.

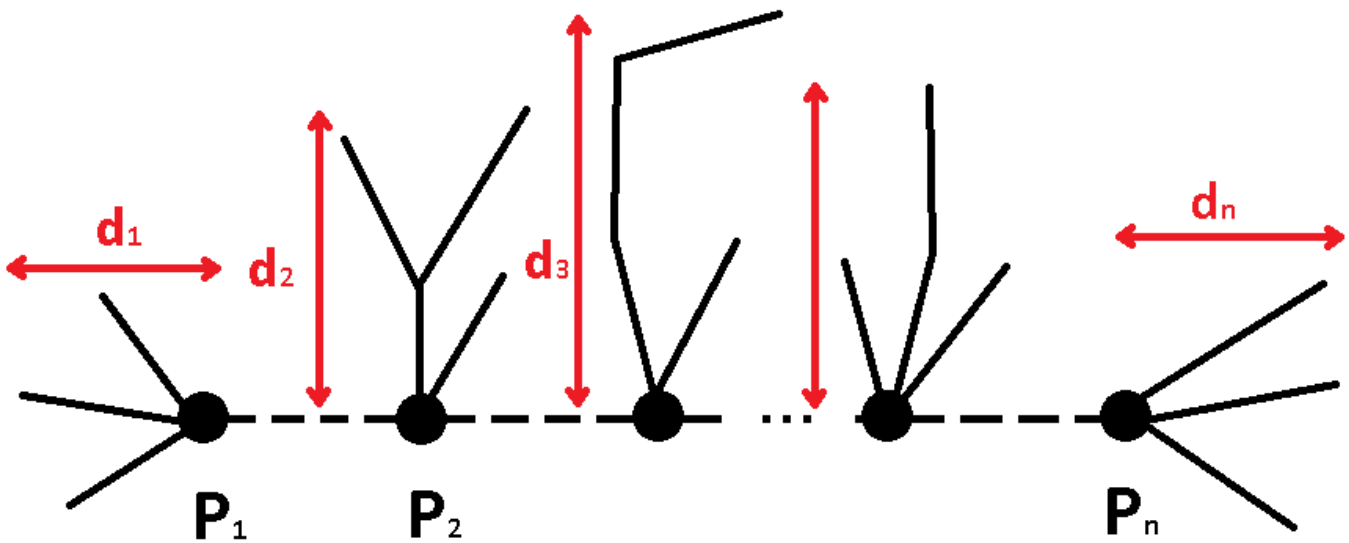
Решение задачи

- Найдём кратчайший путь $P = p_1\, p_2 \ldots p_k$ из u в v. И выкинем все рёбра этого пути из графа.
- Для каждой вершины p_i в оставшемся графе найдём расстояние до самой удалённой вершины — d_i, с помощью BFS.



Решение задачи

- Найдём кратчайший путь $P = p_1\, p_2 \ldots p_k$ из u в v. И выкинем все рёбра этого пути из графа.
- Для каждой вершины p_i в оставшемся графе найдём расстояние до самой удалённой вершины — d_i, с помощью BFS.



Решение задачи

- Тогда введём расстояния $a_i = i - 1 + d_i$ и $b_i = (k - i) + d_i$.
- Тогда ответ будет равен $\max\limits_{i=1}^{k-1} \min(\max\limits_{j=1}^{i} a_j, \max\limits_{j=i+1}^{n} b_j)$.

Решение задачи

- Тогда введём расстояния $a_i = i - 1 + d_i$ и $b_i = (k - i) + d_i$.
- Тогда ответ будет равен $\max\limits_{i=1}^{k-1} \min(\max\limits_{j=1}^{i} a_j, \max\limits_{j=i+1}^{n} b_j)$.

Задача D. «Покраска забора»

Задача D. «Покраска забора»

- Идея задачи — Глеб Евстропов
- Подготовка тестов — Глеб Евстропов
- Разбор задачи — Анна Малова

Постановка задачи

- n человек красят забор, состоящий из k досок.
- i-й человек красит непрерывный участок забора длины a_i, каждый участник может красить, как чистые доски, так и уже покрашенные.
- Необходимо максимизировать количество неокрашеных досок, которые покрасит каждый участник.

Решение задачи

- Запустим двоичный поиск по ответу x — количеству неокрашенных досок, которое покрасит каждый участник.
- Научимся делать проверку для фиксированного x.

Решение. Проверка.

Пусть x зафиксировано.

- Будем рассматривать участников в порядке возрастания a_i.
- Это выгодно, так как участнику надо покрасить только x чистых досок (остальные можно взять уже покрашенные).
- Если на текущем шаге не осталось хотя бы x чистых досок — это число не подходит.
- Осталось узнать, насколько меняется число уже покрашенных досок на каждом шаге.

Решение. Проверка.

Пусть x зафиксировано.

- Будем рассматривать участников в порядке возрастания a_i.
- Это выгодно, так как участнику надо покрасить только x чистых досок (остальные можно взять уже покрашенные).
- Если на текущем шаге не осталось хотя бы x чистых досок — это число не подходит.
- Осталось узнать, насколько меняется число уже покрашенных досок на каждом шаге.

Решение. Проверка.

Пусть x зафиксировано.

- Будем рассматривать участников в порядке возрастания a_i.
- Это выгодно, так как участнику надо покрасить только x чистых досок (остальные можно взять уже покрашенные).
- Если на текущем шаге не осталось хотя бы x чистых досок — это число не подходит.
- Осталось узнать, насколько меняется число уже покрашенных досок на каждом шаге.

Решение. Проверка. Шаг цикла.

Рассмотрим i-ого участника, пусть y — число уже покрашенных досок.

- Каждый участник должен покрасить x еще не покрашенных досок, то есть число покрашенных досок увеличится, как минимум на x.
- Но по условию i-ый участник красит не менее a_i досок, поэтому число покрашенных будет равно $\max(y + x, a_i)$.

Решение. Проверка. Шаг цикла.

Рассмотрим i-ого участника, пусть y — число уже покрашенных досок.

- Каждый участник должен покрасить x еще не покрашенных досок, то есть число покрашенных досок увеличится, как минимум на x.
- Но по условию i-ый участник красит не менее a_i досок, поэтому число покрашенных будет равно $\max(y + x, a_i)$.

Решение без двоичного поиска

- Отсортируем a_i по возрастанию.
- Инициализация: $ans = \min(a_1, \frac{n}{k})$.
- Далее перебираем для $i \in \{1, \ldots, n-1\}$ и выполняем: $ans = \min(ans, \frac{k-a_i}{n-i})$.

Задача E. «Телефонные номера»

Задача E. «Телефонные номера»

- Идея задачи — Андрей Станкевич
- Подготовка тестов — Павел Кротков
- Разбор задачи — Виталик Аксёнов

Постановка задачи

Даны различные коды стран и коды городов внутри них.
Нужно отформатировать номера телефонов «красивым образом» или сказать, что телефон неправильный.

Решение задачи

- Задачу нужно решать аккуратной реализацией.
- Для каждого телефонного номера переберём длину кода страны и длину кода города.
- Проверим, что соответствующие подстроки телефонного номера являются допустимыми кодами.

Решение задачи

- Задачу нужно решать аккуратной реализацией.
- Для каждого телефонного номера переберём длину кода страны и длину кода города.
- Проверим, что соответствующие подстроки телефонного номера являются допустимыми кодами.

Решение задачи

- Любая из трёх частей не может начинаться с 0.
- Выведем полученный номер способом, указанным в условии.
- Если не нашлось ни одного валидного разбиения, то надо вывести «Incorrect».

Решение задачи

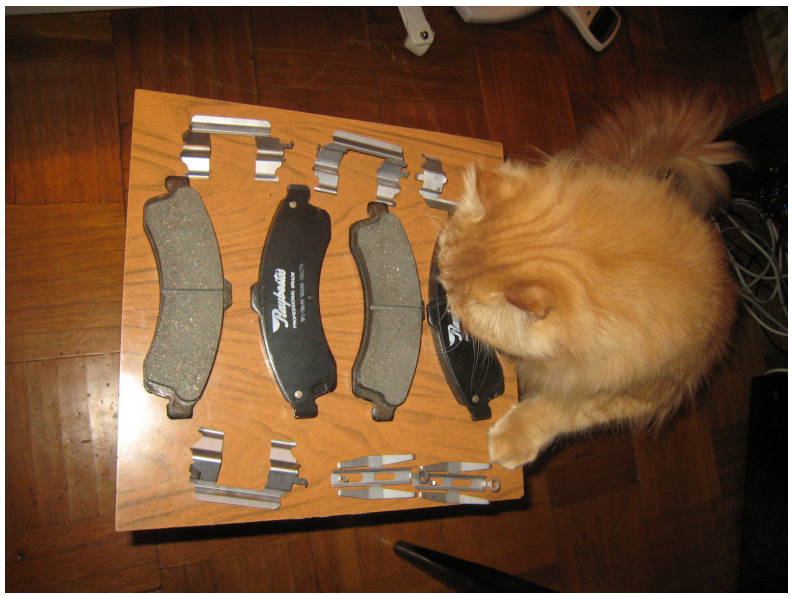
- Любая из трёх частей не может начинаться с 0.
- Выведем полученный номер способом, указанным в условии.
- Если не нашлось ни одного валидного разбиения, то надо вывести «Incorrect».

Решение задачи

- Любая из трёх частей не может начинаться с 0.
- Выведем полученный номер способом, указанным в условии.
- Если не нашлось ни одного валидного разбиения, то надо вывести «`Incorrect`».

Задача F. «Вращающаяся пластина»

Задача F. «Вращающаяся пластина»



- Идея задачи — Виталик Аксёнов
- Подготовка тестов — Павел Кунявский
- Разбор задачи — Павел Кунявский

Постановка задачи

- Дан невыпуклый многоугольник, без самопересечений и самокасаний.
- Многоугольник закреплен в одной из вершин, и может свободно вращаться вокруг нее.
- Есть много мест, куда можно вбить гвоздь. Для каждого надо сказать, на сколько градусов можно будет повернуть многоугольник, если вбить гвоздь в эту точку.

Решение

- Для упрощения можно вращать гвоздь, а не многоугольник. При этом поменяется местами вращение по и против часовой стрелки.
- При вращении гвоздя вокруг первой вершины гвоздь будет описывать окружность.
- Вращать можно до тех пор, пока гвоздь не наткнется на одну из сторон. Для того чтобы найти точку, до которой можно вращать, надо пересечь окружность со всеми сторонами многоугольника, и выбрать ближайшую из них по окружности слева и справа.

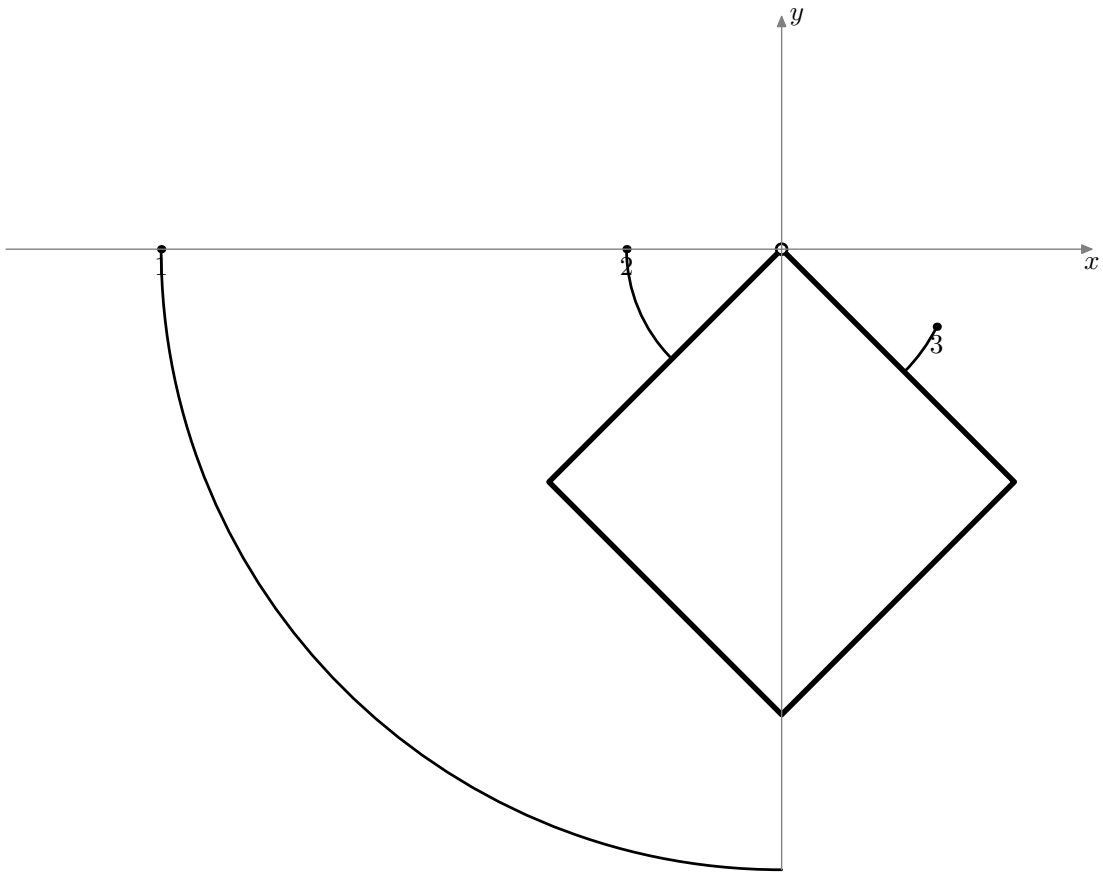
Решение

- Для упрощения можно вращать гвоздь, а не многоугольник. При этом поменяется местами вращение по и против часовой стрелки.
- При вращении гвоздя вокруг первой вершины гвоздь будет описывать окружность.
- Вращать можно до тех пор, пока гвоздь не наткнется на одну из сторон. Для того чтобы найти точку, до которой можно вращать, надо пересечь окружность со всеми сторонами многоугольника, и выбрать ближайшую из них по окружности слева и справа.

Решение

- Для упрощения можно вращать гвоздь, а не многоугольник. При этом поменяется местами вращение по и против часовой стрелки.
- При вращении гвоздя вокруг первой вершины гвоздь будет описывать окружность.
- Вращать можно до тех пор, пока гвоздь не наткнется на одну из сторон. Для того чтобы найти точку, до которой можно вращать, надо пересечь окружность со всеми сторонами многоугольника, и выбрать ближайшую из них по окружности слева и справа.

Решение



Задача G. «Призы»

Задача G. «Призы»

- Идея задачи — Георгий Корнеев
- Подготовка тестов — Борис Минаев
- Разбор задачи — Анна Малова

Постановка задачи

Есть комната размерами $n \times m$. Необходимо перекатить ящик со сторонами, равными 1, из угла комнаты в противоположный угол так, чтобы:

- Минимизировать количество перекатываний ящика.
- Среди всех таких вариантов выбрать вариант, при котором надпись окажется на нижней грани куба наименьшее число раз.

Решение. Количество перекатываний.

- Выгодно перекатывать кубик только вниз и вправо.
- Тогда количество действий составит $n + m$.

Решение. Количество окрашенных клеток.

- Первый случай: $n = 1$ или $m = 1$.
  - В этом случае кубик перекатывается только в одном направлении.
  - Количество испачканных клеток пола $\left\lfloor \frac{n+m}{4} \right\rfloor$.
- Второй случай: $n = 2$ или $m = 2$
  - Перекатим кубик на один вниз, а потом докатим до конца.
  - В этом случае ни одна клетка пола не будет окрашена.

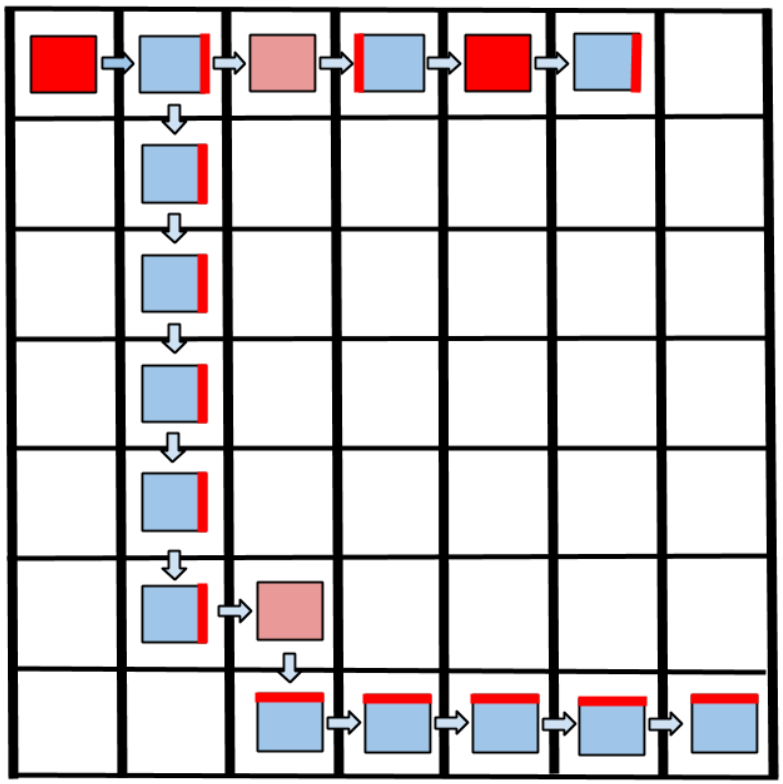
Решение. Количество окрашенных клеток.

- Первый случай: $n = 1$ или $m = 1$.
  - В этом случае кубик перекатывается только в одном направлении.
  - Количество испачканных клеток пола $\lfloor \frac{n+m}{4} \rfloor$.
- Второй случай: $n = 2$ или $m = 2$
  - Перекатим кубик на один вниз, а потом докатим до конца.
  - В этом случае ни одна клетка пола не будет окрашена.

Решение. Количество окрашенных клеток.

- Третий случай. $n \geqslant 3$ и $m \geqslant 3$.
  - В этом случае всего одна клетка пола будет окрашена.
  - Для достижения результата перекатим кубик один раз направо.
  - Потом почти до конца будем катить вниз.
  - Перекатим один раз вправо.
  - Перекатим один раз вниз.
  - Катим до конца вправо.

Решение. Количество окрашенных клеток.



Задача H. «Защищенное соединение»

Задача H.
«Защищенное соединение»

- Идея задачи — Сергей Шедов
- Подготовка тестов — Артем Васильев
- Разбор задачи — Артем Васильев

Входные данные:

- Дан взвешенный граф из n вершин и m ребер.
- Вершины покрашены в три цвета: черный, белый, и серый.

Необходимо найти такую пару «черная вершина-белая вершина», что расстояние между ними минимально.

Алгоритм Флойда

- Найдем кратчайшие расстояния для всех пар вершин, используя алгоритм Флойда.
- Переберем все пары из черных и белых вершин и выберем пару с минимальным расстоянием.
- Сложность: $O(N^3)$.
  Time limit exceeded.

Алгоритм Флойда

- Найдем кратчайшие расстояния для всех пар вершин, используя алгоритм Флойда.
- Переберем все пары из черных и белых вершин и выберем пару с минимальным расстоянием.
- Сложность: $O(N^3)$.
  Time limit exceeded.

Алгоритм Флойда

- Найдем кратчайшие расстояния для всех пар вершин, используя алгоритм Флойда.
- Переберем все пары из черных и белых вершин и выберем пару с минимальным расстоянием.
- Сложность: $O(N^3)$.

  Time limit exceeded.

Алгоритм Флойда

- Найдем кратчайшие расстояния для всех пар вершин, используя алгоритм Флойда.
- Переберем все пары из черных и белых вершин и выберем пару с минимальным расстоянием.
- Сложность: $O(N^3)$.
  Time limit exceeded.

Алгоритм Дейкстры

- Запустим алгоритм Дейкстры из всех черных вершин.
- Для каждой черной вершины посмотрим на все белые вершины и обновим ответ расстоянием до них.
- Сложность: $O(N^3)$ или $O(NM \log N)$.
  Time limit exceeded.

Алгоритм Дейкстры

- Запустим алгоритм Дейкстры из всех черных вершин.
- Для каждой черной вершины посмотрим на все белые вершины и обновим ответ расстоянием до них.
- Сложность: $O(N^3)$ или $O(NM \log N)$.
  Time limit exceeded.

Алгоритм Дейкстры

- Запустим алгоритм Дейкстры из всех черных вершин.
- Для каждой черной вершины посмотрим на все белые вершины и обновим ответ расстоянием до них.
- Сложность: $O(N^3)$ или $O(NM \log N)$.

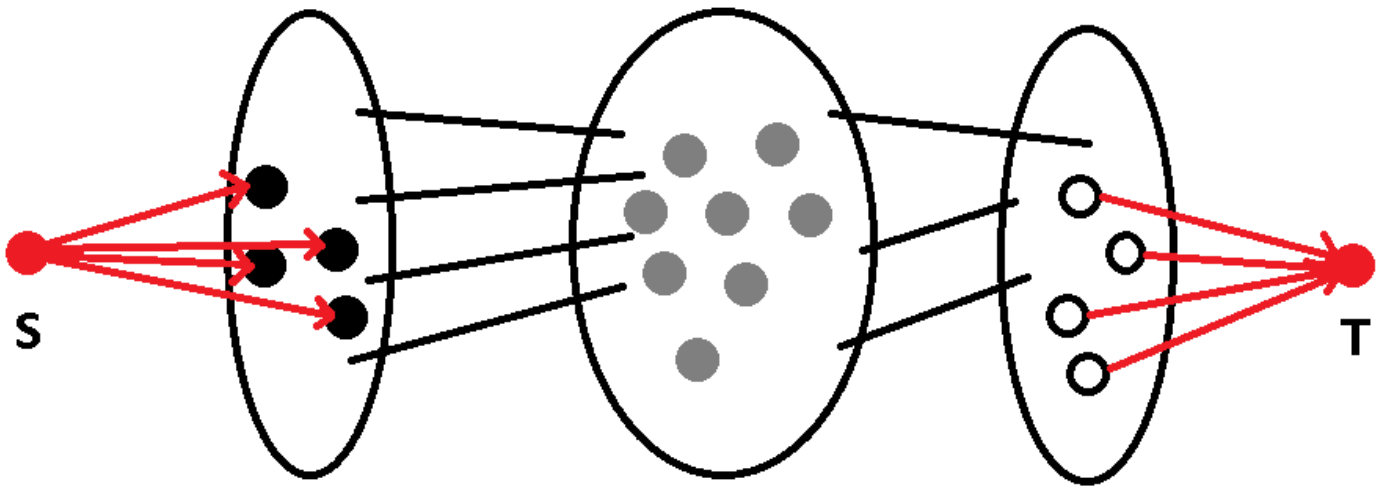
  Time limit exceeded.

Алгоритм Дейкстры

- Запустим алгоритм Дейкстры из всех черных вершин.
- Для каждой черной вершины посмотрим на все белые вершины и обновим ответ расстоянием до них.
- Сложность: $O(N^3)$ или $O(NM \log N)$.
  Time limit exceeded.

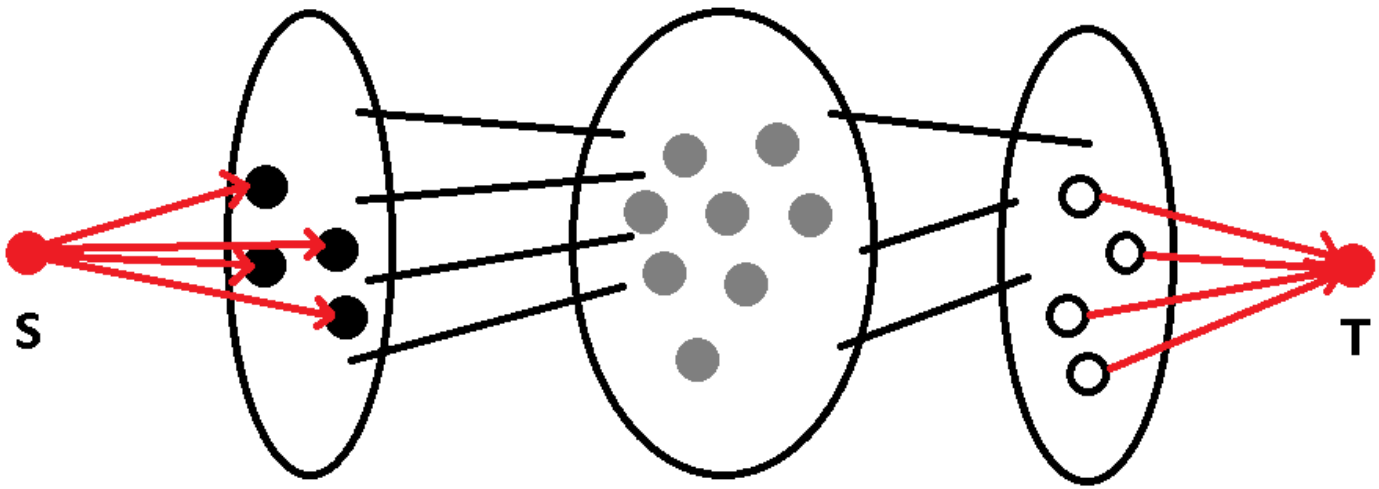
Модификация графа

- Добавим две фиктивные вершины S и T.
- Добавим ребра нулевого веса из S во все черные вершины, из T — во все белые.



Модификация графа

- Добавим две фиктивные вершины S и T.
- Добавим ребра нулевого веса из S во все черные вершины, из T — во все белые.



Окончательное решение

- Найдем кратчайший путь между S и T.
- Искомым расстоянием будет длина этого кратчайшего пути.
- Соответствующими этому расстоянию черной и белой вершинами будут вторая и предпоследняя на кратчайшем пути из S в T.
- Сложность: $O(N^2)$ или $O(M \log N)$.
  Accepted

Окончательное решение

- Найдем кратчайший путь между S и T.
- Искомым расстоянием будет длина этого кратчайшего пути.
- Соответствующими этому расстоянию черной и белой вершинами будут вторая и предпоследняя на кратчайшем пути из S в T.
- Сложность: $O(N^2)$ или $O(M \log N)$.
  Accepted

Окончательное решение

- Найдем кратчайший путь между S и T.
- Искомым расстоянием будет длина этого кратчайшего пути.
- Соответствующими этому расстоянию черной и белой вершинами будут вторая и предпоследняя на кратчайшем пути из S в T.
- Сложность: $O(N^2)$ или $O(M \log N)$.
  Accepted

Окончательное решение

- Найдем кратчайший путь между S и T.
- Искомым расстоянием будет длина этого кратчайшего пути.
- Соответствующими этому расстоянию черной и белой вершинами будут вторая и предпоследняя на кратчайшем пути из S в T.
- Сложность: $O(N^2)$ или $O(M \log N)$.

  Accepted

Окончательное решение

- Найдем кратчайший путь между S и T.
- Искомым расстоянием будет длина этого кратчайшего пути.
- Соответствующими этому расстоянию черной и белой вершинами будут вторая и предпоследняя на кратчайшем пути из S в T.
- Сложность: $O(N^2)$ или $O(M \log N)$.
  Accepted

Задача I. «Перетягивание каната»

Задача I.
«Перетягивание каната»

- Идея задачи — Анна Малова
- Подготовка тестов — Андрей Комаров
- Разбор задачи — Анна Малова

Постановка задачи

Есть n кусков каната различной длины. Необходимо связать некоторые из этих кусков так, чтобы:

- Расстояние между соседними узлами было не меньше d.
- Длина получившегося каната была максимальной. При связывании двух кусков, уходит по d длины каната с каждого конца.

Решение задачи

Поделим все кусочки каната на три группы, в зависимости от длины:

- Куски каната, чья длина в диапазоне $[3d; +\infty)$, которые могут быть связаны с двух сторон.
- Куски каната, чья длина в диапазоне $[2d; 3d)$, которые могут быть связаны только одним концом.
- Куски каната, чья длина в диапазоне $[1; 2d)$, которые нельзя связать с остальными.

Для нахождения ответа необходимо взять все куски из первой группы и два максимальных по длине куска из второй.

Решение задачи

Поделим все кусочки каната на три группы, в зависимости от длины:

- Куски каната, чья длина в диапазоне $[3d; +\infty)$, которые могут быть связаны с двух сторон.
- Куски каната, чья длина в диапазоне $[2d; 3d)$, которые могут быть связаны только одним концом.
- Куски каната, чья длина в диапазоне $[1; 2d)$, которые нельзя связать с остальными.

Для нахождения ответа необходимо взять все куски из первой группы и два максимальных по длине куска из второй.

Решение задачи

Поделим все кусочки каната на три группы, в зависимости от длины:

- Куски каната, чья длина в диапазоне $[3d; +\infty)$, которые могут быть связаны с двух сторон.
- Куски каната, чья длина в диапазоне $[2d; 3d)$, которые могут быть связаны только одним концом.
- Куски каната, чья длина в диапазоне $[1; 2d)$, которые нельзя связать с остальными.

Для нахождения ответа необходимо взять все куски из первой группы и два максимальных по длине куска из второй.

Решение задачи

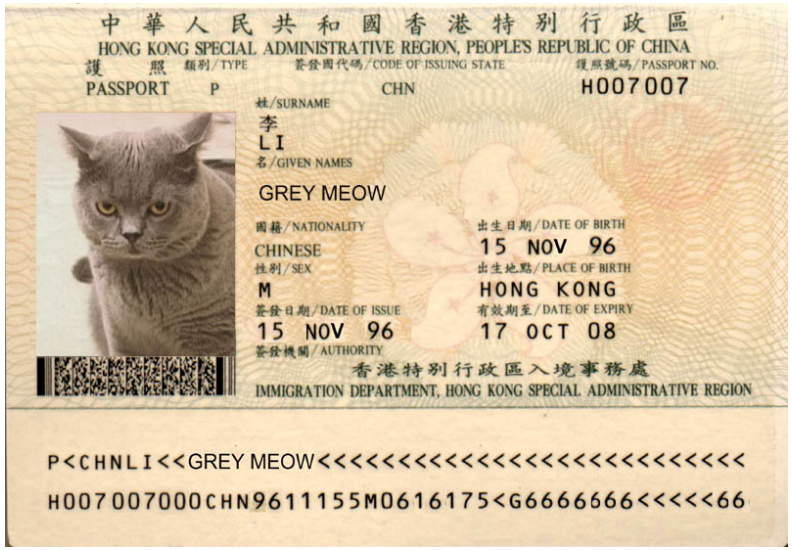
Поделим все кусочки каната на три группы, в зависимости от длины:

- Куски каната, чья длина в диапазоне $[3d; +\infty)$, которые могут быть связаны с двух сторон.
- Куски каната, чья длина в диапазоне $[2d; 3d)$, которые могут быть связаны только одним концом.
- Куски каната, чья длина в диапазоне $[1; 2d)$, которые нельзя связать с остальными.

Для нахождения ответа необходимо взять все куски из первой группы и два максимальных по длине куска из второй.

Задача J. «Загранпаспорт»

Задача J. «Загранпаспорт»



- Идея задачи — Жюри
- Подготовка тестов — Олег Давыдов
- Разбор задачи — Виталик Аксёнов

Постановка задачи

- Дана таблица $w \times h$, состоящая из символов «A» и «T».
- Герой стоит в какой-то ячейке «A».

Нужно посчитать минимальное число шагов и соответствующий путь, чтобы посетить ровно n символов «T».

Идея

- Можно дойти до ближайшего символа «T», и бегать с него на символ «A» и обратно.
- Можно дойти до каких-то двух соседних клеток с символами «T», и потом бегать с одной на другую.
- Понятно, что можно обойтись только этими двумя вариантами.

Первый случай

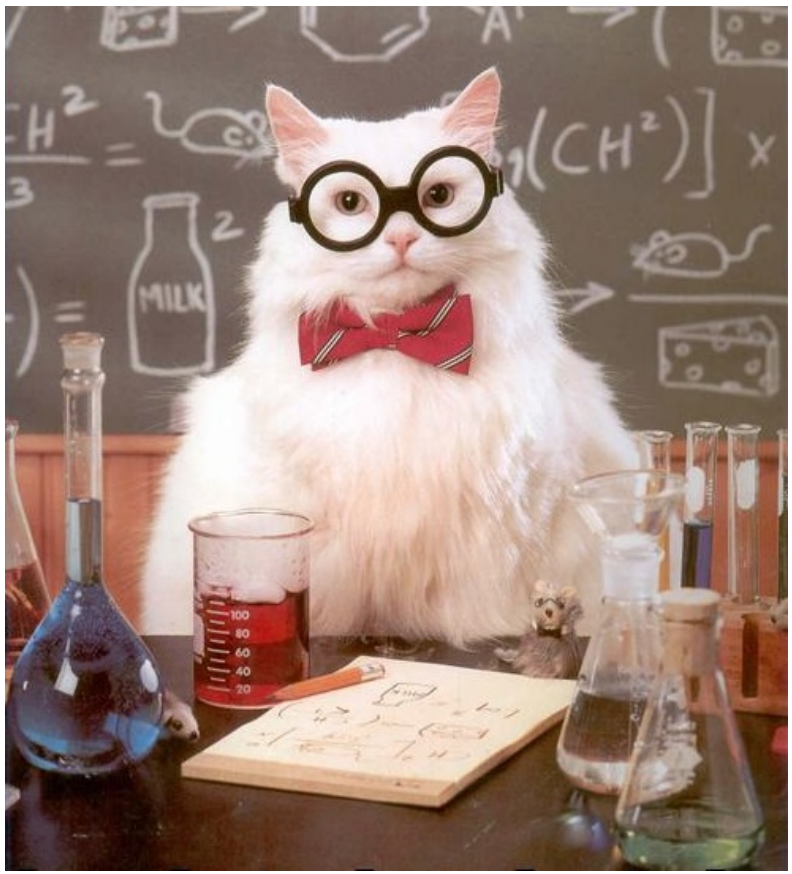
- Первый случай обрабатывается легко.
- Пусть d — расстояние до ближайшего символа «T».
- То ответ для этого случая будет равен $d + 2 \cdot (n - 1)$.

Второй случай

- На пути в «парный» символ «T» мы можем пройти через какие-то другие символы «T».
- Заметим, что ответ равен минимальному количеству символов «A» на пути в «парный символ» плюс n.
- Для нахождения такого пути построим граф: любое ребро в вершину «T» будет иметь вес 0, в вершину «A» — вес 1.
- Данная задача решается стандартным алгоритмом 0-1-BFS.
- Для восстановления ответа нужно для каждой клетки хранить — из какой мы в неё попали.

Задача K. «Вирус»

Задача K. «Вирус»



- Идея задачи — Андрей Станкевич
- Подготовка тестов — Виталик Аксёнов
- Разбор задачи — Виталик Аксёнов

Постановка задачи

- Дано поле размерами не больше 100×100, и не более чем в восьми клетках находится вирусы.
- Каждую секунду заражается ровно одна клетка, у которой есть заражённый сосед.

Нужно было найти количество различных конфигураций заражённых клеток спустя не более чем 6 секунд.

Перебор

- По ограничениям сразу понятно, что в данной задаче должен заходить перебор.
- Но максимальный ответ на эту задачу превышает 8 миллионов.
- Поэтому переборы, которые используют запоминание состояний, например, в хеш-сет не проходят по времени.

Перебор

- По ограничениям сразу понятно, что в данной задаче должен заходить перебор.
- Но максимальный ответ на эту задачу превышает 8 миллионов.
- Поэтому переборы, которые используют запоминание состояний, например, в хеш-сет не проходят по времени.

Перебор

- По ограничениям сразу понятно, что в данной задаче должен заходить перебор.
- Но максимальный ответ на эту задачу превышает 8 миллионов.
- Поэтому переборы, которые используют запоминание состояний, например, в хеш-сет не проходят по времени.

Перебор за величину ответа

- Будем генерировать ответ рекурсивно с возвратом.
- Храним два массива *a* и *tried* — состояние и клетки, которые мы бы могли бы поставить, если бы хотели, и сейчас их ставить уже не будем.
- Тогда псевдокод будет выглядеть следующим образом.

Перебор за величину ответа

```
begin
    if it == 0 then
        ans++;
        return;
    end
    back ← List();
    for (x, y) имеет заражённого соседа do
        if !a[x][y] && !tried[x][y] then
            a[x][y] = true;
            tried[x][y] = true;
            dfs(it − 1);
            a[x][y] = false;
            back.add((x, y));
        end
    end
    for (x, y) in back do
        tried[x][y] = false;
    end
end
```

Algorithm 1: dfs(it)

Вопросы?