Заключительный этап X Открытой олимпиады по программированию

Разбор задач

Задача «Хранители»

Автор идеи: Павлов Илья
Разработчик: Павлов Илья

Формальная постановка

- Дано n точек на плоскости
- Требуется посчитать количество пар точек, таких что евклидово расстояние между ними совпадает с манхэттенским

Решение на 50 баллов

- В данной группе ограничения позволяют перебрать все пары точек
- Для каждой пары точек считаем оба вида расстояние между ними
- Чтобы избежать вещественной арифметики, сравниваем квадраты расстояний
- Решение за $O(n^2)$, набирает 50 баллов

Решение на 100 баллов

- $(dist_{манх})^2 = (|x1 - x2| + |y1 - y2|)^2 =$
  $(x1 - x2)^2 + 2|x1 - x2||y1 - y2| + (y1 - y2)^2$
- $(dist_{евк})^2 = (x1 - x2)^2 + (y1 - y2)^2$
- Первое расстояние совпадает со вторым только если точки на одной вертикали или на одной горизонтали
- Посчитаем для вертикальных и горизонтальных прямых количество пар точек на каждой из них
- Не забудем вычесть один раз пары совпадающих точек
- Координаты большие, пользуемся сортировкой / std::map
- Решение за *O(n log n)*, набирает 100 баллов.

Задача «Сжатие таблицы»

Автор идеи: Михаил Ипатов
Разработчик: Тимур Исхаков

Формальная постановка

- Дана таблица, заполненная положительными числами.
- Требуется получить таблицу, в которой будут сохранены отношения порядка в строках и столбцах.
- При этом требовалось минимизировать значение максимального элемента полученной таблицы.

Решение на 10 баллов ($n \le 1\,000$, $m = 1$)

- Таблица состоит из одного столбца.

- Отсортируем пары (значение, индекс).
- Аккуратно получим требуемую матрицу (не забудем про равенство).

- Решение за $O(n^2)$ или $O(n \log n)$ времени.

Решение на 15 баллов ($n, m \le 100$, $a_{i,j}$ различны)

- Построим граф зависимостей:

- Вершины — все элементы, т. е. пары (i, j).
- Для пары элементов в одной строке или столбце, если один элемент меньше другого, проводим ребро.

- Итого имеем nm вершин и $nm(n + m)$ рёбер.

Решение на 15 баллов (продолжение)

Дальше нам нужно найти «сжатие».

Один из способов:

- Заведём счётчик входящих рёбер в вершину
- Запустим bfs для вершин со значением счётчика 0
- Обрабатывая вершину, уменьшаем счётчики соседей
- Если у вершины обнулился счётчик, добавляем её на следующий уровень

Решение за $O(nm(n+m))$ времени и $O(nm(n+m))$ памяти.

Решение на 40 баллов ($n, m \le 100$)

- Победим равные значения — сожмём компоненты равенства в одну вершину. Эта же идея работает для всех ограничений.
- Для этого можно использовать dfs или СНМ.
- Решение за $O(nm(n+m))$ времени и $O(nm(n+m))$.

Решение на 70 баллов ($n, m \le 400$)

- Предыдущее решение использует слишком много памяти.
- Заметим, что нам не нужно хранить все $nm(n+m)$ рёбер в памяти, просто, находясь в вершине (i, j), пойдём в вершины (i, y) и (x, j).
- Для случая неравных элементов можно хранить список вершин, принадлежащих компоненте.

- Решение за $O(nm(n+m))$ времени и $O(nm)$ памяти.

Решение на 100 баллов ($nm \le 1\ 000\ 000$)

- Заметим, что нам не нужны все рёбра в строках и столбцах.
- Отсортируем значения в строках и столбцах и проведём рёбра между вершинами с соседними значениями.

- Для сжатия рёбра равенства найдём тем же способом.

- Решение за $O(nm\log(nm))$ времени, $O(nm)$ памяти.

Задача «Канатная дорога»

Автор идеи: Михаил Ипатов
Разработчик: Михаил Ипатов

Формальная постановка

- Дана последовательность h_i из n чисел
- Нужно ответить на q запросов: если a_j-е число заменить на b_j, какая будет длина наибольшей возрастающей подпоследовательности в h

Решение на 10–40 баллов

- Каждый раз будем заново пересчитывать НВП
- В зависимости от использованного алгоритма поиска НВП, решение набирает различное число баллов
- Перебор подмножеств за $O(N \cdot 2^N)$ на запрос – 10 баллов
- Динамика за $O(N^2)$ на запрос – 20 баллов
- Динамика за $O(N \log N)$ на запрос – 40 баллов

Решение на 60 баллов

- Поймём, чему может равняться ответ на запрос
- Обозначим за *x* длину НВП в исходной последовательности
- Ответ может быть равен:
  - *x+1*, если изменённое число "встроится" в новую НВП
  - *x*, по умолчанию
  - *x-1,* если все НВП проходят через наше число и они все "испортятся"

Важная идея

- Поймём, чему может равняться ответ на запрос
- Обозначим за *x* длину НВП в исходной последовательности
- Ответ может быть равен:
  - *x+1*, если изменённое число "встроится" в новую НВП
  - *x*, по умолчанию
  - *x-1,* если все НВП проходят через наше число и они все "испортятся"

Важная идея

- Общая схема любого решения
  - Считаем длину НВП, проходящей через новый изменённый элемент
  - Если она равна *x+1*, ответ — *x+1*
  - Определяем, лежал ли исходный элемент на **всех** НВП
  - Если нет, то хотя бы одна из них осталась, и ответ — *x*
  - Иначе ответ — *x-1*

Решение на 60 баллов

- Для каждого числа посчитаем наибольшую длину возрастающей подпоследовательности, начинающейся и заканчивающейся в нём, а также количество таких и таких последовательностей
- В частности, мы знаем длину и количество возрастающих последовательностей во всей последовательности

Решение на 60 баллов

Как проверить, что числов входит во все НВП в исходном массиве:

- Найдем количество НВП, содержащих это число
- Если через него не проходит НВП *(l[i].length+r[i].length-1 < lis.length)*, то их 0
- Иначе их *l[i].count* · *r[i].count*
- Если их число равно lis.count, то число принадлежит всем НВП
- Вероятность ложного совпадения по модулю мала

Решение на 60 баллов

- За *y* обозначим наибольшую длину возрастающей подпоследовательности, содержащей a_i-е число(в обновленном массиве)
- Тогда $y = max(l[i].length: i < a_i, h_i < b_i) + 1 + max(r[i].length: i > a_i, h_i > b_i)$
- Решение работает за $O(N^2 + MN)$ и набирает 60 баллов.

Решение на 80 баллов

- Сожмем числа в исходном массиве.
- Считаем l и r с помощью дерева отрезков
- Находить *y* для запроса можно с помощью двумерного дерева отрезков

Сложность решения — $O(N\log N + M\log^2 N)$, 80+ баллов

Решение на 100 баллов

- Вместо двумерного дерева можно отвечать на запросы offline в два прохода (сначала для всех запросов посчитаем длину НВП слева от измененного числа, а затем справа)
- O(NlogN + MlogN), 100 баллов

Альтернативное решение на 100 баллов

- Не будем считать l[i].count и r[i].count
- Заметим, что для любого числа его позиция во всех НВП одинакова
- Чтобы проверить входит ли число во все НВП, достаточно проверить, что нет другого числа с такой же позицией в НВП
- Ещё решение за $O(N\log N + M\log N)$, 100 баллов

Задача «Часовой механизм»

Автор идеи: Максим Ахмедов
Разработчик: Максим Ахмедов

Формальная постановка

- Дано два дерева *T1* и *T2* на одном и том же наборе вершин
- За один ход разрешается удалить ребро из дерева и добавить новое так, чтобы снова образовалось дерево
- Перейти от первого дерева ко второму за минимальное число шагов

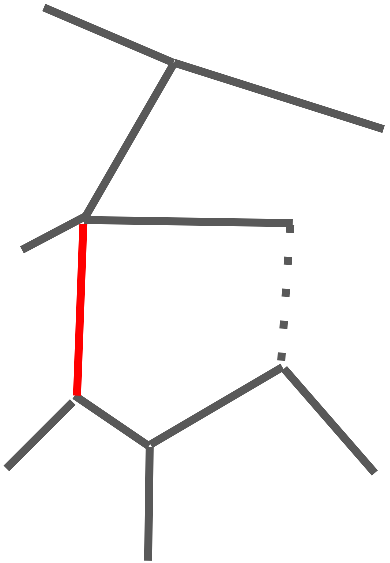
Решение на 20 баллов

- Гарантируется, что можно обойтись не более, чем одним шагом
- Если деревья совпадают, делать ничего не надо
- Иначе нужно найти единственное ребро, которое есть в T1 но не в T2 и единственное ребро, которое есть в T2 но не в T1, и заменить первое на второе

Сложность решения — какая угодно, 20 баллов

Важная идея

- Обозначим за *T1 \ T2* множество рёбер, которые есть в T1, но не в T2 (очевидно, |T1 \ T2| = |T2 \ T1|)
- Никакое ребро из пересечения T1 и T2 трогать не выгодно
- Любое ребро из T1 \ T2 можно выкинуть и заменить на что-нибудь (иначе по T2 две образованные компоненты будут не связны)



Решение на 40 баллов

- Пока деревья не совпадают, повторяем следующее
- Выберем любое ребро *e* из *T1 \ T2*
- Переберём ребро *f* из T2 \ T1
- Проверим, можно ли заменить *e* на *f*, то есть, будет ли образованный граф деревом
- Если да, то делаем такую операцию
- Решение за $O(n^3)$

Решение на 60 баллов

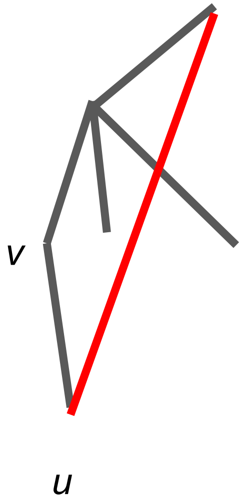
- Вместо того, чтобы перебирать *f*, просто удалим *e* из *T1* и посмотрим, что получится
- По сказанному ранее, должно существовать какое-то ребро *f*, концы которого лежат в разных компонентах
- Найдём его за линейное время и сделаем одну операцию
- Сложность решения — $O(n^2)$

Решение на 80–100 баллов

- Посмотрим на любой лист v из первого дерева, соединённый с вершиной u
- Возможно два варианта
- Во-первых, ребро (u, v) присутствует во втором дереве — тогда можно "склеить" u и v, стянув оба дерева по этому ребру
- Сделаем эту операцию с помощью СНМ, аккуратно объединив списки рёбер, исходящих из вершин (за $O(1)$)

Решение на 80–100 баллов

- Во-вторых, ребра *(u, v)* может не быть во втором дереве — тогда его по той же логике можно заменить на любое ребро, ведущее из *u* во втором дереве, после чего это ребро также надо стянуть
- Получается решение за *O(n log n)* или *O (n alpha)* в зависимости от аккуратности написания и структур для поддержания списков рёбер



Задача «Чемпионат Юпитера»

Автор идеи: Евстропов Глеб
Разработчик: Саакян Вильям

Формальная постановка

- Дано n пар различных цветов от 1 до m — цвета футболок команд.
- Судья может провести матч 2 команд, если можно выбрать попарно различные цвета футболок команд и судьи.
- Требуется найти минимальное количество футболок, чтобы судья смог провести любой матч.

Решение на 20 баллов

- Переберём подмножество цветов, которое будет у судьи, и промоделируем все возможные матчи
- Решение за, например, $O(m * 2^m * n^2)$, набирает 20 баллов.

Главная идея

- **Всегда можно обойтись 3 футболками**
- Надо проверить, можно ли провести турнир, используя один или два цвета. Если нет, то вывести "1 2 3", если цветов хотя бы 3, иначе вывести -1

Решение за 40 баллов

- Переберём все способы выбрать одну или две футболки, смоделировать все матчи
- Решение за $O(m^2 * n^2)$, набирает 40 баллов.

Следующая идея

- Если есть две команды, которые привезли одинаковый набор футболок *(a, b),* то судья не может ограничиться набором из одной из этих футболок, или ровно из этих двух футболок
- В любом другом матче, имея даже всего-лишь одну футболку, судья сможет выбрать командам подходящие цвета

Следующая идея

- Если есть две команды, которые привезли одинаковый набор футболок *(a, b),* то судья не может ограничиться набором из одной из этих футболок, или ровно из этих двух футболок
- В любом другом матче, имея даже всего-лишь одну футболку, судья сможет выбрать командам подходящие цвета

Решение на 60 баллов

- Выпишем все "запрещённые пары", то есть такие, которые встречаются минимум у двух команд
- Проверим, есть ли число, не встречающееся ни в одной такой паре => выводим этот цвет
- Проверим, есть ли не запрещённая пара => выводим эту пару
- Иначе выводим либо "1 2 3", либо "-1", если цветов меньше трёх
- Решение за $O(n^2)$

Решение на 100 баллов

- Аккуратно сжимаем цвета
- Сортируем все запрещённые пары лексикографически
- Ищем не запрещённую пару, просто перебирая её в порядке лексикографического возрастания
- Решение за $O(n \log n)$

Задача «Злая Лига Зла»

Автор идеи: Михаил Пядеркин
Разработчик: Андрей Гаркавый

Формальная постановка

Заменить в строке из знаков «?», «(», «)» знаки вопроса на скобки так, чтобы в получившейся строке была наибольшая правильная скобочная подпоследовательность.

Решение на 10 баллов (перебор)

- Если нет вопросительных знаков, то можно найти длину наибольшей правильной скобочной подпоследовательности за $O(n)$.

Как? Пройдемся и будем считать баланс, но запретим уходить в минус.

Тогда просто переберем, на что заменить знаки вопроса.

Это решение за $\mathbf{O(n2^n)}$.

Решение на 30 баллов (динамика)

Введем d[i][b] — длину максимальной правильной скобочной подпоследовательности для i-го префикса с лишними b открывающимися скобками.

Заметим, что d[i][b] легко обновляется через предыдущие значения.

Заметим, что ответ по массиву d легко восстанавливается.

Это решение за **$O(n^2)$** времени и **$O(n^2)$** памяти.

Решение на 50 баллов (линейная память)

- Заметим, что есть ответ, в котором скобки, бывшие знаками вопроса, должны идти в таком порядке: сначала открывающие, потом закрывающие.

Почему? Ответ не уменьшается при замене «)...(» на «(...)»

Переберем все эти варианты за линейное время и каждый из них проверим.

Это решение за $O(n^2)$ времени и $O(n)$ памяти.

Решение на 70 баллов (тернарный поиск)

- Если заменить все знаки вопроса на «)», а потом идти слева направо и менять эти же символы на «(», то ответ сначала строго возрастает, потом возможно один раз не меняется, а потом строго убывает.

Почему? Посмотрим на скобки, не вошедшие в ответ, они идут в таком порядке: сначала «)», потом «(». Для доказательства нужно рассмотреть случаи, между какими из них находится скобка, которую мы сейчас меняем и посмотреть, как меняется ответ.

Решение на 70 баллов (продолжение)

Поэтому можно найти максимум тернарным поиском. Или даже бинарным!

Это решение за **$O(n \log n)$** времени.

Решение на 100 баллов

Посчитаем баланс и ответ на каждом префиксе, если считать знаки вопроса «(». И посчитаем “обратный баланс” и ответ на каждом суффиксе, если считать знаки вопроса «)».

Тогда найдем дополняющие друг друга префикс и суффикс, т. ч. сумма ответов на них и минимума из двух балансов — минимальна.

До этого места будем менять знаки на «(», а после — на «)».

Это решение за **O(n)** времени.

Задача «Фото от пилота»

Автор идеи: Роман Андреев
Разработчик: Дмитрий Горбунов

Формальная постановка

- Даны N прямых на плоскости. Рассмотрим множество пересечений этих прямых.
- Q запросов: на каком расстоянии от данной прямой находится ближайшее к ней пересечение.

10 баллов: $O(TN^2)$, запросы горизонтальные

- Переберем все пары прямых, найдем их точку пересечения.
- Формула пересечения прямых $ax + by = c$ и $ux + vy = w$:
  - $x = (cv - wb) / (av - bu)$, $y = (aw - cu) / (av - bu)$.
- Заметим, что поскольку запросы горизонтальные, то расстояние от точки до прямой = разница y-координат.
- Обновим ответ.
- Поскольку всего пар прямых O(N^2), а пересечение прямых и обновление ответа делаются за O(1), то итоговое решение имеет асимптотику O(TN^2).

30 баллов: $O(TN^2)$

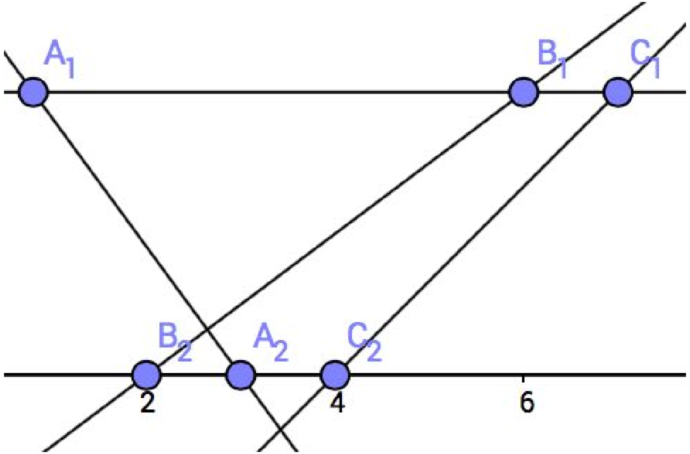
- Все то же самое, но расстояние от точки до прямой нужно уметь считать для произвольной прямой.
- Формула для расстояния от прямой ax + by + c = 0 до точки (x, y):
  - dist = |ax + by + c| / sqrt(a^2 + b^2).
- Асимптотика все еще O(TN^2).

60 баллов: $O(TN \log N \log C)$

- Новая идея: сделаем бинпоиск по ответу. Пусть ближайшая точка пересечения находится на расстоянии не более d.
- Рассмотрим две прямые, параллельные запросу и находящиеся на расстоянии d от него.
- Нам интересно, есть ли пересечение прямых внутри полосы между этими прямыми.

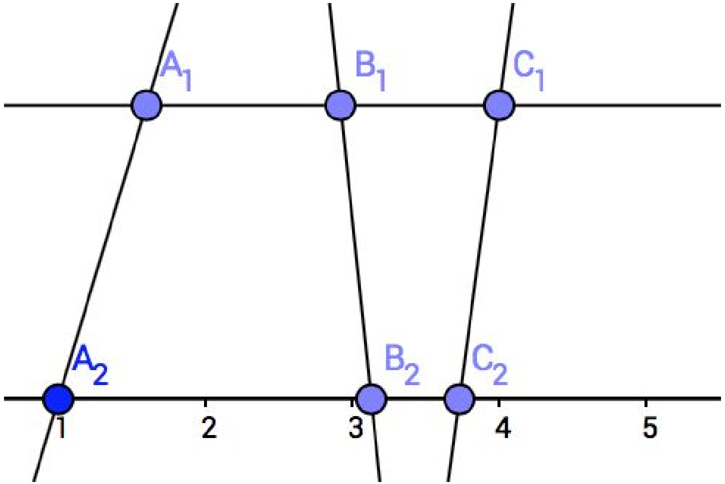
60 баллов: O(TN log N log C)

- Пример: сверху точки идут в порядке ABC, а снизу - BAC, значит, есть пересечение.



60 баллов: O(TN log N log C)

- Еще пример: и сверху, и снизу пересечения идут в порядке ABC, пересечений нет.



60 баллов: $O(TN \log N \log C)$

- Итого: делаем бинпоиск по ответу. Пусть ответ меньше или равен d.
- Возьмем две прямые, параллельные запросу и находящиеся на расстоянии d от запроса.
- Пересечем с этими двумя прямые все прямые, отсортируем точки на них.
- Проверим, что перестановки, образованные номерами прямых, которые дают данную точку пересечения совпадают.
- Поскольку сортировать точки мы можем за $O(N \log N)$, суммарная асимптотика - $O(TN \log N \log C)$.

100 баллов: O(TN log N)

- Рассмотрим все точки пересечения запроса со всеми прямыми.
- Отсортируем их вдоль прямой.
- Можно доказать, что ближайшее пересечение можно искать среди пересечений между соседними в таком порядке прямыми.
- Поскольку сортировать точки на прямой можно за O(N log N), получилось решение за O(TN log N).

Задача «Бэтмен возвращается»

Автор идеи: Глеб Евстропов + Степан Каргальцев
Разработчик: Владислав Епифанов

Формальная постановка

- Даны последовательность из N целых чисел.
- M запросов: на заданном подотрезке входной последовательности найти наиболее удаленную пару чисел такую, что левое число меньше правого.

40 баллов: $O(N^3)$

Для каждого запроса перебираем все пары чисел попавшие внутрь отрезка и выбираем наилучшую.

50 баллов: $O(N^2 \log N)$

Для каждого запроса перебираем левое из чисел, а самое удаленное правое будем находить с помощью спуска по дереву отрезков.

Запросы будем обрабатывать, например, в порядке уменьшения правой границы и поддерживать минус бесконечности во всех позициях правее границы текущего отрезка.

60 баллов: $O(N^2)$

Ответ для подотрезка [i, j] это либо сама пара (i, j) либо лучший из ответов для подотрезков [i+1, j] или [i, j-1].

Таким образом, перебирая подотрезки в порядке увеличения длины можно посчитать ответы для всех запросов за квадратичное время и квадратичную память.

Если при этом хранить только две диагонали этой динамики и отвечать на запросы сразу, как для них посчитан ответ, то можно обойтись линейной памятью.

100 баллов: $O(N^{1.5})$

Научимся решать задачу для фиксированной левой границы и всех возможных правых границ за суммарно линейное время.

Будем двигать правую границу и поддерживать текущий оптимальный ответ, а также минимум на каждом префиксе, который мы прошли.

Пусть ответ для отрезка (L, R-1) это пара чисел на позициях (p, q). Научимся быстро находить ответ для отрезка (L, R). Посмотрим на минимум на префиксе (L, R-(q-p)-1). Если он меньше, чем число на позиции R, то можно обновить ответ используя это число, и число на позиции R. Будем делать это, пока ответ улучшается.

100 баллов: $O(N^{1.5})$

Так как одно улучшение ответа происходит за O(1) и при каждом улучшении разность чисел в ответе увеличивается, то суммарно все улучшения будут выполнены за O(N).

Теперь разобьем всю последовательность на куски длины корень из N. Каждый запрос разобьем на три части: от левой границы корневого куска до правой границы запроса, от левой границы запроса до правой границы корневого куска и все остальное.

Первые два типа мы умеем отвечать за O(N) для каждой левой и правой границы корневого куска (таких границ корень из N).

100 баллов: $O(N^{1.5})$

Для каждого запроса осталось перебрать пары, где левое число лежит среди первых корень из N позиций, а правое среди последних корень из N позиций, находящихся на отрезке запроса.

Для этого можно склеить левую и правую часть, и использовать на ней ту же самую идею, используя в качестве номеров элементов их номера в исходной последовательности.

СПАСИБО ЗА ВНИМАНИЕ!