XXVIII командный чемпионат школьников Санкт-Петербурга по программированию

15 ноября 2020 года

Задача A
Прыгающий аппарат

Автор задачи:
Степан Филиппов
Разработка задачи:
Арсений Кириллов

Постановка задачи

- У аппарата есть набор пружин, который позволяет ему из клетки `(x, y)` попасть
  - либо в клетку $(x + l_i, y)$
  - либо в клетку $(x, y + l_i)$
- Надо посчитать количество клеток, над которыми может пролететь аппарат

Ключевая идея: нужны только Г-пути и ⅃-пути

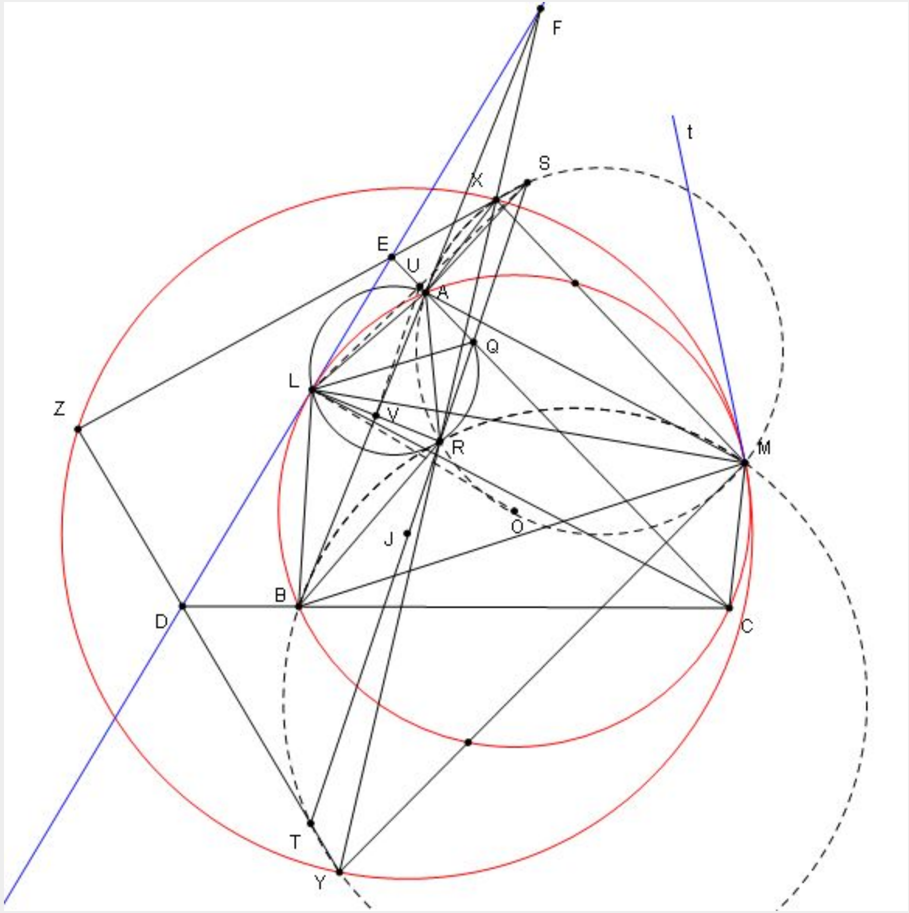
- Посмотрим на любой путь и на любой вертикальный отрезок в нём
- Нам не важно, в каком порядке шли предыдущие отрезки, если они приведут в ту же точку
- Поэтому можно считать, что мы сначала использовали горизонтальные отрезки, а потом вертикальные
- Аналогично, нужно рассмотреть пути, в которых сначала вертикальные отрезки, а потом горизонтальные

Решение задачи

- Посчитаем с помощью задачи о рюкзаке все возможные длины первых частей пути, которые мы можем получить
- Если `S` – сумма сил всех пружин, а `l` – это сумма некоторых из них, то надо в ответ добавить все точки `(l, x)` и `(x, l)`, где `x ≤ S - l`

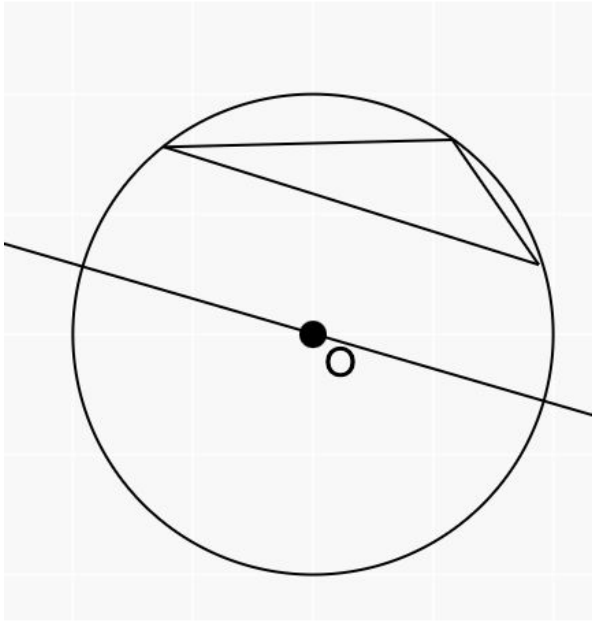
Задача B
Треугольники и окружность

Автор задачи:
Даниил Орешников
Разработка задачи:
Ильдар Гайнуллин



Постановка задачи

- Даны точки на окружности
- Посчитать число треугольников с вершинами в них, которые содержат центр (внутри или на границе)



Решение задачи

- Треугольник не содержит центр, если есть диаметр, «отделяющий» треугольник от центра
- В постановке с углами это значит, что расстояние между какими-то двумя соседними на окружности точками не меньше L/2
- Причем каждый «плохой треугольник» содержит не больше одной такой «плохой пары»
- Для каждой «плохой пары» точек посчитаем количество треугольников, в которых они соседние
- В сумме получим: C_n^3 - <ответ на задачу>

Решение задачи

- Отсортируем точки $x_1 < x_2 < \ldots < x_n$
- Два случая
- Упорядоченная плохая пара – это (x_i, x_j), где $i < j$, тогда число подходящих третьих точек – это $i - 1 + n - j$
- Или это (x_i, x_j), где $i > j$, тогда число подходящих третьих точек — это $i - j + 1$
- Множество подходящих j можно поддерживать «двумя указателями»
- Суммирование по ним: сумма арифметической прогрессии
- Итого: $O(n \times \log n)$ от сортировки + $O(n)$ от «двух указателей»

Задача C
Игра

Автор задачи:
Андрей Станкевич
Разработка задачи:
Ильдар Загретдинов

Постановка задачи

- Игра с вычитанием случайного числа из текущего счёта и началом нового раунда при счёте ноль
- Что будет после k ходов при данных случайных числах начиная с данного счёта n?

Решение: моделирование

- Количество очков больше чем очередное число – уменьшаем
- Меньше – игнорируем
- Достигается равенство – количество сыгранных раундов увеличивается, количество очков становится ноль, раунд считается сыгранным, перед следующим ходом ставим количество очков, равное n
    - в случае, если это был последний ход, мы бы вывели, что количество очков `0`, что и является верным ответом

Задача D
Забор

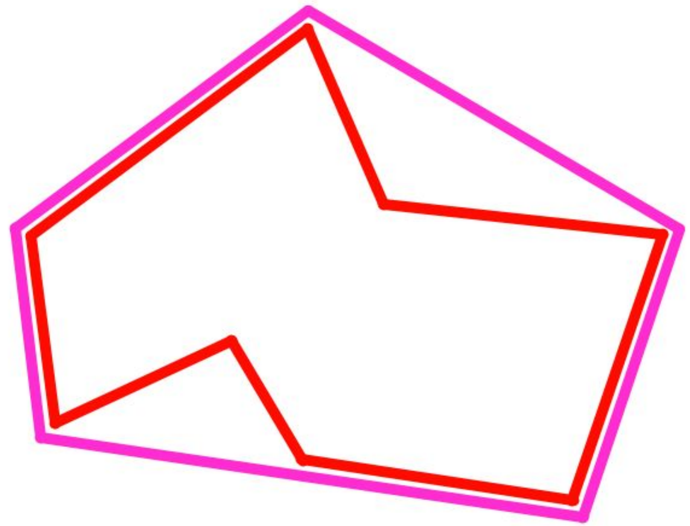
Автор задачи:
Георгий Корнеев
Разработка задачи:
Дмитрий Саютин

Постановка задачи

- Есть дом – многоугольник, у которого все стороны или вертикальные, или горизонтальные
- Требуется построить вокруг дома забор минимальной длины, так что манхэттенское расстояние между любой точкой дома и любой точкой стены было хотя бы L

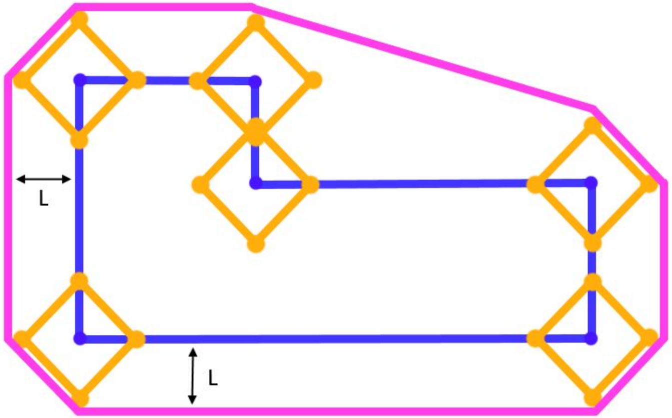
Первая идея: выпуклость

- Заметим, что забор не может быть невыпуклым, в противном случае его выпуклая оболочка строго короче (розовый забор короче красного)



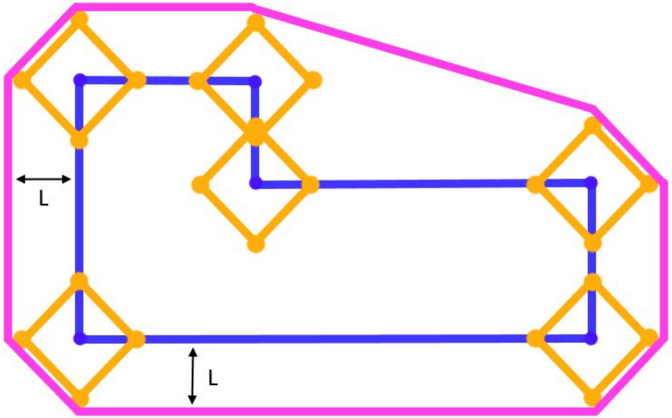
Вторая идея: окрестности вершин

- Для каждой вершины `(x, y)` отметим (на картинке оранжевым) точки `(x ± L, y)` и `(x, y ± L)`
- Заметим что они точно должны лежать внутри забора (или на границе)



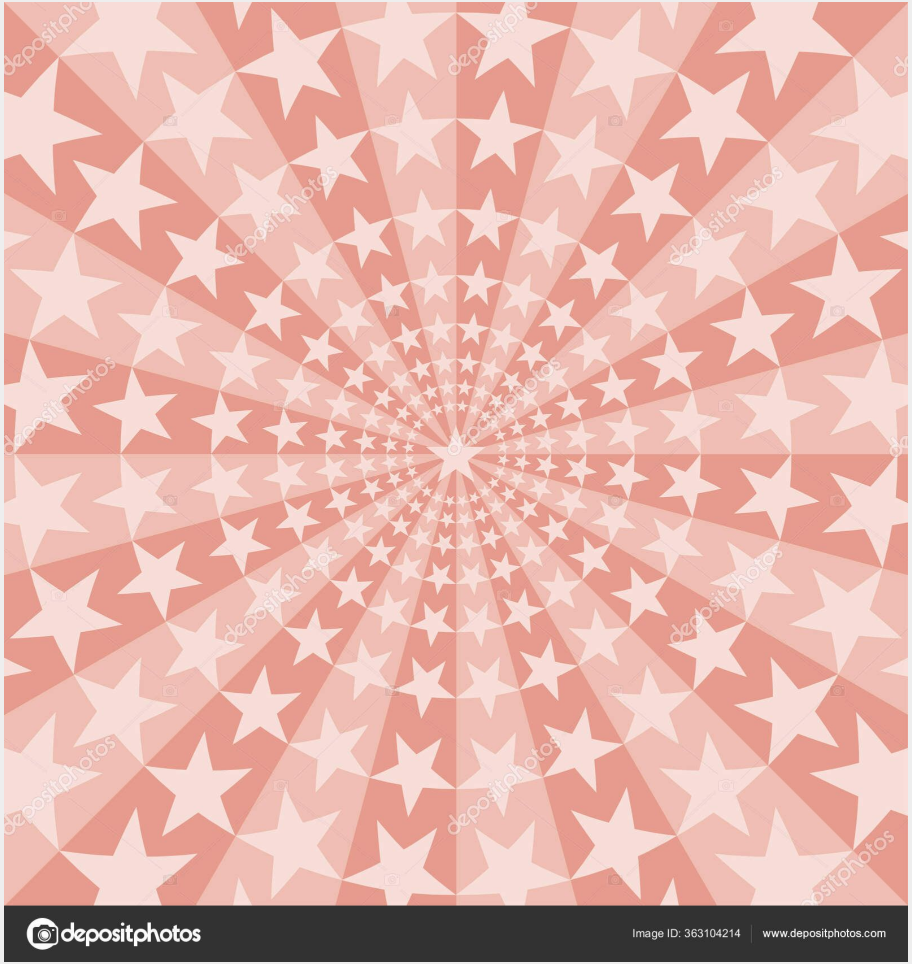
Вторая идея: окрестности вершин

- Выпуклая оболочка этих точек подходит как ответ
- Значит, ответ ей и является, достаточно вычислить выпуклую оболочку и измерить её периметр
- Это можно сделать за `O(n log n)`



Задача E
Флаг со звёздами

Автор задачи:
Дмитрий Штукенберг
Разработка задачи:
Виктор Шарепо



Постановка задачи

- Нужно расположить n звёзд в горизонтальные ряды, у которых длины либо совпадают, либо чередуются и различаются на единицу
- Определить минимальный возможный модуль разности числа рядов и максимального числа звёзд в ряду

Решение задачи

- У любого корректного расположения либо минимальное число звезд в ряду, либо число рядов не больше корня из `n`, так как `n` не меньше их произведения
- Можно перебрать все возможные расположения за время `O(sqrt(n))`:
    - переберем число звезд в первом ряду и три значения разности с числом звезд во втором
    - переберем число рядов и также три значения разности

Решение задачи. Первый перебор

- Расположение из `n` звезд с заданными числом звезд в первом (`= first`) и втором ряду (`= second`) существует в двух случаях:
  - `n` делится на `(first + second)`
  - `(n - first)` делится на `(first + second)`
- В обоих случаях тривиально определить число рядов и максимальное число звезд в одном из них, чтобы обновить ответ

Решение задачи. Второй перебор

- Если существует расположение из `n` звезд с заданными числом рядов (`= rows`) и разностью числа звезд в первом и втором ряду (`= difference`), то число звезд в первом ряду (`= first`) равно:
  - `(n / (rows / 2) - difference) / 2`, если `rows` четное
  - `(n - difference · (rows - 1) / 2) / rows` иначе
- Вычислим это значение и, если оно является целым числом, перейдем к рассмотрению двух случаев с предыдущего слайда.

Задача F
Стринг-арт

Автор задачи:
Павел Маврин
Разработка задачи:
Александра Дроздова

Постановка задачи

- Дан неориентированный связный граф, вершины которого соответствуют гвоздям, а рёбра – ниточкам
- Требуется построить по нему дерево, у каждой вершины которого есть цвет, такое, что после склеивания вершин одинакового цвета получится исходный граф

Решение задачи

- Рассмотрим любое остовное дерево исходного графа
- Дерево подходит под условия на результирующий граф, и в нем ничего менять не нужно
- Достроим это дерево так, чтобы после сжатия оно превратилось в исходный граф

Решение задачи

- Для этого для каждого оставшегося ребра можно добавить вершину того же цвета, что любой из концов ребра, и провести из новой вершины ребро в другой конец рассматриваемого ребра; граф остается деревом
- После склеивания новой вершины и ее соответствующего конца, как раз получится нужное ребро не из остовного дерева
- Найти остовное дерево можно, например, поиском в глубину
- Асимптотика: `O(n + m)`

Задача G
Мороженое

Автор задачи:
Иван Волков
Разработка задачи:
Иван Волков

Постановка задачи

- Есть n стаканчиков мороженого, в i-м – a_i грамм
- В какой-то момент все мороженые начинают таять со скоростью v
- В тот же момент вы начинаете есть их по одному со скоростью u
- Все продолжается, пока мороженое не закончится; останавливаться нельзя
- Нужно минимизировать массу съеденного мороженого

Решение задачи

- Минимизация массы ⇔ минимизация времени
- Оптимально в каждый момент времени есть то мороженое, которого осталось больше всего (если таких несколько, то их надо есть параллельно):
  - если в какой-то момент мороженое стало максимальным, будем есть его до конца, иначе – не будем есть совсем
  - если таким алгоритмом добились времени t, то из i-го стаканчика растаяло $\min(a_i, tv)$ – максимум для фиксированного t
  - Для любого меньшего чем t времени

    $m_{\text{растаявшего}} + m_{\text{съеденного}} < \sum a_i$

Решение задачи

- Решение 1: вещественный двоичный поиск по ответу (при фиксированном `t` сравниваем, сколько можем успеть съесть + сколько может успеть растаять с общей исходной массой)
  - `O(n log C)`
- Решение 2: сортировка + симуляция
  - `O(n log n)`

- Бонус: если уже нашли оптимальное `t`, то достичь его можно просто по очереди съедая $\max(0,\ a_i - tv)$ мороженого ⇒ не придется бесконечно переключаться между стаканчиками

Задача H
Хоровод разнообразия

Автор задачи:
Ильдар Гайнуллин
Разработка задачи:
Даниил Орешников

Постановка задачи

- Есть n пар объектов, каждый объект какого-то цвета
- Объекты из одной пары обязаны стоять рядом
- Пары цветов `(x, y)` и `(z, t)` можно ставить рядом, не меняя порядок внутри пар, только если `y ≠ z`
- Требуется выбрать максимально возможное число пар, которые можно расставить по кругу

Решение задачи (случай 1)

- Первое наблюдение – если нет пар вида `(x, x)` (все пары «разноцветные»), то всегда можно расставить в круге всех
- Действительно, если рассмотреть пары `(x, y)` и `(z, t)`, то
  - если `x = t` или `y = z`, поставим их как `(x, y), (t, z)`
  - иначе, поставим их как `(x, y), (z, t)`
- В обоих случаях мы получаем четверку объектов, причем крайние объекты разных цветов
- Рассмотрим теперь эту четверку как пару `(x, z)` или `(x, t)` соответственно и повторим второй шаг

Решение задачи

- Докажем более общее утверждение – k пар можно расставить $\Leftrightarrow$ нет $k + 1$ объекта одного цвета
- Если есть такой цвет, то после его расстановки на $2k$ мест, хотя бы у двух соседних пар будут крайние объекты этого цвета, что запрещено условием
  - будем выкидывать одноцветные пары с самым часто встречающимся цветом, пока условие не выполнится
- Иначе сведем задачу к первому случаю, объединив некоторые пары в четверки и убрав из рассмотрения центральные объекты этих четверок

Решение задачи (случай 2)

- Сопоставим каждой паре вида `(x, x)` одну пару вида `(y, y)` или `(z, y)`, чтобы полученные четверки можно было рассматривать как пары `(x, y)` с двумя «спрятанными» внутри объектами, а затем воспользуемся решением первого случая
- Рассмотрим все «одноцветные» пары, пусть их всего a, и рассмотрим три случая:
  - есть цвет x такой, что пар `(x, x)` больше половины
  - такого цвета нет, и a четно
  - такого цвета нет, и a нечетно

Решение задачи (случай 2.1)

- Если пар `(x, x)` больше, чем `a/2`, обозначим их количество за b и сопоставим каждой из `a - b` оставшихся одноцветных пар пару `(x, x)`
- Пар `(x, x)` осталось `2b - a`
- Среди «разноцветных» пар было не более `k - 2b` пар с объектом цвета x, потому что всего цвета x не более k
- Соответственно есть хотя бы `(k - a) - (k - 2b) = 2b - a` разноцветных пар без x
- Сопоставим оставшиеся `(x, x)` и разноцветные пары без x, получим сведение к случаю 1

Решение задачи (случай 2.2)

- Если одноцветных пар каждого одного цвета не более половины `a`, и `a` четно, выпишем все одноцветные пары подряд
- Сопоставим друг другу пары на местах `i` и `i + a/2`
- Они обязательно будут разных цветов, потому что если они одного цвета, этот цвет занимает строго больше `a/2` пар
- После этого мы получим `a/2` четверок с разными цветами по краям, и сведем задачу к случаю 1

Решение задачи (случай 2.3)

- Если одноцветных пар каждого цвета не более половины `a`, и `a` нечетно, сопоставим все кроме одной как в случае 2.2, а оставшуюся поставим в пару разноцветной
- Чтобы так можно было сделать, выберем такую одноцветную пару `(x, x)`, что существует разноцветная пара `(y, z)`, где `y ≠ x` и `z ≠ x`
- Для этого заметим, что у всех разноцветных пар не более двух общих цветов, а при нечетном `a`, если каждого цвета не более половины, среди одноцветных пар есть хотя бы три разных цвета, значит есть подходящий

Время работы

- Выкинуть «лишние» одноцветные пары
- Распределить одноцветные пары в четверки
  - несколько проходов по всем парам
- Объединить оставшиеся пары и четверки в круг

- Суммарное время работы – `O(n)`

Задача I
Три тропинки

Автор задачи:
Федор Царев
Разработка задачи:
Дмитрий Гнатюк

Постановка задачи

- Есть `n` лужаек, соединенных `m` тропинками
- Нужно посчитать количество путей длины `3`, старт которых в первой вершине, а конец в одной из соседних со стартовой
- Пути, проходящие по какому-то ребру хотя бы два раза, не учитываются

Решение задачи

- Воспользуемся динамическим программированием: `dp[i][j]` – количество путей длины `i`, заканчивающихся в `j`-й вершине.

- База: `dp[0][0] = 1`, `dp[0][j] = 0`
- Пересчет: `dp[i][j] =` $\sum_{k \mapsto j}$ `dp[i - 1][k]`

- Таким образом, мы почти посчитали ответ, осталось учесть пути, которые используют одно и тоже ребро несколько раз
- Для этого нам понадобиться посчитать степени вершин (`degree[i]` = степень вершины `i`)

Решение задачи

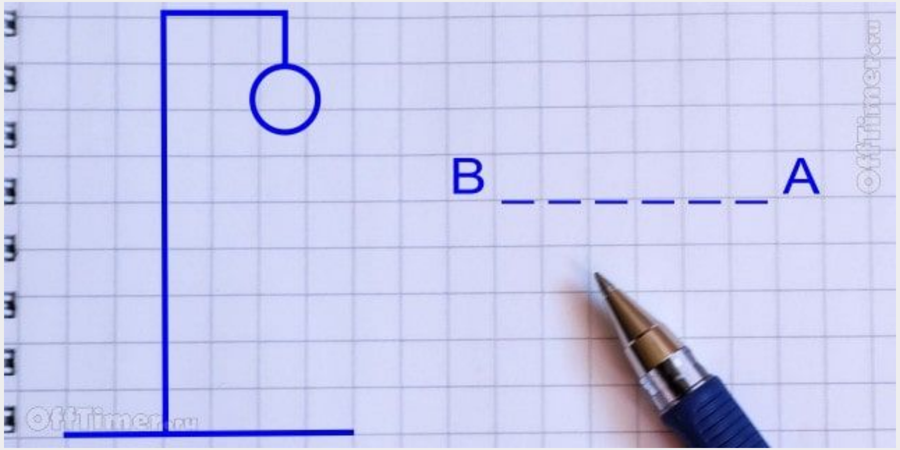
- Учтем пути вида `s ↦ i ↦ s ↦ j`
- Для фиксированной вершины `j`, в ДП учтено `degree[s]` таких путей (`i` – любой сосед первой вершины)
- Осталось рассмотреть пути вида `s ↦ i ↦ j ↦ i`
- Для фиксированной вершины i, в ДП учтено `degree[i]` таких путей (`j` – любой сосед конечной вершины)

- Путь `s ↦ i ↦ s ↦ i` учтен оба раза, поэтому ответ равен

$$\sum_{s \mapsto j} dp[3][j] - \sum_{s \mapsto i} (degree[s] + degree[i] - 1)$$

Задача J
Скучная пара

Автор задачи:
Николай Будин
Разработка задачи:
Николай Будин



Постановка задачи

- Есть строки `s`, `t` и w_i
- Нужно за минимальное количество вставок, удалений и замен символа преобразовать `s` в `t` (необходимое количество операций называется редакционным расстоянием)
- При этом, в процессе нужно получить максимальное число w_i

Решение, Редакционное расстояние

- Научимся находить редакционное расстояние между строками `a` и `b`
- Это стандартная задача, решающаяся методом динамического программирования
- Состояние `dp[i][j]` соответствует префиксу строки `a` длины `i` и префиксу строки `b` длины `j`
- Значением в состоянии является редакционное расстояние между соответствующими префиксами `a` и `b`

Решение, Редакционное расстояние

- Научимся находить редакционное расстояние между строками a и b
- Для пересчета используются следующие переходы:
  - `dp[i][j]` ⟵ `dp[i - 1][j] + 1`
  - `dp[i][j]` ⟵ `dp[i][j - 1] + 1`
  - `dp[i][j]` ⟵ `dp[i - 1][j - 1]`, если `a[i - 1] = b[j - 1]`
  - `dp[i][j]` ⟵ `dp[i - 1][j - 1] + 1`, иначе

Решение

- Используя решение, описанное на предыдущем слайде, можно найти редакционное расстояние между всеми парами строк за $O(L^2)$, где L – суммарная длина всех строк
- Обозначим ред. расстояние между `a` и `b` за `dist(a, b)`
- Строка w_i может встретиться в процессе преобразования если и только если
  $\mathrm{dist}(s, w_i) + \mathrm{dist}(w_i, t) = \mathrm{dist}(s, t)$
- Строки w_i и w_j могут встретиться в таком порядке в процессе преобразования если и только если
  $\mathrm{dist}(s, w_i) + \mathrm{dist}(w_i, w_j) + \mathrm{dist}(w_j, t) = \mathrm{dist}(s, t)$

Решение

- Построим ориентированный граф
- Вершины – строки
- Проведем ребро из `s` в `t`
- Проведём ребро из вершины `s` в вершину w_i и из w_i в `t`, если w_i может получиться в процессе преобразования `s` в `t`
- Проведём ребро из w_i в w_j, если эти две строки могут встретиться друг за другом в процессе преобразования
- В данном графе нужно найти путь из `s` в `t` с максимальным количеством внутренних вершин
- Граф ацикличный

Решение, самый длинный путь

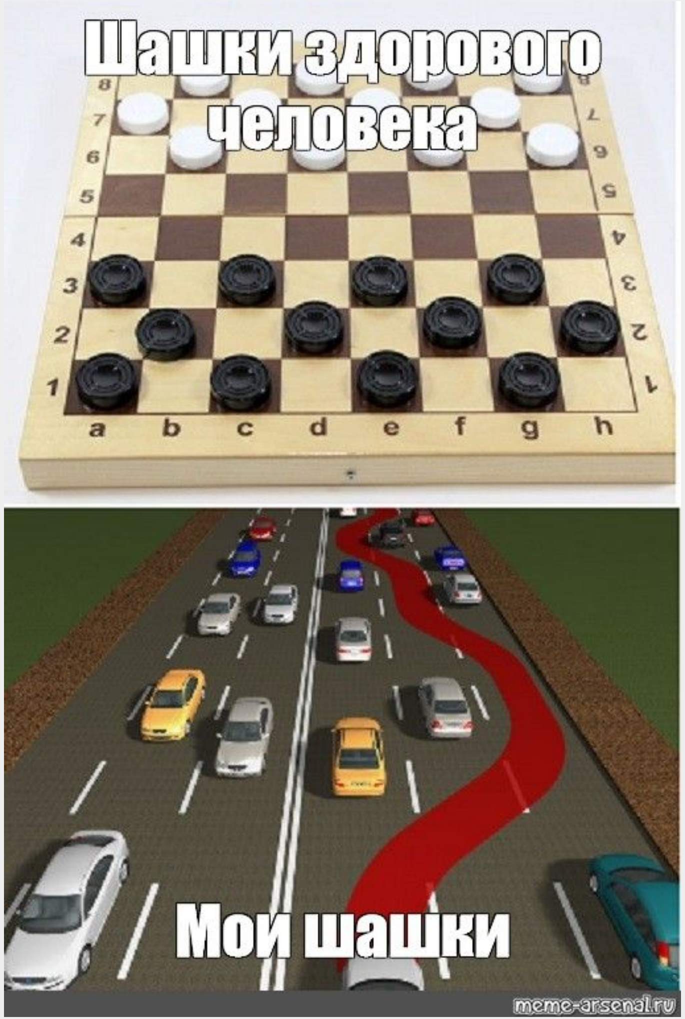
- Надо найти самый длинный путь из `s` в t в ацикличном графе
- Сделаем топологическую сортировку графа
- Снова воспользуемся методом динамического программирования
- Значение в состоянии `dp[v]` – максимальная длина пути из вершины v в вершину `t`
- Переходы:
    - `dp[v] ⟵ 0`, если `v = t`
    - `dp[v] ⟵ dp[to] + 1`, для всех ребер `(v, to)`

Время работы

- Получившееся решение работает за $O(L^2 + n^2)$

Задача K
Шашки

Автор задачи:
Даниил Орешников
Разработка задачи:
Виктор Шарепо



Постановка задачи

- Есть a белых и b черных шашек
- Нужно построить из них башенку, поставив друг на друга в любом порядке
- Определить максимальное число черных полос, которое может получиться на башенке

Решение задачи

- Ответ не больше b
- Когда ответ равен b?
  - Оптимально: чередовать черные и белые шашки, чтобы тратить на каждую черную полосу только одну черную шашку
  - Минимальное число белых шашек необходимое, чтобы получить b черных полос, равно $b - 1$
- Таким образом:
  - Если $a \geq b - 1$, то ответ равен b
  - Иначе ответ равен $a + 1$, так как большего числа полос добиться не получится

Задача L
Магниты

Автор задачи:
Семён Степанов
Разработка задачи:
Семён Степанов,
Максим Кузин

Постановка задачи

- Есть n магнитов, изначально расположенных на прямой Ox
- Приходят запросы двух типов: повернуть отрезок магнитов по часовой или против часовой стрелки, и узнать координаты какого-то магнита
- Гарантируется, что во время поворота магниты образуют непрерывный горизонтальный или вертикальный отрезок

Решение задачи

- Научимся быстро поворачивать отрезок. Для этого рассмотрим, как изменятся координаты точек (x_l, y_l), (x_{l+1}, y_{l+1}), $\ldots$, (x_r, y_r), образующих горизонтальный отрезок
- Точка (x_i, y_i) переместится в точку $(x_i - (x_i - x_l), y_i + (x_i - x_l))$
- Отрезок непрерывен, значит $x_i - x_l = i - l$
  - для вертикального отрезка $y_i - y_l = i - l$
- То есть точка (x_i, y_i) переместится в точку $(x_i - (i - l), y_i + (i - l))$
- Поворот свелся к прибавлению арифметической прогрессии

Решение задачи

- Задачу о прибавлении арифметической прогрессии на отрезке можно решать с помощью дерева отрезков, храня в вершине
  - `add` – суммарное число, прибавленное к первому элементу отрезка
  - `count` – сумму разностей прибавленных прогрессий

- Тогда `i`-й элемент изменился ровно на сумму значений `add + count · (i - left)` на пути в ДО от корня до листа `i`
- Время работы `O(q log n)`

```
printf("%s\n", "Спасибо за внимание")
```

Материалы олимпиады

http://nerc.itmo.ru/school