Всероссийская олимпиада школьников по информатике

Екатеринбург

6–12 апреля 2014 года

Автомат с игрушками

Автомат с игрушками

Автомат с игрушками

Автомат с игрушками

- Идея задачи — Глеб Евстропов
- Подготовка тестов — Сергей Мельников
- Разбор задачи — Павел Маврин

Постановка задачи

- Задан игровой автомат в виде подвешенного дерева, в узлах которого находятся игрушки, для каждого ребра задана его ширина.
- Монетки падают от корня вниз по дереву, каждый раз выбирая более широкое ребро.
- После прохождения монетки ширина ребра уменьшается на 1.
- Требуется найти число монет, необходимых, чтобы получить игрушку в заданном узле.

Решение на 50 баллов. Эмуляция.

- Построим траекторию одной монетки.
- Если траектория проходит через нужный узел, выходим.
- Вычитаем 1 из ширин всех ребер, через которые прошла монета.
- Если оба ребра из корня стали ширины 0, то ответ -1.

Оценка времени работы

- Максимально возможное число монет: $w_1 + u_1$.
- Каждая монета обрабатывается за $O(n)$.
- Время работы: $O(wn)$.

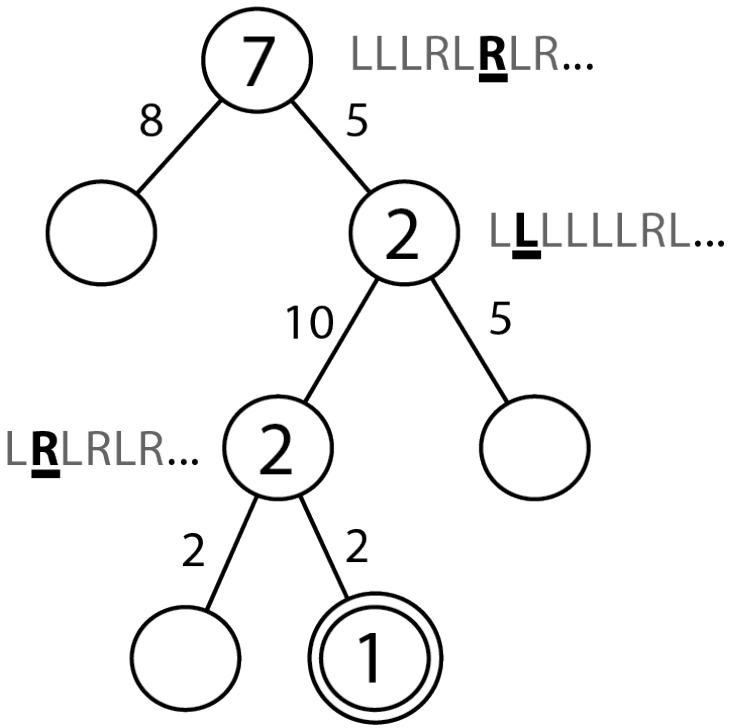
Решение на 100 баллов.

- Будем двигаться от узла с нужной игрушкой вверх по дереву, пересчитывая требуемое число монет.
- Для последнего узла это число равно 1.
- При переходе вверх на один уровень число увеличивается, в зависимости от ширин ребер.

Пересчет числа монет.

- Пока ширины не сравняются, монеты падают в одну сторону, потом по очереди то влево, то вправо.
- LLLLLLLLLLLLLLRLRLRLRLRLRLRLRLRL...
- Пусть нужно забросить s монет в нужное поддерево.
- Нужно найти в последовательности L или R с номером s.

Пример.



До уравнивания 2 упадут налево, после этого нужно 2 направо, ответ $3 + 2 \cdot 2 = 7$

Первые 5 упадут налево, этого хватит, ответ 2

Ширины уже равны, ответ $2 \cdot 1 = 2$

Базовый уровень. Ответ 1.

Оценка времени работы

- Пересчет числа монет делается за $O(1)$.
- Итоговая сложность $O(n)$.

Вопросы?

Задача «Робинзон и крокодилы»

Задача «Робинзон и крокодилы»

Задача «Робинзон и крокодилы»

Задача «Робинзон и крокодилы»

- Идея задачи — Максим Ахмедов
- Подготовка тестов — Ольга Соболева
- Разбор задачи — Александр Кленин

Постановка задачи

- Есть прямоугольное поле $n \times m$, в каждой клетке которого находится не более одного крокодила. Каждый крокодил смотрит в одном из четырёх направлений (север, юг, запад, восток) и может двигаться только в этом направлении. Крокодил может убежать с поля, если на его пути нет других крокодилов.
- Требуется определить, какое максимальное количество крокодилов может убежать.

Постановка задачи

- Есть прямоугольное поле $n \times m$, в каждой клетке которого находится не более одного крокодила. Каждый крокодил смотрит в одном из четырёх направлений (север, юг, запад, восток) и может двигаться только в этом направлении. Крокодил может убежать с поля, если на его пути нет других крокодилов.
- Требуется определить, какое максимальное количество крокодилов может убежать.

Решение

- Заметим, что если в некоторый момент времени может убежать несколько крокодилов, то порядок, в котором они убегают, не имеет значения.
- Смоделируем процесс. Будем по очереди удалять крокодилов, которые могут убежать.

Решение

- Заметим, что если в некоторый момент времени может убежать несколько крокодилов, то порядок, в котором они убегают, не имеет значения.
- Смоделируем процесс. Будем по очереди удалять крокодилов, которые могут убежать.

Решение

- Найдем множество крокодилов, которые изначально могут убежать.
- Изначально убежать могут только крокодилы, которые являются крайними в строке или столбце и смотрят в правильном направлении. Их можно найти за время $O(nm)$.

Решение

- Найдем множество крокодилов, которые изначально могут убежать.
- Изначально убежать могут только крокодилы, которые являются крайними в строке или столбце и смотрят в правильном направлении. Их можно найти за время $O(nm)$.

Решение

- Будем последовательно удалять крокодилов из найденного множества.
- Когда крокодил убегает, следующий за ним (в строке или столбце) получает шанс убежать и может попасть в наше множество.

Решение

- Будем последовательно удалять крокодилов из найденного множества.
- Когда крокодил убегает, следующий за ним (в строке или столбце) получает шанс убежать и может попасть в наше множество.

Решение

- Для каждого крокодила изначально запомним всех его «соседей» — крокодилов, стоящих на той же строке или в том же столбце через какое-то (возможно, нулевое) количество пустых клеток.
- Для хранения множества будем использовать двусвязный список.
- Тогда удаление крокодила из множества и добавление его соседей работает за константное время.

Решение

- Для каждого крокодила изначально запомним всех его «соседей» — крокодилов, стоящих на той же строке или в том же столбце через какое-то (возможно, нулевое) количество пустых клеток.
- Для хранения множества будем использовать двусвязный список.
- Тогда удаление крокодила из множества и добавление его соседей работает за константное время.

Решение

- Для каждого крокодила изначально запомним всех его «соседей» — крокодилов, стоящих на той же строке или в том же столбце через какое-то (возможно, нулевое) количество пустых клеток.
- Для хранения множества будем использовать двусвязный список.
- Тогда удаление крокодила из множества и добавление его соседей работает за константное время.

Сложность

- Сложность работы этого алгоритма $O(nm)$

Частичные решения

- Будем перебирать крокодилов, проверять, могут ли они убежать, и удалять убежавших.
- Если ни один не смог убежать, останавливаемся.
- Такое решение работает за $O(n^2m^2(n+m))$ и проходит первую группу тестов.

Частичные решения

- Будем перебирать крокодилов, проверять, могут ли они убежать, и удалять убежавших.
- Если ни один не смог убежать, останавливаемся.
- Такое решение работает за $O(n^2m^2(n+m))$ и проходит первую группу тестов.

Частичные решения

- Будем перебирать крокодилов, проверять, могут ли они убежать, и удалять убежавших.
- Если ни один не смог убежать, останавливаемся.
- Такое решение работает за $O(n^2m^2(n+m))$ и проходит первую группу тестов.

Частичные решения

- Построим ориентированный граф, вершинами которого будут крокодилы.
- Если один крокодил мешает другому, то соединим их ребром.
- Осталось определить количество вершин, из которых достижим какой-нибудь цикл.
- Такое решение работает за $O(nm(n+m))$ времени и памяти. Проходит вторую группу тестов.

Частичные решения

- Построим ориентированный граф, вершинами которого будут крокодилы.
- Если один крокодил мешает другому, то соединим их ребром.
- Осталось определить количество вершин, из которых достижим какой-нибудь цикл.
- Такое решение работает за $O(nm(n+m))$ времени и памяти. Проходит вторую группу тестов.

Частичные решения

- Построим ориентированный граф, вершинами которого будут крокодилы.
- Если один крокодил мешает другому, то соединим их ребром.
- Осталось определить количество вершин, из которых достижим какой-нибудь цикл.
- Такое решение работает за $O(nm(n+m))$ времени и памяти. Проходит вторую группу тестов.

Частичные решения

- Построим ориентированный граф, вершинами которого будут крокодилы.
- Если один крокодил мешает другому, то соединим их ребром.
- Осталось определить количество вершин, из которых достижим какой-нибудь цикл.
- Такое решение работает за $O(nm(n+m))$ времени и памяти. Проходит вторую группу тестов.

Вопросы?

Задача «Петя и Робот>

Задача «Петя и Робот»

Задача «Петя и Робот>

Задача «Петя и Робот»

- Идея задачи — Николай Ведерников
- Подготовка тестов — Нияз Нигматуллин, Никита Иоффе
- Разбор задачи — Никита Иоффе

Постановка задачи

- Была загадана перестановка
- Запросы участника — поменять два элемента перестановки на позициях i и j $\texttt{swap}(i, j)$
- Ответом на запрос является количество инверсий в получившейся перестановке
- Требуется восстановить исходную перестановку

Пример

- 3, 1, 2, 4, количество инверсий 2
- 3, 1, 2, 4
- 3, 4, 2, 1, количество инверсий 5

Решение на 30 баллов

- Пусть p — загаданная перестановка.
- С помощью двух операций `swap`(i, j) можно определить, меньше ли p_i, чем p_j
- Любой квадратичной сортировкой отсортируем загаданную перестановку
- Ответом является обратная к ней перестановка
- Количество запросов — $O(n^2)$
- Асимптотика решения — $O(n^2)$

Решение на 50 баллов

- Вместо квадратичной сортировки применим любую быструю сортировку
- Количество запросов — $O(n \log n)$
- Асимптотика решения — $O(n \log n)$

Таблица инверсий

- Таблица инверсий перестановки p — массив t, такой, что t_i равно числу элементов перестановки p стоящих правее, чем i и меньших p_i
- $3,\ 4,\ 1,\ 2$ — перестановка
- $2,\ 2,\ 0,\ 0$ — таблица инверсий

Оптимизация

- Каждая перестановка однозначно задаётся своей таблицей инверсий
- Восстановить перестановку по таблице инверсий — линейный проход с деревом отрезков
- Осталось научиться получать таблицу инверсий перестановки за $O(n)$ запросов

Более оптимальное решение

- За $2 \times n$ запросов найдём в перестановке максимум и поставим его на первую позицию.
- После этого построим таблицу инверсий перестановки.

Более оптимальное решение

- Для этого сделаем запрос `swap(1, `i`)` дважды
- Пусть количество инверсий после первого запроса — a, после второго — b
- Тогда количество инверсий для i–го элемента $inv_i = (b - a + 1)/2$
- Количество запросов — $O(4 \times n)$. Асимптотика решения — $O(n \log n)$
- Данное решение набирает 84 балла

Правильно решение

- Разобьём перестановку на две перестановки
- Первая перестановка — элементы, меньшие первого элемента перестановки
- Вторая перестановка — элементы, большие первого элемента перестановки

Правильно решение

- Рассмотрим i–й элемент исходной перестановки
- Сделаем запрос `swap(1, `i`)` дважды
- Пусть количество инверсий после первого запроса — a, после второго — b

Правильно решение

- Если $a > b$, то i–й элемент исходной перестановки добавляется в первую перестановку
- Количество инверсий у этого элемента в перестановке перестановке $(a - b - 1)/2$

Правильно решение

- Если $a < b$, то i–й элемент исходной перестановки добавляется во вторую перестановку
- Количество инверсий у этого элемента во второй перестановке $|right| - (b - a - 1)/2$
- Где $|right|$ — количество элементов во второй перестановке перед добавлением i–го

Правильно решение

- Отдельно восстанавливаем первую и вторую перестановки
- Восстанавливаем ответ

Вопросы?

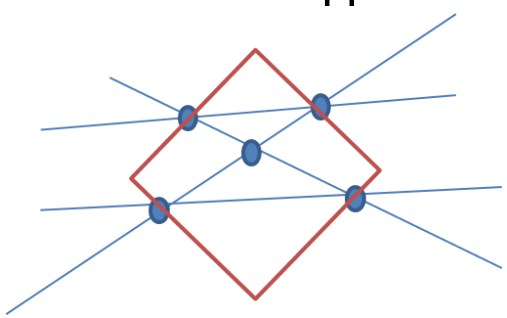
Задача «Коллайдер 2.0»

Задача «Коллайдер 2.0»

- Идея задачи — Глеб Евстропов
- Подготовка тестов — Глеб Евстропов, Михаил Пядеркин
- Разбор задачи — Михаил Пядеркин

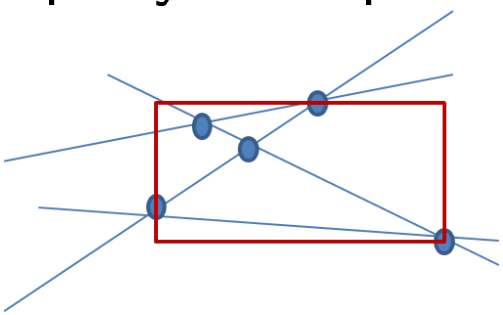
Постановка задачи

- В каждый момент времени на плоскости расположено некоторое множество прямых
- Поступают запросы двух типов:
  1. Добавить новую прямую в множество
  2. Рассмотрим все точки пересечения текущего множества прямых. Необходимо найти прямоугольник минимальной площади, содержащий все эти точки, и одна из сторон которого была бы параллельна заданному в запросе направлению.



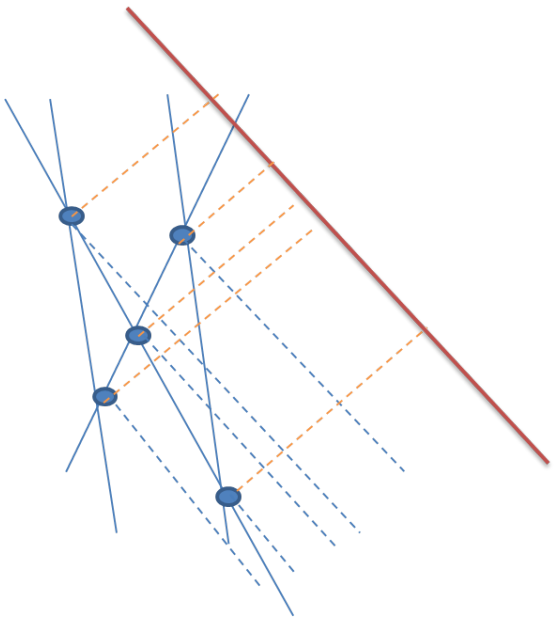
Простое решение

- Добавляя очередную прямую, будем искать точки пересечения с добавленными прямыми.
- В случае, если стороны прямоугольника параллельны осям координат, для ответа на запрос достаточно находить самую левую, правую, верхнюю и нижнюю точки.



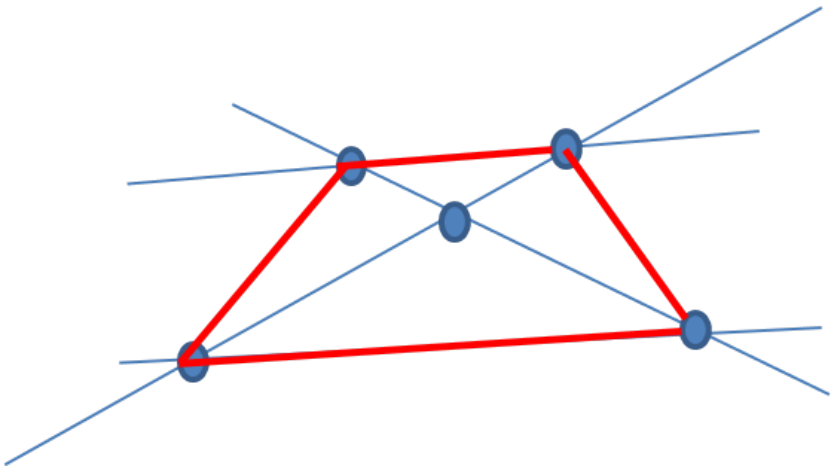
Простое решение

- В случае, если прямоугольник является произвольным, необходимо находить точки с экстремальной (минимальной или максимальной) проекцией на направление.



Основная идея

- Заметим, что для нахождения точек с экстремальной проекцией достаточно оставить лишь точки на выпуклой оболочке.

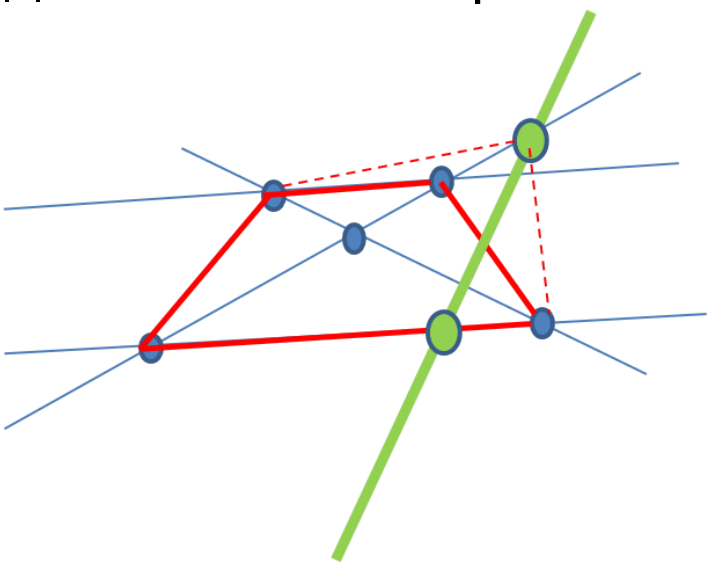


Основная идея

- Необходимо научиться быстро пересчитывать выпуклую оболочку точек пересечения прямых при добавлении прямой, а также быстро находить точки с экстремальной проекцией.
- Обратите внимание, что стороны выпуклой оболочки точек пересечения не обязательно являются отрезками исходных прямых.

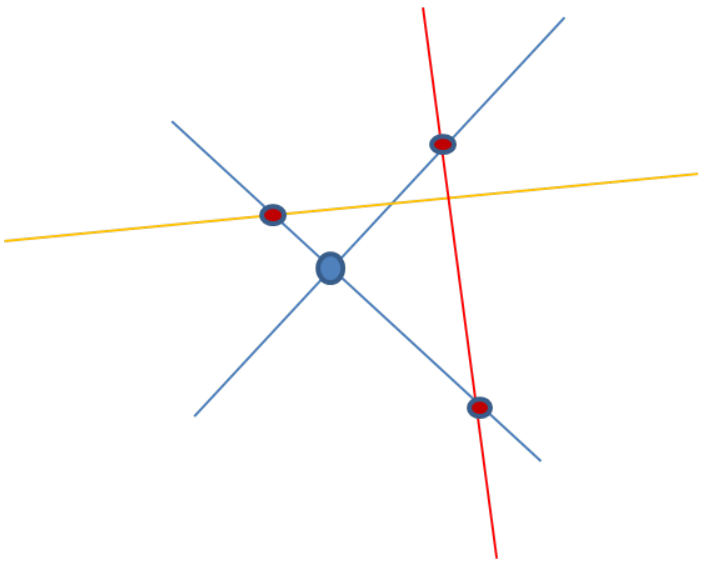
Геометрические соображения

- При добавлении очередной прямой в выпуклую оболочку достаточно найти крайние точки пересечения — остальные точки в нее войти не могут. И затем лишь эти две точки попробовать добавить.



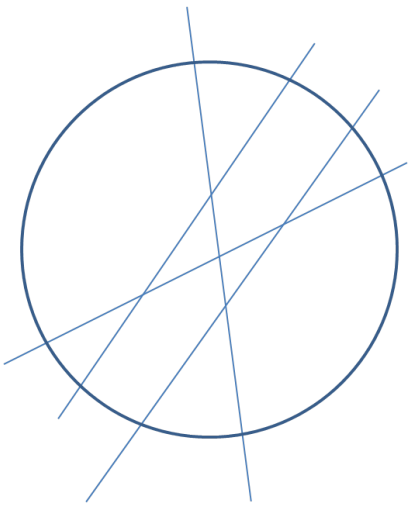
Геометрические соображения

- Предположим, что нас интересует выпуклая оболочка точек пересечения прямых. Не нужно пересекать все прямые со всеми, достаточно пересечь прямые, которые являются соседними по углу нормали.



Геометрические соображения

- В случае параллельных прямых, их необходимо корректно сортировать.

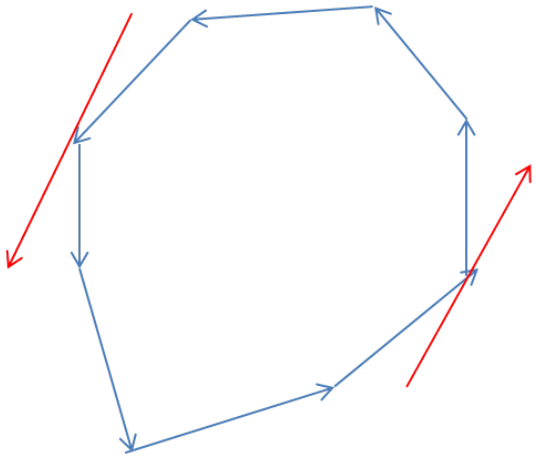


Техническая постановка задачи

- При добавлении очередной прямой необходимо найти соседние с ней по углу нормали прямые, пересечь с ними и получить две крайние точки.
- Эти две точки необходимо попробовать добавить в выпуклую оболочку.
- Для ответа на запрос необходимо найти точки с экстремальной проекцией на данное направление.

Поиск экстремальной проекции

- Так как в каждый момент мы имеем выпуклый многоугольник, то его стороны являются отсортированными по углу.
- Необходимо бинарным поиском найти данное направление (и противоположное) среди его сторон — это и будет экстремальная точка.

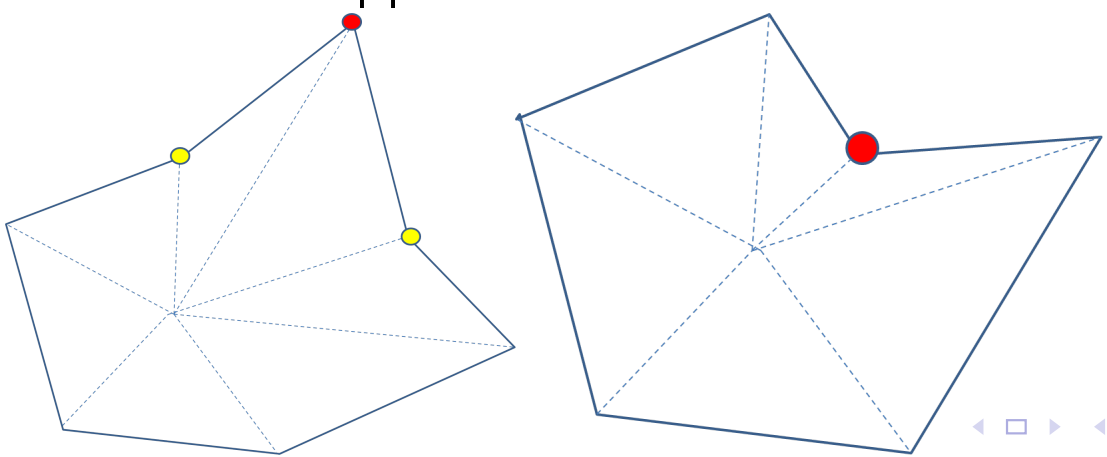


Поиск крайних точек на прямой

- В каждый момент мы можем поддерживать множество прямых, упорядоченных по углу.
- Для поиска соседней по углу прямой мы можем воспользоваться бинарным поиском.
- Для быстрой вставки можно использовать `<set>` (сбалансированное дерево).

Перестроение выпуклой оболочки

- Так как в каждый момент мы имеем выпуклый многоугольник, и точки лишь добавляются, то и его вершины являются отсортированными по полярному углу.
- Вставим точку в нужное место, и пока образуется невыпуклость, будем удалять точки по одной.



Итоговое решение

- Поддерживаются прямые, упорядоченные по полярному углу и по дополнительному параметру для параллельных прямых.
- Поддерживаются ребра выпуклой оболочки, упорядоченные по углу.
- Поддерживаются вершины выпуклой оболочки, упорядоченные по углу.
- Для ответа на запрос используется бинарный поиск.
- `<set>`, `lower_bound()`
- $(N + Q) \log N$ и 100 баллов

Вопросы?...

Задача «Киноакадемия»

Задача «Киноакадемия»

Задача «Киноакадемия»

Задача «Киноакадемия»

- Идея задачи — Михаил Пядёркин
- Подготовка тестов — Андрей Гейн, Михаил Пядёркин
- Разбор задачи — Андрей Гейн

Математическая формулировка

- Входные данные: число n; n троек целых чисел a_i, b_i, c_i.
- Результат: два индекса i и j $(i \neq j)$ таких, что

$$b_i + c_j + \sum_{k \neq i, k \neq j} a_k$$

максимально.

Решение: подзадача 1

- Переберём все возможные варианты индексов i и j $(i \neq j)$ за $O(n^2)$.
- Для каждой пары посчитаем суммарный уровень ликования.
- Время: $O(N^3)$. Проходит тесты для $N \leqslant 100$, набирает 20 баллов.

Решение: подзадача 1

- Лишние действия: каждый раз считается практически вся сумма $\sum a_k$.

Решение: подзадача 2

- Заметим, что максимизируемая сумма равна

$$b_i + c_j + \sum_{k \neq i, k \neq j} a_k = b_i + c_j + \sum_k a_k - a_i - a_j =$$

$$= (b_i - a_i) + (c_j - a_j) + \sum_k a_k$$

- Сумма всех a_k постоянна, поэтому нам нужно найти два индекса i и j $(i \neq j)$, для которых максимальна сумма $(b_i - a_i) + (c_j - a_j)$.

Решение: подзадача 2

- Такую пару индексов можно найти простым перебором.
- Время: $O(N^2)$. Проходит тесты для $N \leqslant 2000$, набирает 45 баллов.

Решение: подзадача 3

- Посчитаем заранее $b'_i = b_i - a_i$ и $c'_j = c_j - a_j$ для всех возможных i и j за $O(n)$.
- Если последовательности b' и c' достигают своих максимумов при $i = m_1$ и $j = m_2$ соответственно, причём $m_1 \neq m_2$, то пара индексов m_1 и m_2 является ответом.
- Если нет, то выберем максимум из

$$b'_{m_1} + \max_{k \neq m_1} c'_k$$

$$\max_{k \neq m_2} b'_k + c'_{m_2}$$

Решение: подзадача 3

- Посчитаем заранее $b'_i = b_i - a_i$ и $c'_j = c_j - a_j$ для всех возможных i и j за $O(n)$.
- Если последовательности b' и c' достигают своих максимумов при $i = m_1$ и $j = m_2$ соответственно, причём $m_1 \neq m_2$, то пара индексов m_1 и m_2 является ответом.
- Если нет, то выберем максимум из

$$b'_{m_1} + \max_{k \neq m_1} c'_k$$

$$\max_{k \neq m_2} b'_k + c'_{m_2}$$

Решение: подзадача 3

- Посчитаем заранее $b'_i = b_i - a_i$ и $c'_j = c_j - a_j$ для всех возможных i и j за $O(n)$.
- Если последовательности b' и c' достигают своих максимумов при $i = m_1$ и $j = m_2$ соответственно, причём $m_1 \neq m_2$, то пара индексов m_1 и m_2 является ответом.
- Если нет, то выберем максимум из

$$b'_{m_1} + \max_{k \neq m_1} c'_k$$

$$\max_{k \neq m_2} b'_k + c'_{m_2}$$

Решение: подзадача 3

- Время: $O(N)$. Проходит тесты для $N \leqslant 100\,000$, набирает 100 баллов.

Решение: важно

- Чтобы посчитать суммарный уровень ликования, необходимо использовать тип данных для хранения 64-битных целых чисел.

Вопросы?

Задача «Съезд кинозвёзд»

Задача «Съезд кинозвёзд»

Задача «Съезд кинозвёзд»

Задача «Съезд кинозвёзд»

- Идея задачи — Михаил Тихомиров
- Подготовка тестов — Павел Кунявский
- Разбор задачи — Елена Андреева, Александр Кленин

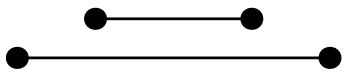
Постановка задачи

- Дано n человек. Определить, в каком порядке они должны входить и выходить, чтобы:
- a пар временных интервалов не пересеклись,
- b пар временных интервалов оказались вложенными.

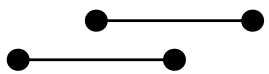
Математическая модель

- n человек образуют $\frac{n \cdot (n-1)}{2}$ пар,
- a пар временных интервалов не пересеклись,

- b пар временных интервалов оказались вложенными,



- $c = \frac{n \cdot (n-1)}{2} - a - b$ пар пересекающихся, но не вложенных интервалов.

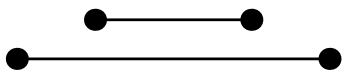


- Требуется построить такую конструкцию из интервалов с различными концами.

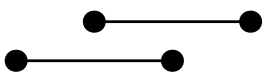
Математическая модель

- n человек образуют $\frac{n \cdot (n-1)}{2}$ пар,
- a пар временных интервалов не пересеклись,

- b пар временных интервалов оказались вложенными,



- $c = \frac{n \cdot (n-1)}{2} - a - b$ пар пересекающихся, но не вложенных интервалов.



- Требуется построить такую конструкцию из интервалов с различными концами.

Решение перебором

- Переберём порядок расположения $2n$ чисел $1\ 1\ 2\ 2\ 3\ 3\ \ldots\ n\ n$.
- Все перестановки с проверкой каждой на соответствие условию ($n \leqslant 7$, 19-20 баллов)
- Будем перебирать только нужные перестановки с помощью перебора с возвратом...

Решение перебором

- Переберём порядок расположения $2n$ чисел $1\ 1\ 2\ 2\ 3\ 3\ \ldots\ n\ n$.
- Все перестановки с проверкой каждой на соответствие условию ($n \leqslant 7$, 19-20 баллов)
- Будем перебирать только нужные перестановки с помощью перебора с возвратом...

Решение перебором

- Переберём порядок расположения $2n$ чисел $1\ 1\ 2\ 2\ 3\ 3\ \ldots\ n\ n$.
- Все перестановки с проверкой каждой на соответствие условию ($n \leqslant 7$, 19-20 баллов)
- Будем перебирать только нужные перестановки с помощью перебора с возвратом...

Перебор с возвратом

- Для каждого из $2n$ мест будем или открывать новый интервал, или закрывать один из открытых, для чего будем поддерживать список открытых интервалов.
- Не снижая общности, можно присваивать вновь открываемому интервалу первый свободный номер.
- 25 баллов

Перебор с возвратом

- Для каждого из $2n$ мест будем или открывать новый интервал, или закрывать один из открытых, для чего будем поддерживать список открытых интервалов.
- Не снижая общности, можно присваивать вновь открываемому интервалу первый свободный номер.
- 25 баллов

Отсечения перебора

- Прекратим перебор, если количество пар хотя бы одного типа превышает заданное.
- Ограничим перебор для каждого **съезда** по времени.
- Прекратим перебор, если из оставшихся чисел невозможно набрать оставшееся количество пар хотя бы одного вида...
- 60+ баллов

Отсечения перебора

- Прекратим перебор, если количество пар хотя бы одного типа превышает заданное.
- Ограничим перебор для каждого **съезда** по времени.
- Прекратим перебор, если из оставшихся чисел невозможно набрать оставшееся количество пар хотя бы одного вида...
- 60+ баллов

Отсечения перебора

- Прекратим перебор, если количество пар хотя бы одного типа превышает заданное.
- Ограничим перебор для каждого **съезда** по времени.
- Прекратим перебор, если из оставшихся чисел невозможно набрать оставшееся количество пар хотя бы одного вида...
- 60+ баллов

Отсечения перебора по b

- Прекратим перебор, если:
- $b_i + \frac{opn \cdot (opn-1)}{2} + opn \cdot (n-k) + \frac{(n-k) \cdot (n-k-1)}{2} < b.$
- k — последний использованный номер,
- opn — количество открытых временных интервалов,
- b_i — количество уже полученных вложенных пар.

Отсечения перебора по c

- Прекратим перебор, если:
- $c_i + \frac{opn \cdot (opn-1)}{2} + opn \cdot (n-k) + \frac{(n-k) \cdot (n-k-1)}{2} < c.$
- k — последний использованный номер,
- opn — количество открытых временных интервалов,
- c_i — количество уже полученных пересекающихся, но не вложенных пар.

Отсечения перебора по a

- Прекратим перебор, если:
- $a_i + k \cdot (n-k) + \frac{(n-k)\cdot(n-k-1)}{2} < a$ или $a_i + cls \cdot (n-k) > a$.
- k — последний использованный номер,
- cls — количество уже закрытых временных интервалов,
- a_i — количество уже полученных не пересекающихся пар.

Эффективная работа со списком открытых интервалов 1

- Перебираем закрываемый интервал с конца.
- За $O(1)$ удаляем и возвращаем интервал в список, меняя его с последним элементом.

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

- 95 баллов

Эффективная работа со списком открытых интервалов 1

- Перебираем закрываемый интервал с конца.
- За $O(1)$ удаляем и возвращаем интервал в список, меняя его с последним элементом.

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

- 95 баллов

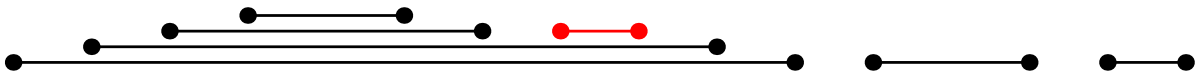
Эффективная работа со списком открытых интервалов 2

- Нельзя закрывать первые $s = opn - (c - c_i) - 1$ интервалов $(s \geqslant 0)$.
- Нельзя закрывать последние $f = opn - (b - b_i) - 1$ интервалов $(f \geqslant 0)$.
- Перебираем $opn - s - f$ допустимых закрываемых интервалов.
- За $O(n)$ удаляем и возвращаем интервал.
- 95 баллов

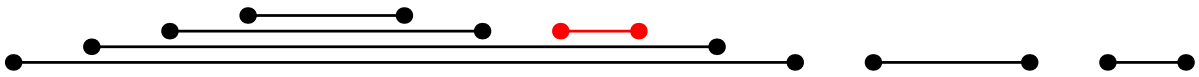
Эффективная работа со списком открытых интервалов 2

- Нельзя закрывать первые $s = opn - (c - c_i) - 1$ интервалов $(s \geqslant 0)$.
- Нельзя закрывать последние $f = opn - (b - b_i) - 1$ интервалов $(f \geqslant 0)$.
- Перебираем $opn - s - f$ допустимых закрываемых интервалов.
- За $O(n)$ удаляем и возвращаем интервал.
- 95 баллов

Конструктив 1

- Пусть $c = 0$, тогда ответ может выглядеть так: 
- «Башня» из h_1 вложенных интервалов, где $h_1 = \max\limits_i \frac{i \cdot (i-1)}{2} \leqslant b$.
- Добавим один интервал на уровень башни $b - h_1$.
- Добавим остальные интервалы справа как неперекрывающиеся.

Конструктив 1

- Пусть $c = 0$, тогда ответ может выглядеть так: 
- «Башня» из h_1 вложенных интервалов, где $h_1 = \max\limits_i \frac{i \cdot (i-1)}{2} \leqslant b$.
- Добавим один интервал на уровень башни $b - h_1$.
- Добавим остальные интервалы справа как неперекрывающиеся.

Конструктив 1

- Пусть теперь $c > 0$:
- Будем сдвигать левые границы интервалов влево, начиная с первого интервала после «башни» в порядке добавления.
- При сдвиге количество пересечений изменяется с шагом 1. Если выполнить все сдвиги, непересекающихся интервалов не останется. Следовательно, можно получить любое значение c.
- Оптимизировать вывод ответа.
- 90-100 баллов

Конструктив 1

- Пусть теперь $c > 0$:
- Будем сдвигать левые границы интервалов влево, начиная с первого интервала после «башни» в порядке добавления.
- При сдвиге количество пересечений изменяется с шагом 1. Если выполнить все сдвиги, непересекающихся интервалов не останется. Следовательно, можно получить любое значение c.
- Оптимизировать вывод ответа.
- 90-100 баллов

Конструктив 2

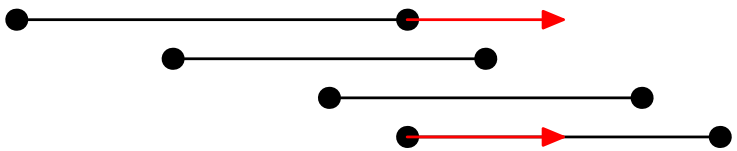
- Пока $a > n - 2$ расположим один отрезок слева, переходим к задаче для $n' = n - 1$, $a' = a - (n - 1)$.
- Пока $b > n - 2$ расположим остальные отрезки внутри первого, переходим к задаче для $n' = n - 1$, $b' = b - (n - 1)$.

Конструктив 2

- Пока $a > n - 2$ расположим один отрезок слева, переходим к задаче для $n' = n - 1$, $a' = a - (n - 1)$.
- Пока $b > n - 2$ расположим остальные отрезки внутри первого, переходим к задаче для $n' = n - 1$, $b' = b - (n - 1)$.

Конструктив 2

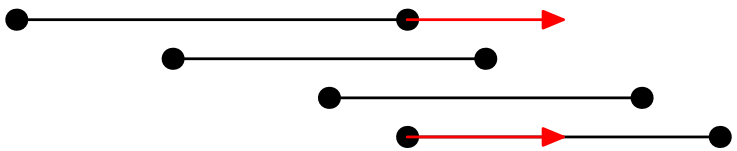
- Теперь $a \leqslant n-2$ и $b \leqslant n-2$. Расположим интервалы «лесенкой». Это даст $a' = b' = 0$.



- Сдвигаем правый конец верхней ступени вправо, увеличивая b'.
- Сдвигаем левый конец нижней ступени вправо, увеличивая a'.
- Каждое из действий можно выполнить до $n-2$ раз.
- 100 баллов

Конструктив 2

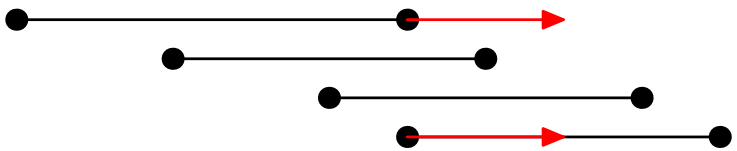
- Теперь $a \leqslant n-2$ и $b \leqslant n-2$. Расположим интервалы «лесенкой». Это даст $a' = b' = 0$.



- Сдвигаем правый конец верхней ступени вправо, увеличивая b'.
- Сдвигаем левый конец нижней ступени вправо, увеличивая a'.
- Каждое из действий можно выполнить до $n-2$ раз.
- 100 баллов

Конструктив 2

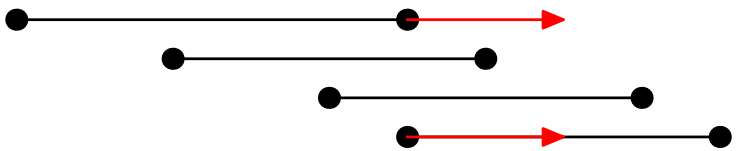
- Теперь $a \leqslant n-2$ и $b \leqslant n-2$. Расположим интервалы «лесенкой». Это даст $a' = b' = 0$.



- Сдвигаем правый конец верхней ступени вправо, увеличивая b'.
- Сдвигаем левый конец нижней ступени вправо, увеличивая a'.
- Каждое из действий можно выполнить до $n-2$ раз.
- 100 баллов

Конструктив 2

- Теперь $a \leqslant n-2$ и $b \leqslant n-2$. Расположим интервалы «лесенкой». Это даст $a' = b' = 0$.



- Сдвигаем правый конец верхней ступени вправо, увеличивая b'.
- Сдвигаем левый конец нижней ступени вправо, увеличивая a'.
- Каждое из действий можно выполнить до $n-2$ раз.
- 100 баллов

Вопросы?

Задача «Здоровое питание»

Задача «Здоровое питание»

Задача «Здоровое питание»

Задача «Здоровое питание»

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

Перебор

- Переберем все пути из верхнего-левого угла в правый-нижний.
- Для каждого пути попробуем обновить ответ для всех его клеток.
- Количество путей: $C_{2 \cdot n - 2}^{n-1}$. Оценка сверху: $2^{2 \cdot n}$.
- Каждый путь обрабатывается за $O(n^2)$. Итоговая сложность решения: $O(2^{2 \cdot n} \cdot n^2)$.
- Укладывается в TL при $n \leqslant 10$, баллы: 10.

Перебор

- Переберем все пути из верхнего-левого угла в правый-нижний.
- Для каждого пути попробуем обновить ответ для всех его клеток.
- Количество путей: $C_{2 \cdot n - 2}^{n-1}$. Оценка сверху: $2^{2 \cdot n}$.
- Каждый путь обрабатывается за $O(n^2)$. Итоговая сложность решения: $O(2^{2 \cdot n} \cdot n^2)$.
- Укладывается в TL при $n \leqslant 10$, баллы: 10.

Перебор

- Переберем все пути из верхнего-левого угла в правый-нижний.
- Для каждого пути попробуем обновить ответ для всех его клеток.
- Количество путей: $C_{2 \cdot n - 2}^{n-1}$. Оценка сверху: $2^{2 \cdot n}$.
- Каждый путь обрабатывается за $O(n^2)$. Итоговая сложность решения: $O(2^{2 \cdot n} \cdot n^2)$.
- Укладывается в TL при $n \leqslant 10$, баллы: 10.

Перебор

- Переберем все пути из верхнего-левого угла в правый-нижний.
- Для каждого пути попробуем обновить ответ для всех его клеток.
- Количество путей: $C_{2 \cdot n-2}^{n-1}$. Оценка сверху: $2^{2 \cdot n}$.
- Каждый путь обрабатывается за $O(n^2)$. Итоговая сложность решения: $O(2^{2 \cdot n} \cdot n^2)$.
- Укладывается в TL при $n \leqslant 10$, баллы: 10.

Перебор

- Переберем все пути из верхнего-левого угла в правый-нижний.
- Для каждого пути попробуем обновить ответ для всех его клеток.
- Количество путей: $C_{2 \cdot n - 2}^{n-1}$. Оценка сверху: $2^{2 \cdot n}$.
- Каждый путь обрабатывается за $O(n^2)$. Итоговая сложность решения: $O(2^{2 \cdot n} \cdot n^2)$.
- Укладывается в TL при $n \leqslant 10$, баллы: 10.

$O(n^4)$

- Количество клеток цвета c на пути от клетки A_{ij} до всех остальных вычисляется динамикой за $O(n^2)$.
- Используем данную динамику чтобы посчитать оптимальные пути от каждой клетки до всех остальных.
- Ответ для клетки: $f_{11} + f_{nn} - 1$. Баллы: 30-40.

$O(n^4)$

- Количество клеток цвета c на пути от клетки A_{ij} до всех остальных вычисляется динамикой за $O(n^2)$.
- Используем данную динамику чтобы посчитать оптимальные пути от каждой клетки до всех остальных.
- Ответ для клетки: $f_{11} + f_{nn} - 1$. Баллы: 30-40.

$O(n^4)$

- Количество клеток цвета c на пути от клетки A_{ij} до всех остальных вычисляется динамикой за $O(n^2)$.
- Используем данную динамику чтобы посчитать оптимальные пути от каждой клетки до всех остальных.
- Ответ для клетки: $f_{11} + f_{nn} - 1$. Баллы: 30-40.

$O(n^2 \cdot c)$

- Достаточно знать оптимальные пути от клеток $(1, 1)$ и (n, n) до всех остальных.
- Данные пути необходимо вычислить для всех цветов. Обозначим количество различных цветов в табличке за c.
- Переберем цвет, для каждого вычислим динамику и получим ответы для клеток данного цвета.
- Асимптотика: $O(n^2 \cdot c)$, то есть $O(n^4)$ в худшем случае.
- Баллы: 40-50.

$O(n^2 \cdot c)$

- Достаточно знать оптимальные пути от клеток $(1, 1)$ и (n, n) до всех остальных.
- Данные пути необходимо вычислить для всех цветов. Обозначим количество различных цветов в табличке за c.
- Переберем цвет, для каждого вычислим динамику и получим ответы для клеток данного цвета.
- Асимптотика: $O(n^2 \cdot c)$, то есть $O(n^4)$ в худшем случае.
- Баллы: 40-50.

$O(n^2 \cdot c)$

- Достаточно знать оптимальные пути от клеток $(1, 1)$ и (n, n) до всех остальных.
- Данные пути необходимо вычислить для всех цветов. Обозначим количество различных цветов в табличке за c.
- Переберем цвет, для каждого вычислим динамику и получим ответы для клеток данного цвета.
- Асимптотика: $O(n^2 \cdot c)$, то есть $O(n^4)$ в худшем случае.
- Баллы: 40-50.

$O(n^2 \cdot c)$

- Достаточно знать оптимальные пути от клеток $(1, 1)$ и (n, n) до всех остальных.
- Данные пути необходимо вычислить для всех цветов. Обозначим количество различных цветов в табличке за c.
- Переберем цвет, для каждого вычислим динамику и получим ответы для клеток данного цвета.
- Асимптотика: $O(n^2 \cdot c)$, то есть $O(n^4)$ в худшем случае.
- Баллы: 40-50.

$O(n^2 \cdot c)$

- Достаточно знать оптимальные пути от клеток $(1, 1)$ и (n, n) до всех остальных.
- Данные пути необходимо вычислить для всех цветов. Обозначим количество различных цветов в табличке за c.
- Переберем цвет, для каждого вычислим динамику и получим ответы для клеток данного цвета.
- Асимптотика: $O(n^2 \cdot c)$, то есть $O(n^4)$ в худшем случае.
- Баллы: 40-50.

$O(\sum_{i=1}^{c} {d_i}^2)$

- Обозначим через d_i количество клеток цвета i.
- Научимся решать задачу для цвета i за $O({d_i}^2)$.
- Отсортируем клетки по номеру строки, при равенстве - по номеру столбца.
- Будем вычислять такую же динамику как и до этого, но вместо двух переходов будем рассматривать d_i.
- Асимптотика в худшем случае остается $O(n^4)$. Баллы: 50-60.

$O(\sum_{i=1}^{c} {d_i}^2)$

- Обозначим через d_i количество клеток цвета i.
- Научимся решать задачу для цвета i за $O({d_i}^2)$.
- Отсортируем клетки по номеру строки, при равенстве - по номеру столбца.
- Будем вычислять такую же динамику как и до этого, но вместо двух переходов будем рассматривать d_i.
- Асимптотика в худшем случае остается $O(n^4)$. Баллы: 50-60.

$O(\sum_{i=1}^{c} {d_i}^2)$

- Обозначим через d_i количество клеток цвета i.
- Научимся решать задачу для цвета i за $O({d_i}^2)$.
- Отсортируем клетки по номеру строки, при равенстве - по номеру столбца.
- Будем вычислять такую же динамику как и до этого, но вместо двух переходов будем рассматривать d_i.
- Асимптотика в худшем случае остается $O(n^4)$. Баллы: 50-60.

$O(\sum_{i=1}^{c} {d_i}^2)$

- Обозначим через d_i количество клеток цвета i.
- Научимся решать задачу для цвета i за $O({d_i}^2)$.
- Отсортируем клетки по номеру строки, при равенстве - по номеру столбца.
- Будем вычислять такую же динамику как и до этого, но вместо двух переходов будем рассматривать d_i.
- Асимптотика в худшем случае остается $O(n^4)$. Баллы: 50-60.

$O(\sum_{i=1}^{c} {d_i}^2)$

- Обозначим через d_i количество клеток цвета i.
- Научимся решать задачу для цвета i за $O({d_i}^2)$.
- Отсортируем клетки по номеру строки, при равенстве - по номеру столбца.
- Будем вычислять такую же динамику как и до этого, но вместо двух переходов будем рассматривать d_i.
- Асимптотика в худшем случае остается $O(n^4)$. Баллы: 50-60.

$O(n^3)$

- Объеденим идеи использованные в двух предыдущих решениях.
- Для цветов i таких что $d_i > n$ будем использовать решение сложности $O(n^2)$.
- Если же $d_i \leqslant n$, то воспользуемся решением за $O({d_i}^2)$.
- Несложно видеть, что такая комбинация дает сложность решения $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$

- Объеденим идеи использованные в двух предыдущих решениях.
- Для цветов i таких что $d_i > n$ будем использовать решение сложности $O(n^2)$.
- Если же $d_i \leqslant n$, то воспользуемся решением за $O({d_i}^2)$.
- Несложно видеть, что такая комбинация дает сложность решения $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$

- Объеденим идеи использованные в двух предыдущих решениях.
- Для цветов i таких что $d_i > n$ будем использовать решение сложности $O(n^2)$.
- Если же $d_i \leqslant n$, то воспользуемся решением за $O({d_i}^2)$.
- Несложно видеть, что такая комбинация дает сложность решения $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$

- Объеденим идеи использованные в двух предыдущих решениях.
- Для цветов i таких что $d_i > n$ будем использовать решение сложности $O(n^2)$.
- Если же $d_i \leqslant n$, то воспользуемся решением за $O({d_i}^2)$.
- Несложно видеть, что такая комбинация дает сложность решения $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$

- Объеденим идеи использованные в двух предыдущих решениях.
- Для цветов i таких что $d_i > n$ будем использовать решение сложности $O(n^2)$.
- Если же $d_i \leqslant n$, то воспользуемся решением за $O({d_i}^2)$.
- Несложно видеть, что такая комбинация дает сложность решения $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$ другой способ

- Оптимизируем алгоритм решающий цвет за $O({d_i}^2)$ до $O(d_i \cdot n)$.
- Заметим, что в каждом столбце нас интересует только нижняя клетка из уже рассмотренных.
- Пусть b_i лучший на текущий момент результат в столбце i. Тогда $f_{ij} = 1 + max(b_k), 1 \leqslant k \leqslant j$.
- После присвоим $b_i = f_{ij}$. Таким образом мы решаем задачу за $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$ другой способ

- Оптимизируем алгоритм решающий цвет за $O({d_i}^2)$ до $O(d_i \cdot n)$.
- Заметим, что в каждом столбце нас интересует только нижняя клетка из уже рассмотренных.
- Пусть b_i лучший на текущий момент результат в столбце i. Тогда $f_{ij} = 1 + max(b_k), 1 \leqslant k \leqslant j$.
- После присвоим $b_i = f_{ij}$. Таким образом мы решаем задачу за $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$ другой способ

- Оптимизируем алгоритм решающий цвет за $O({d_i}^2)$ до $O(d_i \cdot n)$.
- Заметим, что в каждом столбце нас интересует только нижняя клетка из уже рассмотренных.
- Пусть b_i лучший на текущий момент результат в столбце i. Тогда $f_{ij} = 1 + max(b_k), 1 \leqslant k \leqslant j$.
- После присвоим $b_i = f_{ij}$. Таким образом мы решаем задачу за $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$ другой способ

- Оптимизируем алгоритм решающий цвет за $O({d_i}^2)$ до $O(d_i \cdot n)$.
- Заметим, что в каждом столбце нас интересует только нижняя клетка из уже рассмотренных.
- Пусть b_i лучший на текущий момент результат в столбце i. Тогда $f_{ij} = 1 + max(b_k), 1 \leqslant k \leqslant j$.
- После присвоим $b_i = f_{ij}$. Таким образом мы решаем задачу за $O(n^3)$.
- Баллы: 70-80.

$O(n^3)$ другой способ

- Оптимизируем алгоритм решающий цвет за $O({d_i}^2)$ до $O(d_i \cdot n)$.
- Заметим, что в каждом столбце нас интересует только нижняя клетка из уже рассмотренных.
- Пусть b_i лучший на текущий момент результат в столбце i. Тогда $f_{ij} = 1 + max(b_k), 1 \leqslant k \leqslant j$.
- После присвоим $b_i = f_{ij}$. Таким образом мы решаем задачу за $O(n^3)$.
- Баллы: 70-80.

Оптимизация до $O(n^2 \cdot log(n))$

- Заметим, что для определения значения f_{ij} необходимо запросить максимум на некотором префиксе b_i.
- Реализуем запрос максимума и операцию изменения за $O(logn)$.
- Баллы: 90-100.

Оптимизация до $O(n^2 \cdot log(n)$

- Заметим, что для определения значения f_{ij} необходимо запросить максимум на некотором префиксе b_i.
- Реализуем запрос максимума и операцию изменения за $O(logn)$.
- Баллы: 90-100.

Оптимизация до $O(n^2 \cdot log(n))$

- Заметим, что для определения значения f_{ij} необходимо запросить максимум на некотором префиксе b_i.
- Реализуем запрос максимума и операцию изменения за $O(logn)$.
- Баллы: 90-100.

Вопросы?

Задача «Магистраль Урал»

Задача «Магистраль Урал»

Задача «Магистраль Урал»

Задача «Магистраль Урал»

- Идея задачи — Макс Ахмедов
- Подготовка тестов — Нияз Нигматуллин, Никита Иоффе
- Разбор задачи — Нияз Нигматуллин

Формулировка

Входные данные:

- На плоскости есть n горизонтальных отрезков, их точное расположение вам неизвестно
- Для каждого отрезка вам задали координаты левого и правого конца: l_i и r_i. Высота, на которой находится отрезок, неизвестна

Формулировка

- Было выбрано m вертикальных прямых, заданных своей координатой
- Для i-й прямой рассмотрим все отрезки, которые пересекаются с этой прямой. Во входе заданы k_i из этих отрезков, упорядоченных сверху вниз, начиная с самого верхнего: $v_{i,j}$
- Требуется упорядочить отрезки сверху вниз, чтобы заданная вам информация была корректна

Первая подзадача

- Есть дополнительное условие: для каждой прямой выведены все пересечения
- Построим граф, где вершины — отрезки, а ребра — $v_{i,j} \to v_{i,j+1}$
- Топологически отсортируем граф, получим ответ
- Время работы: $O((n+m)^2 + \sum k_i)$

Вторая подзадача

- Построим граф, где вершины — отрезки, а ребра двух типов:
  - $v_{i,j} \to v_{i,j+1}$
  - из v_{i,k_i} во все отрезки с номером j, которые не перечислены в v_i и $l_j \leqslant x_i \leqslant r_j$
- Чтобы построить второй тип ребер, используем метод сканирующей прямой
- Топологически отсортируем граф, получим ответ
- Время работы: $O((n+m)^2 + \sum k_i)$

Третья подзадача

- Строим такой же граф, как и во второй подзадаче
- Используем битовое сжатие для построения, хранения и топологической сортировки графа
- Время работы: $O((n+m)^2 + \sum k_i)$ с маленькой константой.

Полное решение

- Построим граф, где вершины — отрезки и прямые, а ребра двух типов:
  - $v_{i,j} \to v_{i,j+1}$
  - $v_{i,k_i} \to u_i$, u_i — прямая с номером i
- Будем топологически сортировать граф, восстанавливая сортировку снизу вверх
- Для каждой прямой будем поддерживать число необработанных отрезков $under_i$, которые лежат под скважиной на этой прямой
- Также для каждой вершины будем поддерживать число ребер out_i, исходящих из нее в необработанные вершины

Полное решение

- Вершина v может быть обработана (может быть последней в топологической сортировке оставшихся вершин):
  - Если $under_v = 0$, в случае, если v — прямая
  - Если $out_v = 0$, в случае, если v — отрезок
- Заведем очередь вершин, которые предстоят обработке
- Обработка вершины v:
  - Для всех u, из которых есть ребро в v уменьшить out_u на единицу
  - Уменьшить на единицу $under_i$ для всех прямых i, под скважинами которых, находится отрезок v

Полное решение

- Для уменьшения $under_i$ нужно воспользоваться структурой данных, с помощью которой мы сможем прибавить число и найти минимум на отрезке.
- После обработки i-й прямой, присвоить $under_i$ значение «бесконечность»
- В итоге порядок обработки вершин-отрезков, совпадает с каким-то из порядков снизу вверх.
- Время работы: $O((n+m)\log(n+m)+\sum k_i)$

Четвертая подзадача

- Решения, которые недостаточно оптимально реализуют правильные идеи либо используют то, что число пересечений прямых с отрезками маленькое, получают 80 баллов

Вопросы?