Всероссийская олимпиада школьников по информатике

Иннополис

26 марта – 1 апреля 2017 года

Задача 1 «Программирование квадрокоптеров»

- Идея задачи — жюри олимпиады
- Подготовка тестов — Максим Ахмедов, Григорий Резников, Рамис Ямилов
- Разбор задачи — Рамис Ямилов

Постановка задачи

- Загадана правильная скобочная последовательность длины n
- Разрешается задавать вопросы про правильность некоторых фрагментов этой последовательности
- Требуется отгадать эту последовательность

Решение на 49 баллов

- Спросим про все подотрезки
- Если подотрезок является правильной скобочной последовательностью, то на ее концах находятся открывающая и закрывающая скобки
- Каждый символ принадлежит границе какого-либо правильного подотрезка
- За $n(n-1)/2$ запросов мы выясним информацию про все символы

Решение

- Поскольку последовательность является правильной, для каждой открывающей скобки имеется парная закрывающая, находящаяся правее
- Будем находить символы последовательности по порядку
- Пусть мы уже выяснили, какие символы находятся на позициях $1 \dots i-1$, а последняя открывающая скобка без пары находится на позиции x

Решение

- Для того, чтобы узнать, какой символ находится на позиции i, зададим вопрос про фрагмент $[x \dots i]$
- Если на позиции i находится закрывающая скобка, то она является парной к открывающей с позиции x, и поэтому фрагмент $[x \dots i]$ будет правильной скобочной последовательностью

Решение

- Для того чтобы эффективно находить последнюю открывающую скобку без пары, будем хранить их позиции в стеке
- Если на вопрос про фрагмент $[x \dots i]$ получили ответ «No», тогда на позиции i находится открывающая скобка, поэтому добавим i в стек
- Иначе мы нашли закрывающую скобку, соответствующую скобке на позиции x, поэтому удалим верхний элемент из стека

Решение

- Для каждого символа, кроме первого, мы зададим по одному вопросу
- Итого потребуется не более n вопросов
- Асимптотика решения $O(n)$

Вопросы?

Задача 2 «Иллюзия сортировки»

- Идея задачи — Станислав Наумов
- Подготовка тестов — Михаил Тихомиров, Андрей Календаров
- Разбор задачи — Андрей Календаров

Постановка задачи

- Дан массив из целых неотрицательных чисел
- Поступают запросы вида «присвоить некоторое значение элементу»
- После каждого запроса необходимо вывести минимальное целое неотрицательное число x (если оно существует), такое что массив $a \oplus x$ отсортирован

Решение на 30 баллов $\mathcal{O}(qnA)$

Заметим, что если все числа меньше 2^k, то минимальное x также меньше 2^k, поскольку более старшие биты не будут влиять на сравнения
Сделаем, что написано:

- После каждого запроса перебёрем число x
- Для каждого $1 \leqslant i \leqslant n-1$ проверим что $a_i \oplus x \leqslant a_{i+1} \oplus x$

Исследование

- Массив отсортирован, если для каждой пары соседних элементов выполнено $a_i \leqslant a_{i+1}$
- Рассмотрим, как числа a_i и a_{i+1} влияют на x
- Если числа равны, то они не задают никаких ограничений
- Иначе, старший бит, в котором числа отличаются, задаёт ограничение на x
- Таким образом, есть $n - 1$ ограничение на x вида «i-й бит равен 0 или 1»

Решение на 59 баллов $\mathcal{O}(qn \log A)$

- После очередного запроса для каждого $1 \leqslant i \leqslant n-1$ получим ограничение на x
- Если ограничения противоречивы (одновременно присутствуют $\{i, 0\}$ и $\{i, 1\}$), то ответа не существует
- Иначе для каждого ограничения $\{i, 1\}$ выставим в x i-й бит
- Легко доказать, что такое x минимально

Решение на 80 баллов $\mathcal{O}(q \log n \log A)$

Используем структуры данных:

- Сохраним ограничения в дереве отрезков
- В вершине для каждой пары $\{i, 0/1\}$ хранится 0 или 1 — есть ли на отрезке данное ограничение
- Запрос изменения — подъём по дереву за $\mathcal{O}(\log n \log A)$
- Ответ на запрос формируем в корне дерева отрезков за $\mathcal{O}(\log A)$

Полное решение $\mathcal{O}(q(\log n + \log A))$

Оптимизируем дерево отрезков:

- Заметим, что в предыдущем решении в каждой вершине хранилось 60 булевых значений
- Используем битовое сжатие
- Запрос изменения — подъём по дереву $\mathcal{O}(\log n)$
- Ответ на запрос формируем в корне дерева отрезков за $\mathcal{O}(\log A)$

Полное решение $\mathcal{O}(q \log A)$

Избавимся от дерева отрезков:

- Заметим, что мы использовали только корень дерева отрезков для ответа на запрос
- Заменим дерево отрезков на массив (количество ограничений для каждой пары $\{i, 0/1\}$)
- Обрабатываем запрос и отвечаем за $\mathcal{O}(\log A)$

Дальнейшие оптимизации

Работаем с битами:

- Что нас тормозит?
- Поиск старшего бита в числе x
- Используем функцию `__builtin_clz` (count leading zeroes за $O(\log w)$)
- На некоторых компьютерах доступна ассемблерная инструкция BSR
- Предподсчитаем ответ для всех 16-битных слов

Вопросы?

Задача 3 «Тигры»

- Идея задачи — Максим Ахмедов
- Подготовка тестов — Сергей Копелиович, Роман Горбунов, Дмитрий Иващенко
- Разбор задачи — Дмитрий Иващенко

Постановка задачи

- Система обнаружения должна последовательно определить местоположение каждого из q тигров на плане заповедника
- Для этого она может запрашивать информацию у n приёмников, расположенных на территории
- Задача интерактивная: необходимо реализовать стратегию обнаружения тигров в процессе обмена информации с приёмниками

Математическая модель

- n точек на плоскости
- Загадывается q точек
- Найти выпуклый многоугольник, такой что в нём находится только загаданная точка
- Запрос: выпуклый многоугольник с вершинами в приёмниках.
- Ответ: есть ли в нём загаданная точка
- Никакие три приёмника, никакие два приёмника и тигр не лежат на одной прямой

Полный перебор

- Переберём $n!$ перестановок точек, проверим, что они образуют выпуклый многоугольник
- Проверим, что внутри него нет никакой другой точки-приёмника
- Сделаем запрос, если ответ положительный, то это ответ
- $O(n!)$ времени и запросов, **15** баллов
- Оптимизация: можно не проверять то, что внутри нет других точек, а просто выбрать многоугольник наименьшей площади

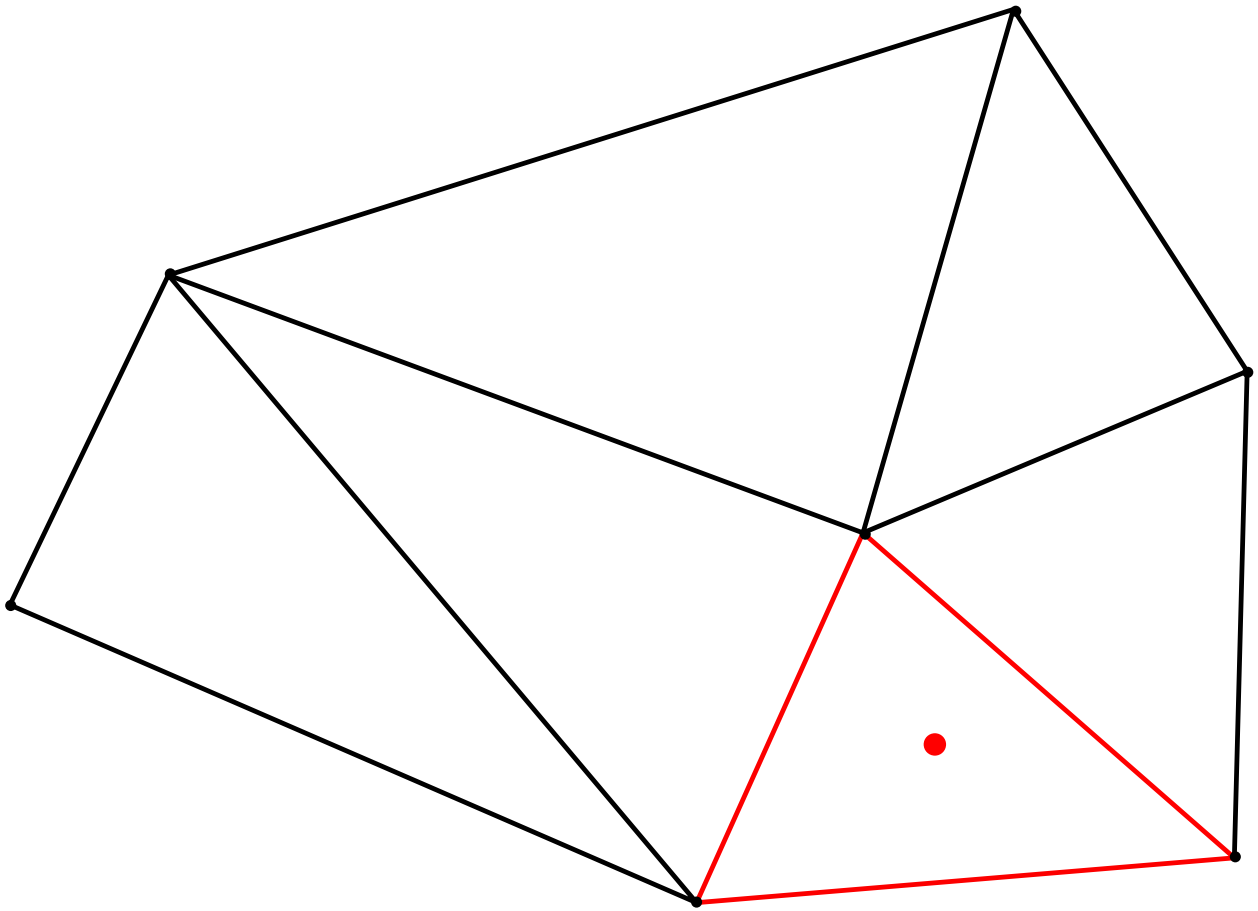
Пробуем треугольники

- Соображение: достаточно проверять только треугольники (*теорема Каратеодори*)
- Проверим все треугольники, выберем наименьший по площади
- $O(n^3)$ времени и запросов, **32** балла

Триангуляция

- Идея: построим триангуляцию данного набора точек
- В триангуляции не больше $2n$ треугольников
- $O(n^2)$ или $O(n \log n)$ времени на построение, не больше $2n$ запросов, от **51** до **60** баллов
- Если перебирать элементы триангуляции в случайном порядке, то в среднем будет сделано вдвое меньше запросов

Триангуляция: пример



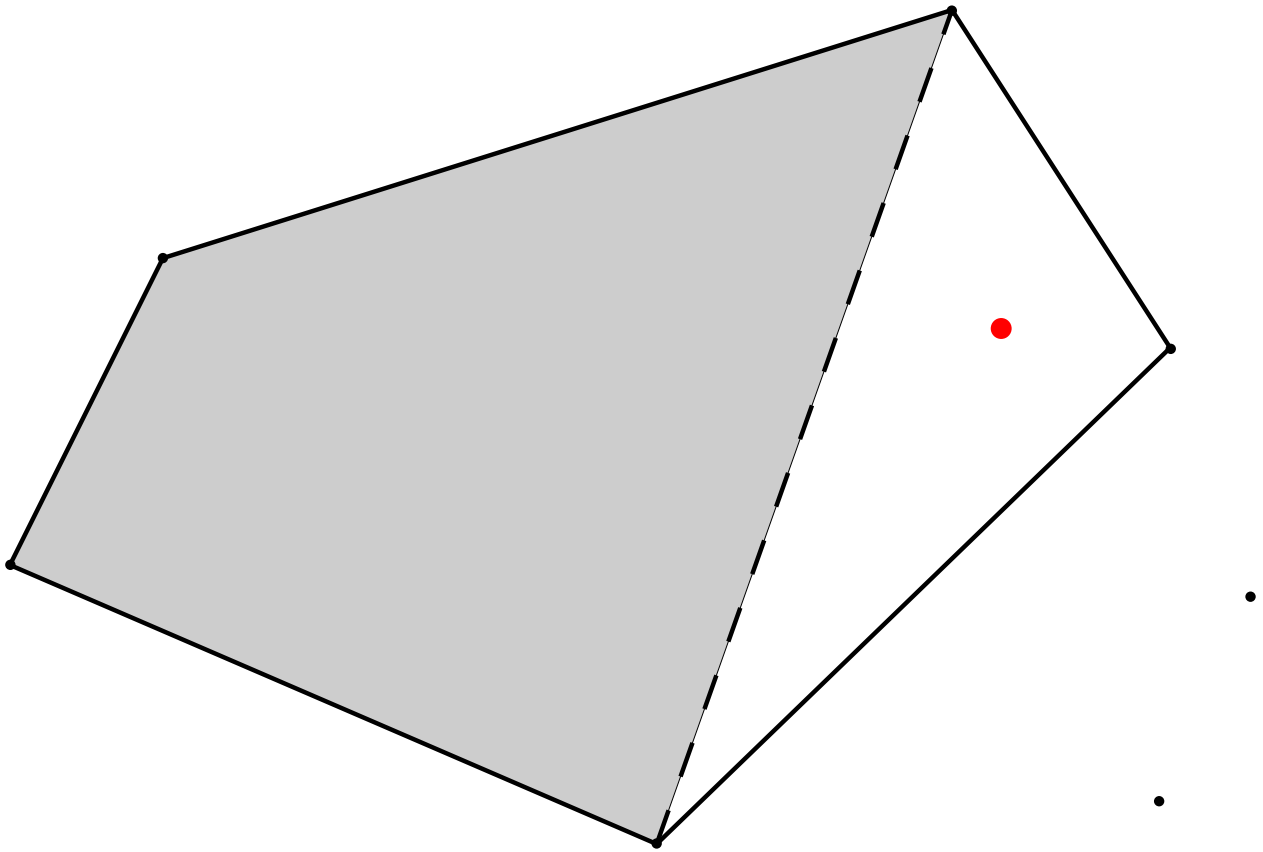
Поиск в треугольнике

- Предположим, что искомая точка находится в известном нам треугольнике
- Выберем случайную точку внутри него, она разбивает треугольник на три
- Сделаем запрос про каждый из них, перейдём в нужную часть
- Последний запрос делать не обязательно. Остальные два можно делать в случайном порядке, чтобы в среднем было 1.5 запроса.

Локализация точки

- Построим выпуклую оболочку, выберем самую нижнюю точку, из них самую левую. Остальные отсортируем по углу
- Бинарным поиском найдём многоугольник на первых k точках в этом порядке, который содержит тигра
- Разность этого многоугольника и предыдущего — треугольник, в котором находится тигр
- От **70** до **90** баллов в зависимости от реализации

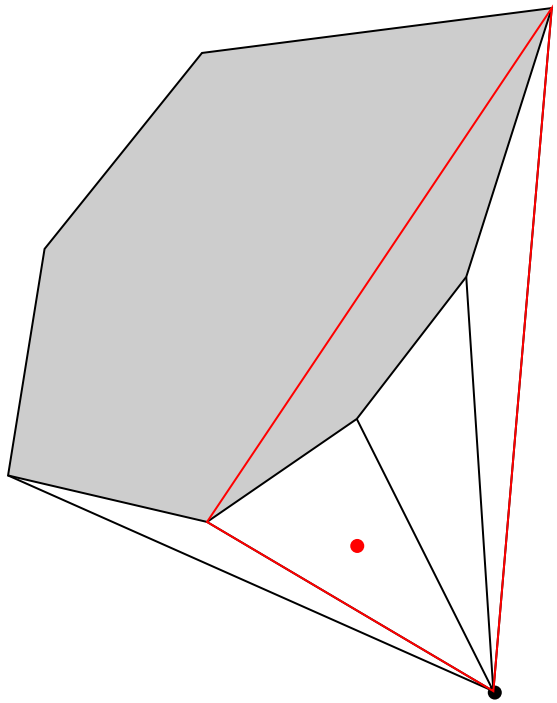
Локализация точки: пример



Сканирование по Грэхему

- Строим выпуклую оболочку алгоритмом *Грэхема*
- На i-м шаге имеем выпуклую оболочку первых i точек
- Бинарным поиском найдём шаг k, на котором тигр в первый раз попадает в выпуклую оболочку
- Разность многоугольников на этом шаге и предыдущем представляет собой набор треугольников с общей вершиной

Сканирование по Грэхему: пример



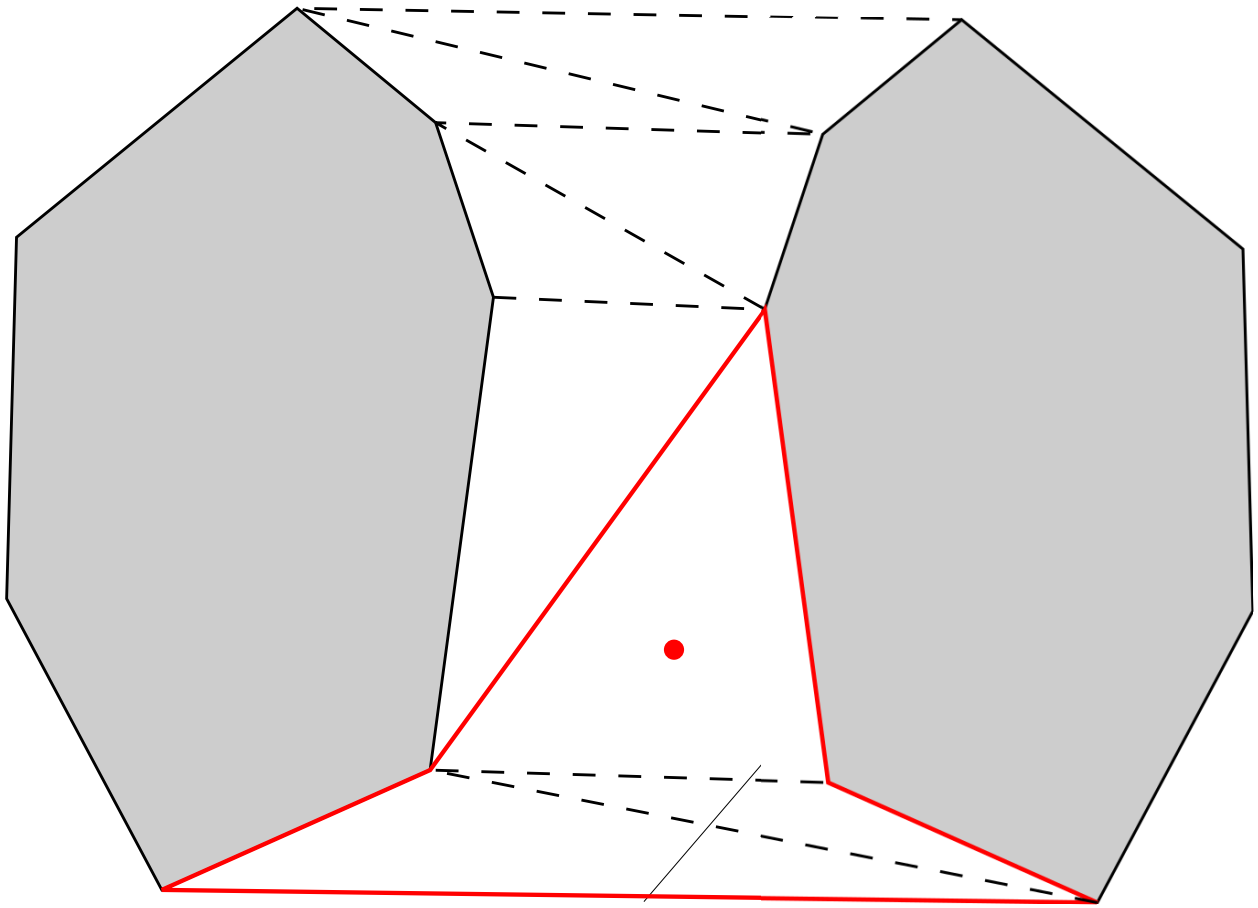
Сканирование по Грэхему II

- Можем делать по этому набору бинарный поиск, используя для запроса только треугольники
- **90 – 100** баллов
- Проблема нескольких предыдущих решений: суммарный объём запросов $O(qn \log n)$

Разделяй и властвуй

- Применим технику «*разделяй и властвуй*»
- Вертикальной прямой делим точки на две равные части, делаем два рекурсивных вызова
- Выпуклая оболочка всего набора точек за вычетом выпуклых оболочек двух частей представляет собой невыпуклую фигуру вида «песочные часы»
- Двумя указателями строим триангуляцию полученной фигуры

Разделяй и властвуй: пример



Разделяй и властвуй II

- Первые i нижних треугольников образуют фигуру, которую можно досторить до четырёхугольника, так как в выделенных частях искомой точки точно нет
- Четырёхугольник может быть невыпуклым. В этом случае разобъём его на два треугольника
- Бинарный поиск по полученной фигуре
- Время $O(n \log n)$, число запросов $O(\log n)$, суммарный объём запросов $O(n)$, **100** баллов

Вопросы?

Задача 4 «Путешествие в Метрополис»

- Идея задачи — Михаил Пядёркин
- Подготовка тестов — Григорий Резников, Павел Кунявский, Михаил Тихомиров
- Разбор задачи — Григорий Резников

Постановка задачи

- Дан взвешенный ориентированный граф и маршруты на нём
- Необходимо проехать из города 1 в город n за минимальное время
- Среди всех вариантов с минимальным временем выбрать вариант с максимальной суммой квадратов времён между пересадками

$s_i = 1$

- $O(n!)$ — Переберём перестановку вершин и выберем лучшую.
- $O(n^2)$ — модификация алгорита Дейкстры.

Решение на 54 балла

- При помощи алгоритма Дейкстры посчитаем d_i — кратчайшее расстояние от вершины 1 до вершины i
- Для минимизации суммарного времени можно передвигаться только по таким рёбрам из u в v веса w, что $d_v = d_u + w$
- Такие рёбра образуют ориентированный ациклический граф

Решение на 54 балла

- Посчитаем dp_i — максимальную сумму квадратов времён если мы едем от вершины i
- Для подсчёта dp_i переберём, какой маршрут будет следующим и в какой вершине будет следующая пересадка
- $O(n \cdot \sum_{i=1}^{p} s_i)$ и **54** или **73** балла

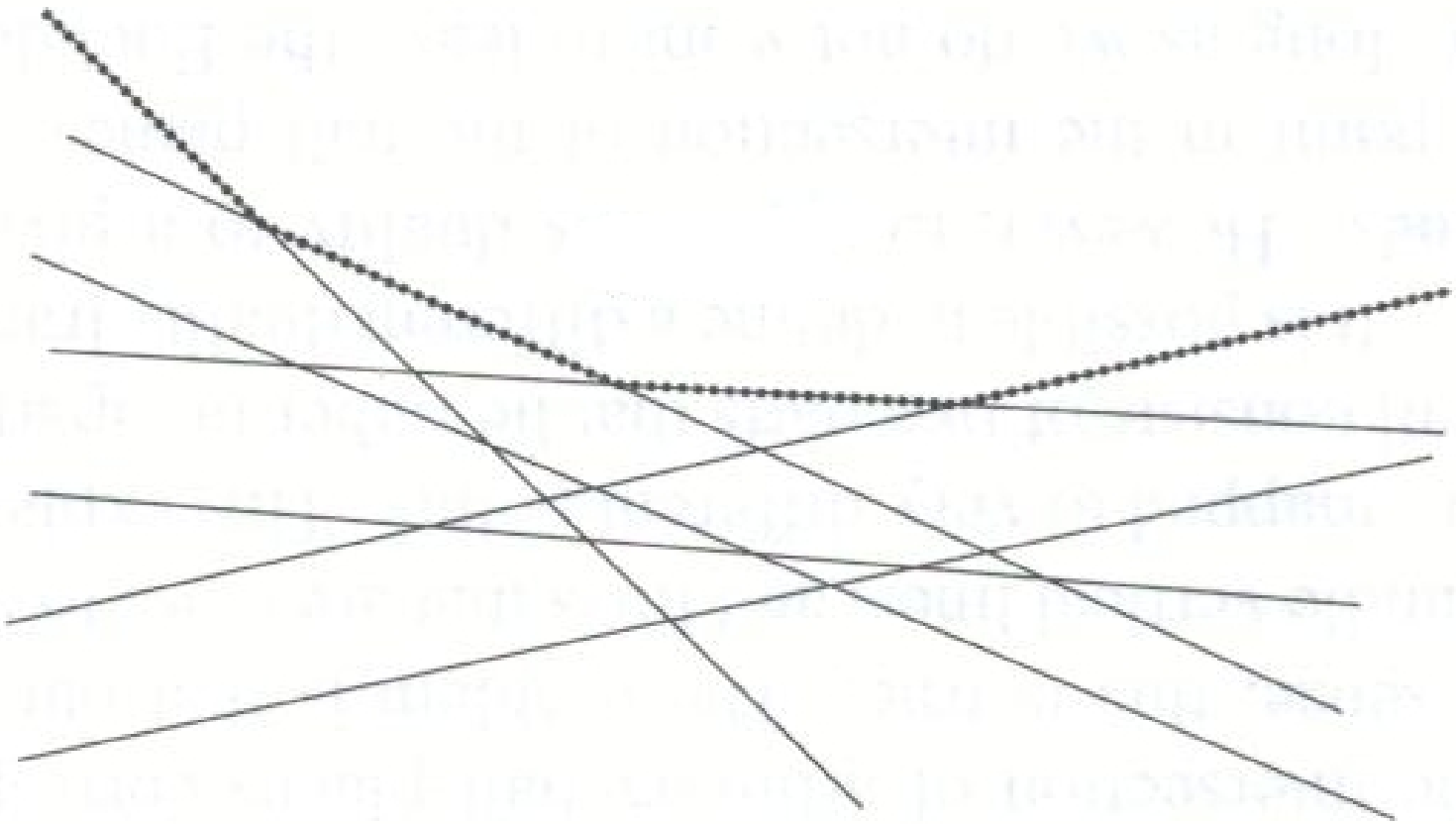
Решение на 73 балла

- Предподсчитаем $r_{u,v}$ — можно ли добраться из u в v при помощи ровно одного маршрута, двигаясь по кратчайшим путям
- Этот массив можно посчитать за $O(\frac{(\sum_{i=1}^{p} s_i)^2}{w})$ при помощи битового сжатия, где w — длина машинного слова в битах (32 или 64)
- Теперь при подсчёте dp_i можно перебрать следующую вершину, в которой произойдёт пересадка
- $O(\frac{(\sum_{i=1}^{p} s_i)^2}{w} + n^2)$ и гарантированные **73** балла

Полное решение

- Аккуратно выпишем формулу перехода при подсчёте dp_i
- Пусть мы зафиксировали путь P, мы хотим продолжить движение
- $dp_i = \max\limits_{v \in P} dp_v + (d_v - d_i)^2 = \max\limits_{v \in P} dp_v + d_v^2 + d_i^2 - 2 \cdot d_v d_i = \max\limits_{v \in P} -2 \cdot d_v d_i + (dp_v + d_v^2) + d_i^2$
- Рассмотрим для каждого пути множество прямых вида $y = 2 \cdot d_v x + dp_v + d_v^2$
- Необходимо быстро добавлять прямую в множество и искать для данного x $\max\limits_i y_i(x)$

Полное решение



Решение

- Заметим, что прямые добавляются в порядке возрастания угловых коэффициентов
- Можно воспользоваться стандартным алгоритмом Convex Hull Trick
- $O(\sum_{i=1}^{p} s_i \log n)$ и **92 - 100** баллов в зависимости от аккуратности реализации

Вопросы?

Задача 5 «Накопитель»

- Идея задачи — Александр Кленин
- Подготовка тестов — Михаил Тихомиров, Андрей Календаров
- Разбор задачи — Александр Кленин

Постановка задачи

- Строки s и t состоят из символов «+» и «-»
- Фрагмент — максимальная по включению подстрока из одинаковых символов
- Если рядом есть два фрагмента разной длины, то больший может «съесть» меньший
- Требуется превратить s в t любым способом

Подзадачи 1 и 2 (50 баллов)

- Заменим строку на последовательность длин фрагментов
- Строка t короткая, состоит только из «+»
- Никогда не нужно заменять «+» на «-»
- Превратим любой фрагмент s из «-», который можно превратить в «+», повторим пока возможно
- Сложность $O(n^2)$

Подзадача 3 (70 баллов)

- Строка t длинная, состоит только из «+»
- Пройдём по фрагментам s слева направо, превратим «-» в «+» пока возможно, затем сложим в стек
- Перед помещением в стек превращаем фрагмент в вершине стека, пока возможно
- Амортизированная сложность $O(n)$

Подзадача 4 (70 баллов)

- Строка t короткая, из любых символов
- Рассмотрим каждый фрагмент t независимо
- Фрагменты не разбиваются $\Rightarrow$ граница фрагментов t должна быть границей в s
- Граница не должна исчезнуть $\Rightarrow$ символы вокруг границы в t должны совпадать с s

s	?	?	?	$\ominus$	$+$	?	?
t	$+$	$+$	$+$	$\oplus$	$-$	$-$	$-$

Подзадача 5 (100 баллов)

- Бонус! Автоматически проходится при сочетании предыдущих идей

Вопросы?

Задача 6 «Серверы на Меркурии»

- Идея задачи — Михаил Путилин
- Подготовка тестов — Глеб Евстропов, Роман Горбунов, Рамис Ямилов, Павел Маврин
- Разбор задачи — Роман Горбунов

Постановка задачи

- Есть n серверов, соединённых по цепочке. Для каждого сервера нужно узнать минимальное время, когда на него можно послать обновление, чтобы оно установилось на все серверы
- Для каждого сервера известно время хранения обновления, для каждого соединения известно время, когда через него могло пройти обновление

Решение за $O(n^2 \cdot t)$, 20 баллов

- Переберём сервер, в который отправим обновление
- Переберём время, в которое мы отправим это обновление
- Промоделируем процесс за $O(n)$

Решение за $O(n^2)$, при $t_i \geqslant \max r_j$, 10 баллов

- Заметим, что ответ всегда либо 0 или -1
- Переберём вершину, в которую отправим обновление
- Промоделируем процесс за линию, отправляя обновление в момент времени 0

Решение за $O(n^2 \cdot \log C)$, при $r_i = C$, 10 баллов

- Мы можем отправить обновление в момент времени C, так как все соединения будут в этот момент открыты
- Также заметим, что если мы можем корректно отправить обновление в момент x, то можем и в любое время позже
- Переберём сервер, запустим двоичный поиск по времени и промоделируем процесс за $O(n)$

Решение за $O(n \cdot t)$, 50 баллов

- Динамическое программирование
- Состояние — пара (сервер, время)
- Считаем, правда ли, что если в этот сервер в это время придет обновление, то оно распространится до конца вправо
- Пересчет за $O(1)$
- Аналогично влево

Решение за $O(n)$, при при $t_i \geqslant \max r_j$, 15 баллов

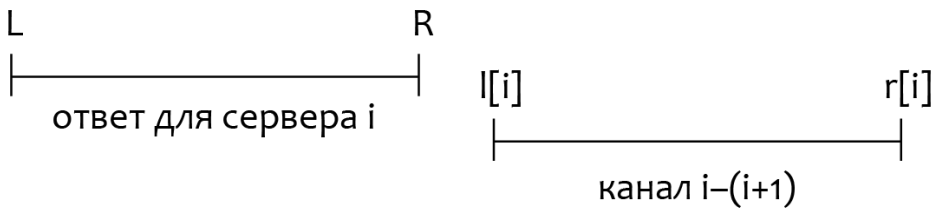
- Будем решать отдельно левую и правую подзадачи
- Ответ — отрезок $[0, R]$
- Решим правую подзадачу, левая аналогично

Решение за $O(n)$, при при $t_i \geqslant \max r_j$, 15 баллов

$$l[i] \leqslant R \Rightarrow R' = \min(R, r_i)$$

Решение за $O(n)$, при при $t_i \geqslant \max r_j$, 15 баллов

$l_i > R \Rightarrow$ ответ -1



Решение за $O(n)$, при $r_i = C$, 15 баллов

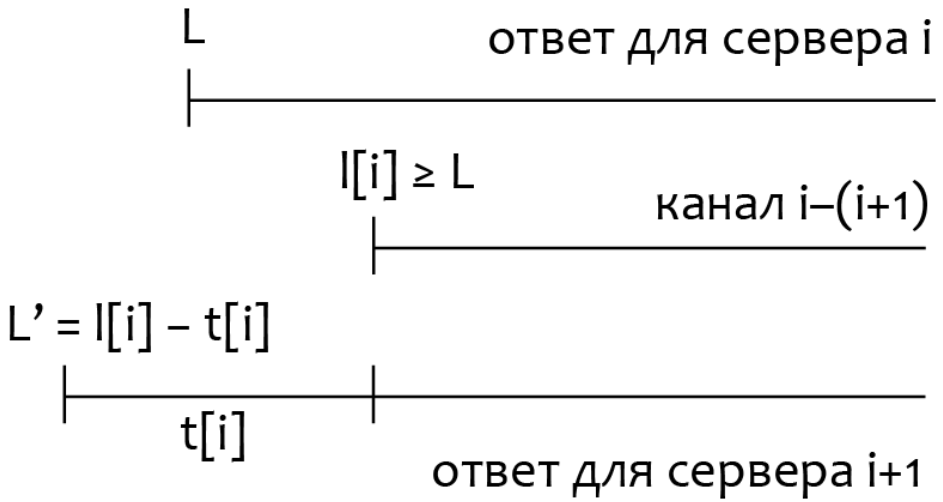
- Будем решать отдельно левую и правую подзадачи
- Ответ — отрезок $[L, \infty)$
- Решим правую подзадачу, левая аналогично

Решение за $O(n)$, при $r_i = C$, 15 баллов

$$l[i] < L \Rightarrow L' = L$$

Решение за $O(n)$, при $r_i = C$, 15 баллов

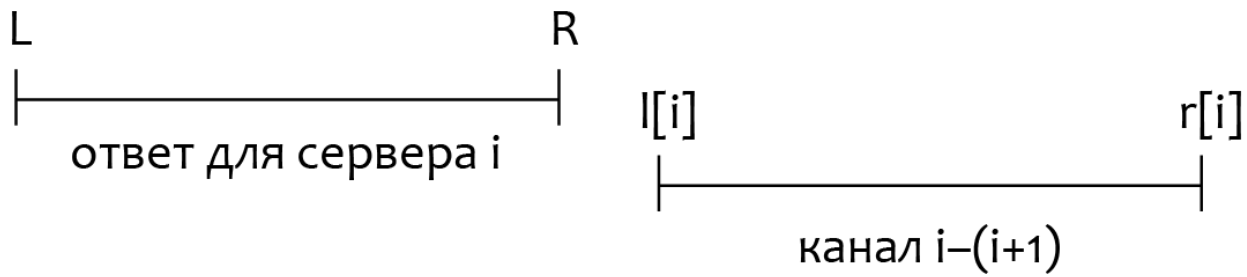
$$l_i \geqslant L \Rightarrow L' = l_i - t_i$$



Решение за $O(n)$, 100 баллов

- Решаем отдельно левую и правую подзадачи
- Ответ является непрерывным отрезком времени
- Решим правую подзадачу, левая аналогично
- Для первого сервера ответ $[0; \infty)$

Решение за $O(n)$, 100 баллов



Если отрезок времени, когда канал открыт, не пересекается с допустимым отрезком, то для $i + 1$ и всех следующих ответа нет

Решение за $O(n)$, 100 баллов

Левая граница так же как в предыдущей подзадаче

Решение за $O(n)$, 100 баллов

- Считаем ответы для каждой из двух подзадач: влево и вправо
- Когда считаем ответ для сервера просто пересекаем отрезки ответов для левой и правой сторон, в случае отсутствия их пересечения, или не существования одного из них ответ -1

Вопросы?

Задача 7 «Антиматерия»

- Идея задачи — Максим Ахмедов
- Подготовка тестов — Павел Кунявский, Григорий Резников, Дмитрий Иващенко
- Разбор задачи — Павел Кунявский

Постановка задачи

- Можно проводить n типов экспериментов
- i-й эксперимент даёт от l_i до r_i антиматерии, стоит c_i денег
- Нельзя набирать больше a антиматерии
- 1 единица антиматерии стоит 10^9 денег
- Надо заработать как можно больше

Решение. Случай $n = 1$

- Единица антиматерии дороже любого допустимого количества экспериментов
- Проводим эксперимент пока нет шанса набрать больше a единиц антиматерии
- Нас интересует минимальное число $x > a - r$, которое может получиться
- Найдём x, перебрав количество нажатий на кнопку

Решение. Случай $l_i = r_i$

- В этом случае результаты каждого эксперимента однозначно определены
- Необходимо выбрать такой набор экспериментов, чтобы сумма была максимальной, не превосходящей a
- Это является одной из форм «задачи о рюкзаке» и решается динамическим программированием

Решение. ДП за $\mathcal{O}(nA^2)$ на 40 баллов

- dp_x — максимальная прибыль, которую можно дополнительно получить, если уже есть x единиц
- Перебираем следующий эксперимент, выбираем худший сценарий (с минимальной прибылью)
- $dp_x =$ максимальное среди значений:
  - $x \cdot 10^9$ (закончить эксперименты)
  - $\min\limits_{l_i \leqslant j \leqslant r_i} dp_{x+j} - c_i$, если $j + r_i \leqslant a$ (провести i-й эксперимент)

Решение. Общий случай $\mathcal{O}(nAlogA)$

- dp_x = максимальное среди значений:
  - $x \cdot 10^9$
  - $\min\limits_{l_i \leqslant j \leqslant r_i} dp_{x+j} - c_i$, если $j + r_i \leqslant a$
- Переход в ДП — взятие минимума на отрезке
- Используем дерево отрезков
- В зависимости от реализации может получить от 60 до 76 баллов

Решение. Общий случай $\mathcal{O}(nA), \mathcal{O}(nA)$

- dp_x = максимальное среди значений:
  - $x \cdot 10^9$
  - $\min\limits_{l_i \leqslant j \leqslant r_i} dp_{x+j} - c_i$, если $j + r_i \leqslant a$
- Для каждого эксперимента, и левая и правая границы сдвигаются влево
- Задача может быть решена любым из способов реализации очереди с минимумом
- Некоторые из таких способов могут работать используя $O(nA)$ памяти

Решение. Общий случай $\mathcal{O}(nA), \mathcal{O}(AlogA)$

- dp_x = максимальное среди значений:
  - $x \cdot 10^9$
  - $\min\limits_{l_i \leqslant j \leqslant r_i} dp_{x+j} - c_i$, если $j + r_i \leqslant a$
- Другой способ искать минимум на отрезке — разреженная таблица
- Разреженную таблицу можно достраивать по ходу вычисления

Решение. Общий случай $\mathcal{O}(nA\alpha), \mathcal{O}(A)$

- dp_x = максимальное среди значений:
  - $x \cdot 10^9$
  - $\min\limits_{l_i \leqslant j \leqslant r_i} dp_{x+j} - c_i$, если $j + r_i \leqslant a$
- Самое быстрое решение, известное жюри — offline-rmq с использованием системы непересекающихся множеств

Решение. Offline-rmq

- Поддерживаем стек «потенциальных минимумов» — элементов, левее которых нет меньшего
- Блоки между ними будем хранить в системе непересекающих множеств
- При удалении из стека, необходимо объединить два множества
- Чтобы выполнить запрос, надо найти самый левый элемент множества, содержащего правую границу

Вопросы?

Задача 8 «Траектория обучения»

Задача 8 «Траектория обучения»

- Идея задачи — Георгий Халин
- Подготовка тестов — Михаил Пядёркин, Максим Ахмедов, Нияз Нигматуллин
- Разбор задачи — Максим Ахмедов

Постановка задачи

- Даны две последовательности курсов длины n и m соответственно
- Каждый курс задаётся дисциплиной и рейтингом, все дисциплины в одной последоватлеьности различны
- Выбрать по одному подотрезку в каждой последовательности, чтобы никакая дисциплина не встречалась дважды и суммарный рейтинг был максимален

Дисклеймер

В данной презентации длины обеих последовательностей обозначаются как n :)

Простейшая реализация

- Реализуем ровно по тексту условия задачи
- Фиксируем подотрезок в каждой последовательности
- Выписываем объединение этих подотрезков
- Проверяем, что никакое число не встречается дважды с помощью сортировки, std::set или массива пометок
- Сложность $O(n^5 \log n)$ или $O(n^5)$
- 10 – 20 баллов

Простая реализация

- Проверяем пары отрезков в специальном порядке
- Фиксируем первый отрезок и левую границу второго
- Перебираем правую границу второго отрезка, добавляем каждый очередной элемент в массив пометок, и останавливаем процесс, когда появляется дубль
- Сложность $O(n^4)$
- 30 баллов

Два указателя

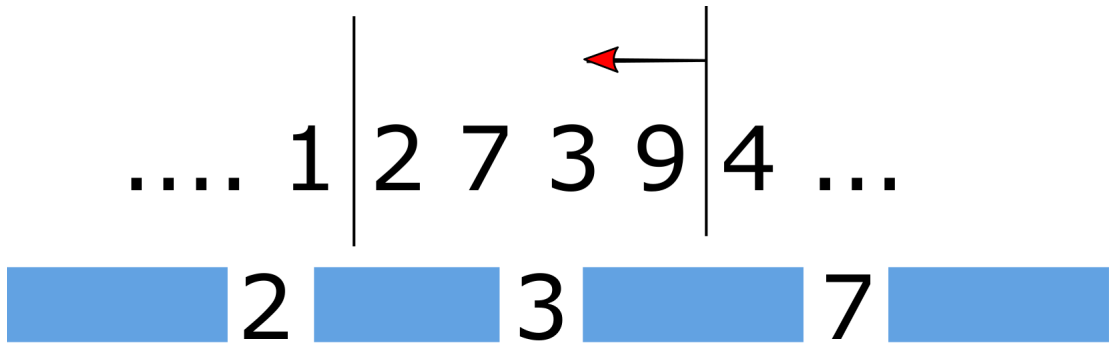
- Зафиксируем левые границы отрезков
- По фиксированной правой границе первого отрезка r_1 определяется лучшая правая допустимая граница второго отрезка r_2
- r_2 уменьшается при увеличении r_1
- Последовательно увеличиваем r_1 на 1 и уменьшаем r_2 пока требуется
- Для фиксированных l_1 и l_2 работает за $O(n)$, итоговое время работы — $O(n^3)$
- 40 баллов

Разные квадратичные решения

- Зафиксируем отрезок первой последовательности
- Использованные элементы разбивают вторую последовательность на отрезки
- Из этих отрезков нас всегда интересует отрезок максимального веса

Разные квадратичные решения

- Если двигать правую границу первого отрезка влево, то элементы разрешаются, и отрезки склеиваются
- Будем поддерживать множество отрезков в какой-нибудь структуре данных



Дерево отрезков

- Можно воспользоваться деревом отрезков, для изменения элементов и нахождения подотрезка с максимальной суммой
- В каждой вершине дерева отрезков храним тройку чисел (max, max_l, max_r)
  - max = максимальная из сумм на подотрезках
  - max_l = максимальная из сумм на префиксах
  - max_r = максимальная из сумм на суффиксах
- Отсутствующие элементы заменяем на $-\infty$
- Сложность $O(n^2 \log n)$, около 50 баллов

`std::set` / `TreeSet`

- Поддерживаем отрезки в каком-нибудь сбалансированном двоичном дереве, например, пользуемся стандартным `std::set` / `TreeSet`
- Когда очередная позиция разрешается, объединяем не более чем два отрезка
- Поддерживаем отрезок максимального веса
- Сложность $O(n^2 \log n)$, около 50 баллов

Система непересекающихся множеств

- Поддерживаем элементы, состоящие в одном отрезке, в системе непересекающихся множеств
- Делаем всё то же самое, что и в предыдущем решении
- Сложность $O(n^2\alpha)$, около 55 баллов

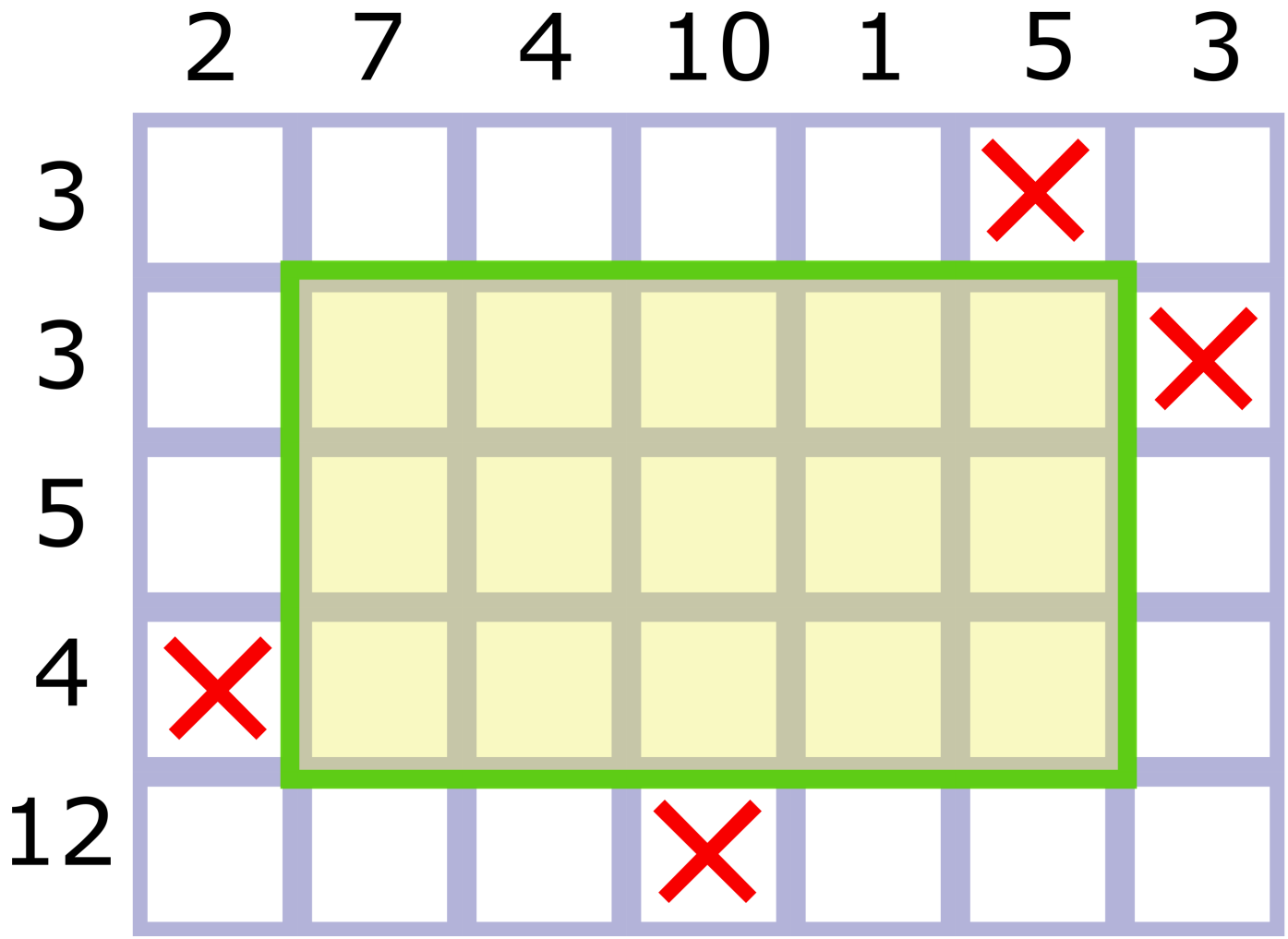
Двусвязные списки

- Поддерживаем запрещённые элементы в двусвязном списке, запоминаем соседей слева и справа
- При разрешении очередного элемента перестраиваем список и релаксируем ответ новообразовавшимся отрезком
- Сложность $O(n^2)$, около 60 баллов

Графическая интерпретация

- Рассмотрим матрицу n на m
- Если какой-то элемент встречается в первой последовательности на позиции i и во второй последовательности на позиции j, отметим клетку (i, j)
- Сопоставим двум подотрезкам прямоугольник в этой матрице
- Этот прямоугольник не может содержать отмеченные клетки

Графическая интерпретация



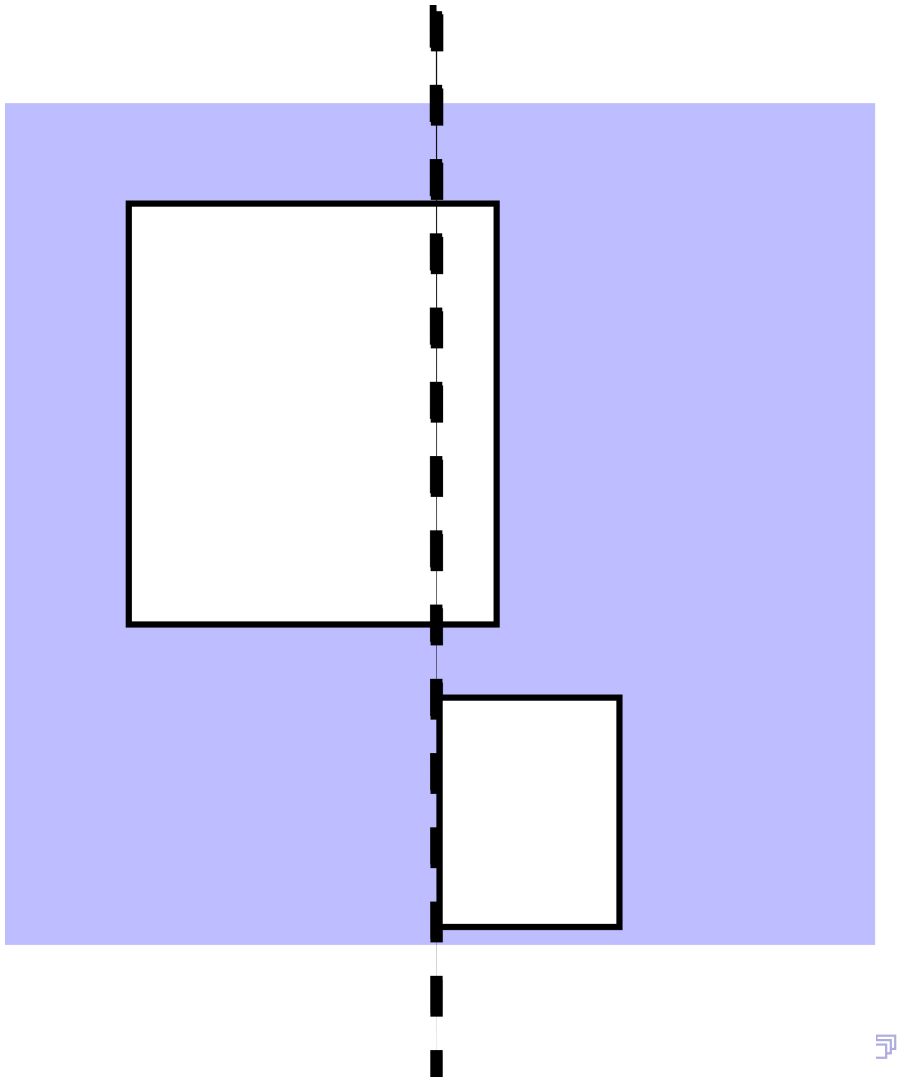
Пустой прямоугольник

- Мы ищем пустой подпрямоугольник максимального взвешенного периметра
- Здесь, кстати, возможно альтернативное решение за $O(nm)$, но останавливаться на нём не будем

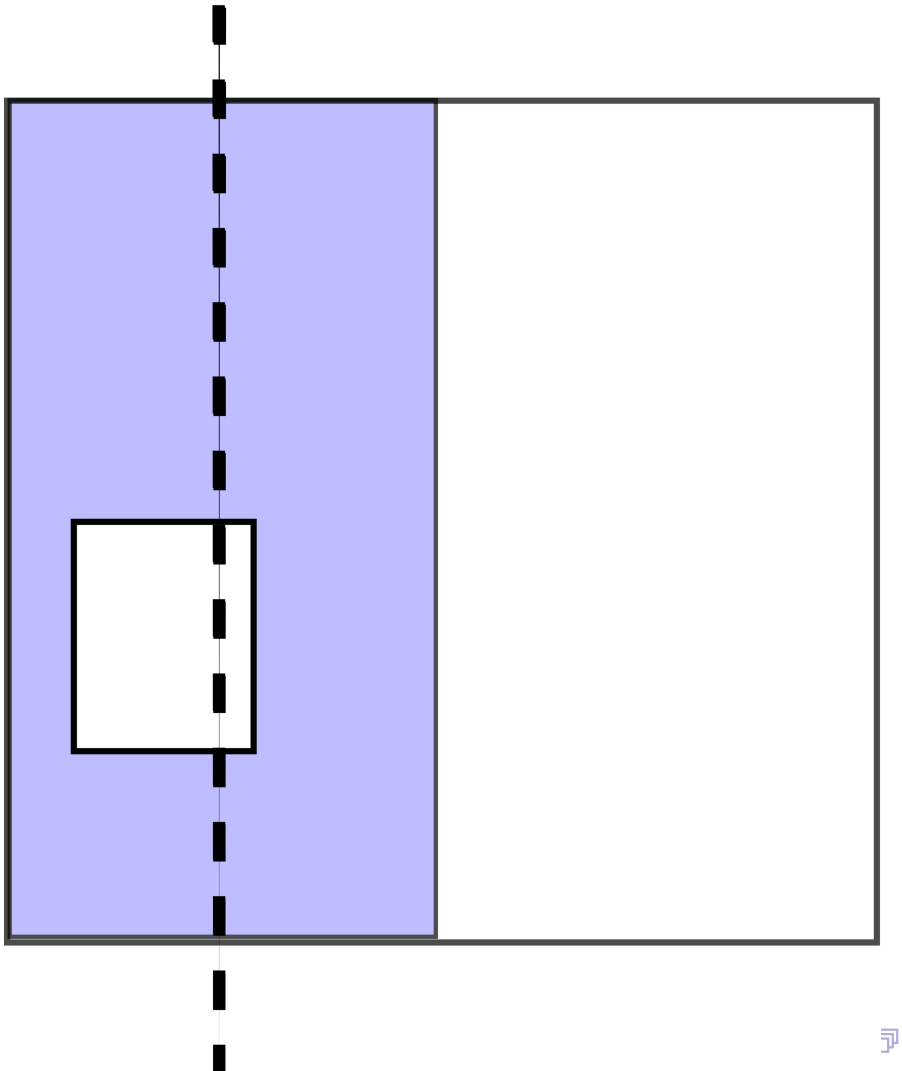
Divide & Conquer

- Разделим матрицу пополам вертикальной прямой
- Прямоугольники делятся на три вида:
  1. Лежащие слева от прямой
  2. Лежащие справа от прямой
  3. Пересекающие прямую
- Прямоугольники первого и второго вида обработаем, вызвавшись рекурсивно от левой и правой половины
- Осталось обработать прямоугольники, пересекающие вертикальную прямую

Divide & Conquer



Divide & Conquer

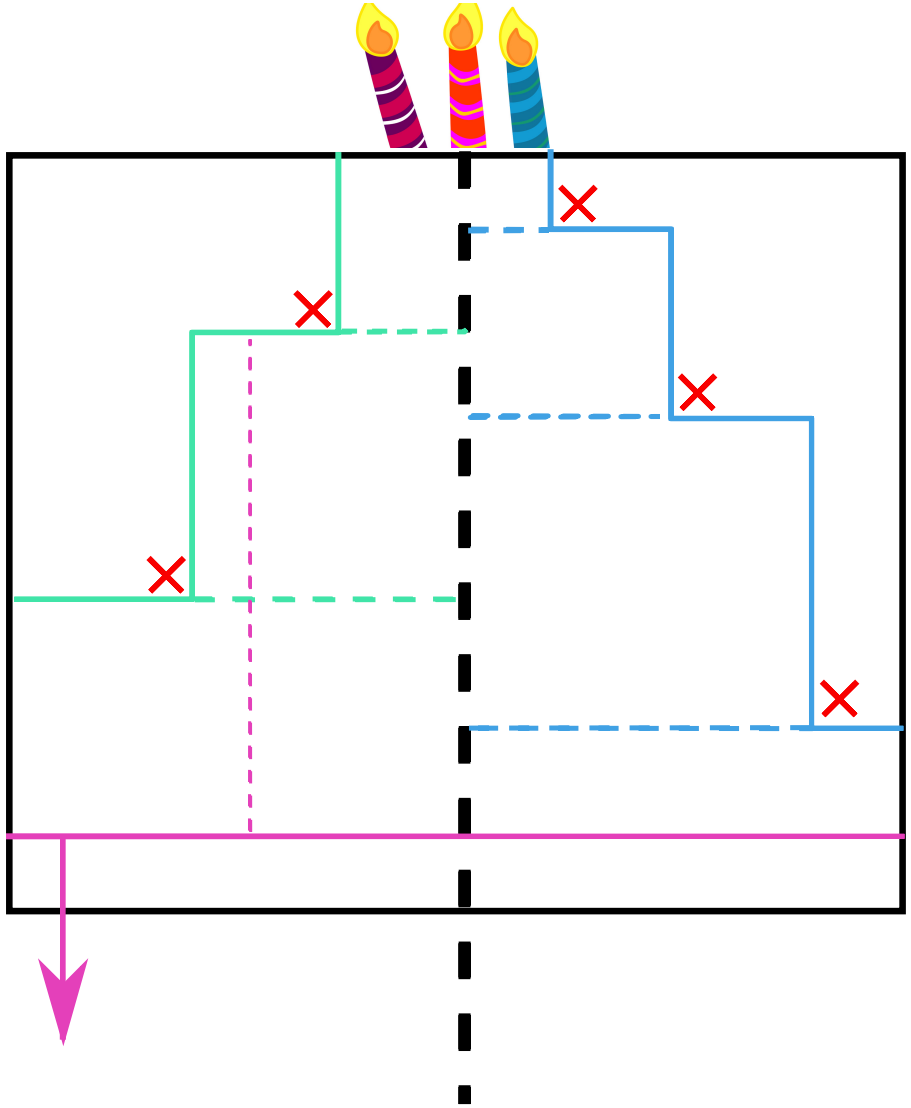


Divide & Conquer

- Разделим матрицу пополам вертикальной прямой
- Прямоугольники делятся на три вида:
  1. Лежащие слева от прямой
  2. Лежащие справа от прямой
  3. Пересекающие прямую
- Прямоугольники первого и второго вида обработаем, вызвавшись рекурсивно от левой и правой половины
- Осталось обработать прямоугольники, пересекающие вертикальную прямую

Divide & *Conquer*

- Будем перебирать нижнюю границу сверху вниз
- Слева и справа от прямой образуется по лесенке из заблокированных клеток, в которые будет упираться итоговый прямоугольник
- Между лесенками образуется «торт»



Divide & *Conquer*

- Мы оптимизируем $Sh[y_u] - Sh[y_d] + Sw[x_r] - Sw[x_l]$ — взвешенный периметр прямоугольника
- Здесь $Sw[x_r] - Sw[x_l]$ это ширина слоя торта на высоте u, переобозначим как $W[u]$
- $Sh[y_u]$ это константа, зависящая только от текущего положения нижней границы
- Значит, мы ищем $\min\limits_u (W[u] + Sh[y_u])$

Divide & *Conquer*

- Поддерживаем лесенки в стеках
- Поддерживаем ширину «торта» плюс расстояние до верхней границы в каждой строке в дереве отрезков
- Когда добавляется очередная заблокированная клетка, из какого-то из стеков выбрасываются нижние точки и добавляется одна новая
- На каждую выброшенную точку надо скорректировать ширину торта с помощью прибавления на отрезке

Divide & *Conquer*

- Сложность обработки одной вертикальной полосы — $O(k \log n)$, где k — количество точек в полосе
- Каждая точка попадает в $O(\log n)$ полос, поэтому итоговая сложность — $O(n \log^2 n)$, 80 – 90 баллов

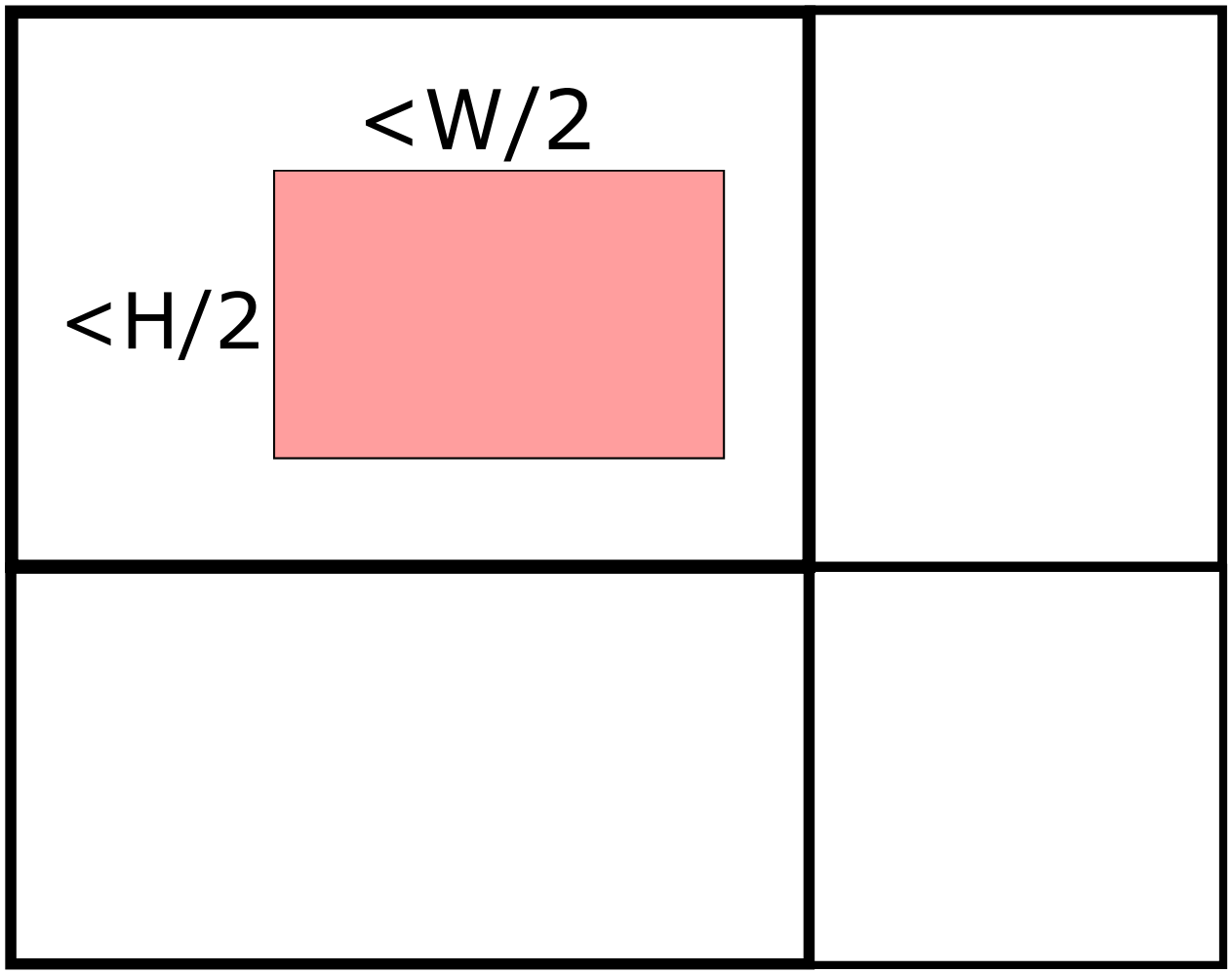
Наблюдение

- Обозначим за w вес первой последовательности и за h вес второй последовательности
- Заметим, что у нас заведомо есть ответ веса $\max(w, h)$ — просто берём целиком одну из последовательностей

Наблюдение

- Проведём вертикальную прямую, которая делит первую последовательность пополам по весу и горизонтальную прямую, которая вторую последовательность пополам по весу (округлив в произвольную сторону)
- Правильный ответ не может целиком содержаться внутри одной из образовавшихся четырёх частей, потому что иначе его вес строго меньше $\frac{w+h}{2} \leq \max(w, h)$

Наблюдение



- Нетривиальный ответ обязательно пересекает либо полученную вертикальную прямую, либо горизонтальную прямую
- Значит, достаточно два раза позвать conquer-процедуру из предыдущего решения
- Сложность решения — $O(n \log n)$
- 100 баллов

Вопросы?