Московская городская командная олимпиада 2020 года

Лига А

Разбор задач

Задача «Отборочный этап»

Автор задачи: Елена Андреева
Разработчик: Азат Исмагилов

Формальная постановка

- Дан массив пар чисел.
- Если отсортировать их по невозрастанию первого числа, то у 100-го места оно будет равно A, при этом у всех топ-100 второе число будет не меньше B.
- Если отсортировать их по невозрастанию второго числа, то у 100-го места оно будет равно C, при этом у всех топ-100 первое число будет не меньше D.
- Какая минимальная сумма обоих чисел будет у 100-го места, если всех отсортировать по невозрастанию сумм?

Решение

- Ответ не меньше $\max(A + B, C + D)$
- Почему?
  - Гарантированно найдётся 100 пар с суммой не меньше $A + B$.
  - Гарантированно найдётся 100 пар с суммой не меньше $C + D$.
  - Значит ответ не меньше каждого из этих чисел.
- Ответ равен $\max(A + B, C + D)$
- Почему?
  - Пусть у нас будет ровно 100 пар (A, B), и еще 100 пар (D, C).
  - 100-е место будет иметь подходящую сумму чисел.
  - Т.к. $B \le C$ и $D \le A$, пример будет подходить под условие.
- Решение за $O(1)$

Задача «Деление»

Автор задачи: Григорий Резников
Разработчик: Филипп Грибов

Формальная постановка

- Даны числа p и q
- Требуется найти максимальное число x, такое что p делится нацело на x, а x не делится нацело на q

Ключевая идея

- Пусть y = p / x.
- Если есть простое число a, такое что y делится на a, и q не делится на a, то можно домножить x на a, и увеличить ответ
- Если есть простые числа a и b, такие что y делится на a и b, и q делится на a и b, то хотя-бы одно число из x * a и x * b не делится на q, но является делителем p, и значит тоже подходит как ответ на задачу
- Значит y - это простой делитель q в какой-то степени

Решение

- Разложим число q на простые делители за асимптотику O(√q)
- Для каждого числа a, которое является простым делителем q, найдём минимальную степень k числа a, такую что p / a^k не делится на q. Такое число будет кандидатом на ответ
- Пользуясь ключевой идеей, знаем, что максимум из всех перебранных кандидатов на ответ и есть сам ответ на задачу.
- Итоговая асимптотика: O(√q + log p * log q)

Задача «Разбивай и суммируй»

Автор и разработчик задачи: Степан Стёпкин

Формальная постановка

- Дан массив a состоящий из $2n$ элементов
- Его всеми возможными способами разбивают на два массива p и q длины n
- Затем p сортируют по неубыванию, q по невозрастанию и считают сумму $|p_i - q_i|$ по всем i от 1 до n
- Нужно найти сумму всех таких сумм по модулю 998244353

Ключевая идея

- Как бы мы ни разбивали массив, стоимость разбиения всегда будет одинаковой
- Разобьем массив a размера $2n$ на два любых массива p и q размера n. Отсортируем p по неубыванию, а q по невозрастанию.
- Пронумеруем элементы массива a по возрастанию.
- За L обозначим множество элементов с номерами от 1 до n, а за R множество элементов с номерами от $n + 1$ до $2n$.

Доказательство

- Докажем, что любая разница в нашей сумме, будет разницей одного элемента из R и одного элемента из L.
- Допустим это не так, то есть найдется такой индекс i, что и p_i и q_i принадлежат одному множеству. Пусть это множество L.
- Все элементы с индексами меньшими или равными i в p принадлежат L
- Все элементы с индексами большими или равными i в q принадлежат L
- Тогда в L хотя бы i + n - (i - 1) = n + 1 элементов, а их должно быть ровно n. Противоречие.
- Для множества R доказательство аналогичное.

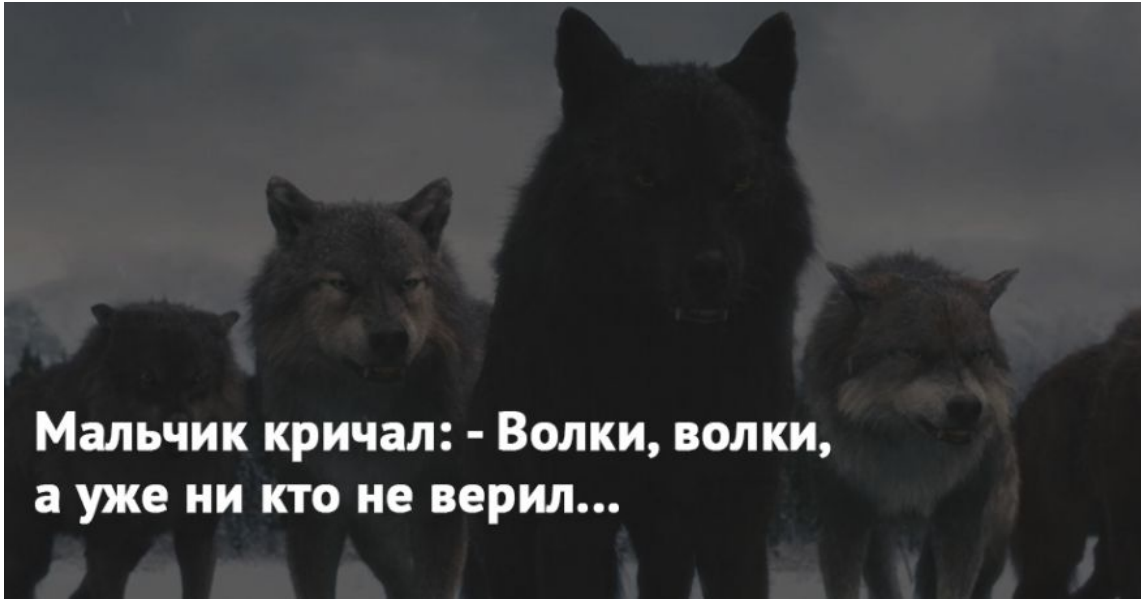
Решение

- Обозначим за sum сумму элементов множества R минус сумму элементов множества L

- Ответ на задачу sum * (количество разбиений на p и q) = sum * C_{2n}^{n}

- Итоговая асимптотика: O (n log n) (из-за сортировки массивов)

Задача «Фэйк Ньюз»



Автор и разработчик задачи: Константин Амеличев

Формальная постановка

- В городе существуют СМИ с параметрами *A[i]*
- Публикация новости выглядит так:
  - Для новости считается число публикаций *X*
  - СМИ с *A[i]* ≤ *X* публикуют новость
  - Процесс идет, пока публикации не прекратятся
- Надо обрабатывать запросы на добавление и удаление *A[i]* из множества
- После запроса требуется вывести итоговое *X*

Ключевая идея

- Перейдем к префиксным суммам на массиве подсчета $P[x]$
- Тогда процесс задается последовательностью прыжков
  - $x \to P[x] + 1$
- Прыжки идут до момента, когда $x = P[x] + 1$
- Достаточно найти момент остановки

Ключевое утверждение

- Рассмотрим массив $F[x] = P[x] - x + 1$
- Мы остановимся в каком-то его нуле
- Утверждается, что в самом первом
- Почему?
  - Потому что мы не смогли бы его перепрыгнуть
  - От противного. $P[x'] + 1 > x$; $x' < x$
  - Тогда $P[x'] - x + 1 > 0$
  - Но $P[x] \geq P[x']$, значит $P[x] - x + 1 \neq 0$
  - Противоречие

Ключевой алгоритм

- Будем поддерживать массив $F[x]$
- Тогда запросы превращаются в +/- 1 на суффиксе
- Надо искать первый ноль — слишком сложно
- До первого нуля все числа будут положительными, ведь $F[i + 1] - F[i] \geq -1$
- То есть, до первого отрицательного числа обязательно был ноль
- Значит, можно искать первое неположительное на префиксе с помощью спуска по дереву отрезков на минимум
- Решение за $O(n \log n)$

Задача «Тимбилдинг»

Автор задачи: Владислав Макеев
Разработчик: Александр Курилкин

Формальная постановка

- Дан неориентированный граф без петель и кратных ребер, у каждой вершины есть какой-то цвет от 1 до k.
- Посчитать количество пар цветов таких, что если оставить в графе только вершины этих двух цветов и все ребра между такими вершинами, то он будет являться двудольным.

Медленное решение

- Проверим подграфы, образованные каждым цветом, на двудольность (например, поиском в глубину). Это можно сделать за $O(n + m)$.
- Недвудольные цвета далее рассматривать не будем, они не могут участвовать ни в каких парах.
- Рассмотрим какой-то цвет x. Пусть из вершин такого цвета есть ребра в вершины цвета $y_1, y_2, \ldots, y_k$
- Проверим на двудольность (также поиском в глубину) пары (x, y_1), (x, y_2), …, (x, y_k), тем самым выяснив, какие цвета нельзя взять в пару с x. Остальные можно.
- Сделаем так для каждого цвета, и тем самым найдем ответ.

Анализ медленного решения

- Заметим, что каждое ребро, соединяющее различные цвета, мы посмотрим два раза.
- Проблема: ребра, соединяющие один цвет, которых может быть много, мы можем посмотреть до k раз.

Ключевая идея

- Критерий двудольности графа: отсутствие в нем нечетного цикла.
- Представим какую-то двудольную компоненту связности, образованную одним цветом.
- Если цикл в графе, образованном двумя цветами, проходил по этой компоненте, то нам неважно, как именно он это делал. Если он начал и закончил в одной и той же доле, то путь внутри этой компоненты имел четную длину, а если в разных, то нечетную.
- Это позволяет нам сжать такую компоненту в две вершины - по одной на каждую долю, и провести ребро между этими вершинами.
- Для каждого цвета сожмем таким образом все образованные им компоненты.

Ключевая идея

- В результате сжатия у нас появился новый граф, где каждая компонента связности представляет из себя либо одну вершину, либо две вершины, соединенные ребром.
- Будем делать так же, как в решении, разобранном ранее - проверять каждую пару цветов, соединенную ребрами, на двудольность, но теперь в сжатом графе.
- Пусть мы хотим проверить пару (x, y). Для каждого ребра между этими цветами в исходном графе проведем соответствующее ребро в сжатом графе между вершинами, которые соответствуют компонентам и их долям, которые соединяло ребро в исходном. После этого поиском в глубину проверим на двудольность получившийся граф, затем откатим изменения и сделаем все то же самое для других пар.

Анализ времени работы

- За сколько работает проверка одной пары цветов (x, y)?
- В сжатый граф мы добавили только ребра, соединяющие цвета x и y. При поиске в глубину будем запускать его только от компонент, которые были затронуты добавленными ребрами, потому что незатронутые компоненты никак не повлияют на двудольность, но их может быть много.
- При таком подходе поиск в глубину рассмотрит все добавленные ребра, а также какое-то количество ребер, соединяющие вершины одного цвета. Но таких ребер будет максимум в два раза больше, чем добавленных, потому что каждое добавленное ребро соединяет максимум две новых компоненты, а в каждой компоненте максимум одно ребро.

Анализ времени работы

- Проверка каждой пары цветов работает за количество ребер между этими цветами, а значит, суммарно при проверке всех пар мы потратим $O(m)$ времени!
- Итоговое время работы: $O(n + m)$ или $O(m \log n)$, в зависимости от реализации.

Задача «TV»

Автор и разработчик задачи: Роман Глазов

Формальная постановка

- Задано n телеканалов, на каждом идет интересная передача в какой-то временной отрезок
- Вы смотрите телевизор следующим образом: если на текущем телеканале идет интересная передача, то досматриваете ее до конца и переходить к телеканалу с большим номером, иначе просто переходите к телеканалу с большим номером. С последнего телеканала никуда не переходите и заканчиваете просмотр
- Есть m запросов, каждый запрос это момент времени и телеканал, с которого вы начинаете просмотр
- На каждый запрос нужно ответить, сколько суммарно времени вы потратите на просмотр телепередач

Подзадача

- Научимся отвечать в оффлайне на k запросов вида: задан телеканал i и момент времени t, найдите телеканал с минимальным номером $j > i$ такой, что в момент времени t на нем идет интересная передача
- Для этого будем идти сканирующей прямой по телеканалам и запросам в порядке увеличения индекса телеканала и поддерживать множество запросов, на которые еще не нашелся ответ
- Когда встречаем запрос - добавляем его в множество
- Когда встречаем телеканал - смотрим, какие запросы из множества попадают в его отрезок и удаляем их из множества
- Асимптотика подзадачи $O(k \log k)$

Основная задача

- Теперь, умея отвечать на описанные выше запросы, можно решить первоначальную задачу
- Для этого посчитаем динамику dp[x] равную времени, которое мы потратим на просмотр телепередач если начнем с телеканала x в момент времени, равное окончанию передачи на этом канале
- После этого для первоначальных m запросов мы можем узнать, на каком телеканале мы встретим первую интересную передачу и при помощи динамики этого телеканала узнаем ответ
- Итоговая асимптотика $O((n + m) \log (n + m))$

Задача «Парад во время чумы»

- БЫЛО СЛОЖНО, НО Я СДАЛА
- НЕ ЗРЯ ВСЮ НОЧЬ УЧИЛА
- И ШПОРЫ ПОМОГЛИ ТОЖЕ

Автор задачи: Елена Андреева

Разработчик: Владимир Романов

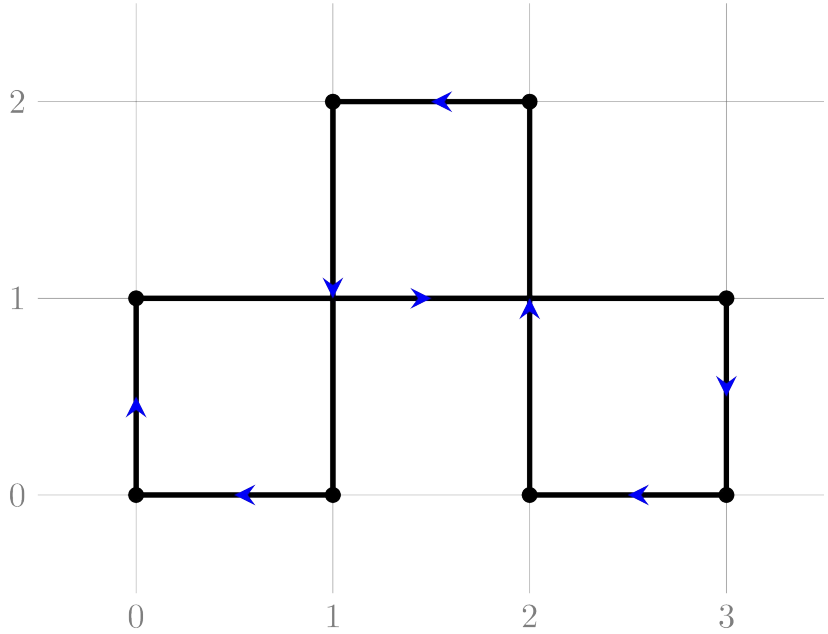
Формальная постановка

- Есть n строк по m чисел
- j-ое число в i-строке равно $a_{i, j}$
- Мы хотим выбрать в каждой строк по числу так, чтобы максимальный модуль разницы между соседними выбранными числам было минимальным

Ключевая идея

- Применим в данной задаче бинпоиск по ответу. Пусть M проверяемая величина.
- Воспользуемся методом динамического программирования, пусть dp[i][j] - можно ли построить шеренгу с разностью не превосходящей M из первый i строк и последним будет j-й элемент i-й строки.
- Для вычисления dp[i][j] нужно для каждого элемента понять есть ли элемент на прошлой строке, который отличается на не более M и при этом для него есть положительное состояние.
- Для этого можно применить метод двух указателей (предварительно отсортировав все строки) и воспользоваться префиксной суммой для нахождения суммы на отрезке.
- Итого O(nm log max_ans).

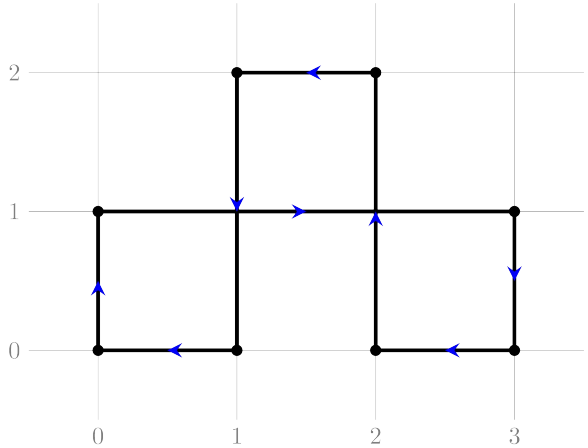
Задача «Прямоугольная ломаная»



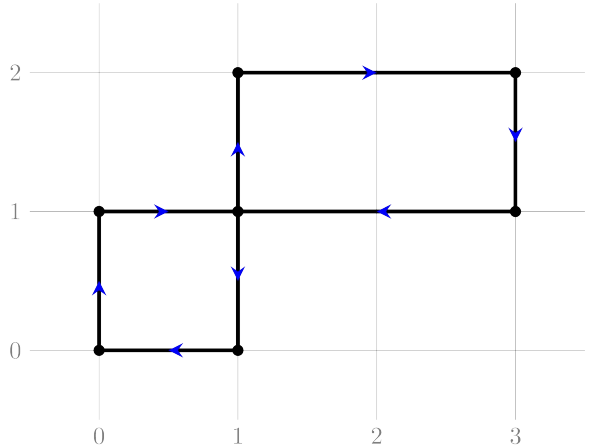
Автор задачи: Максим Ахмедов
Разработчик: Максим Деб Натх

Формальная постановка

- Дан набор из горизонтальных и вертикальных отрезков
- Проверить, можно ли из них составить замкнутую ломаную с самопересечениями только на концах отрезков:



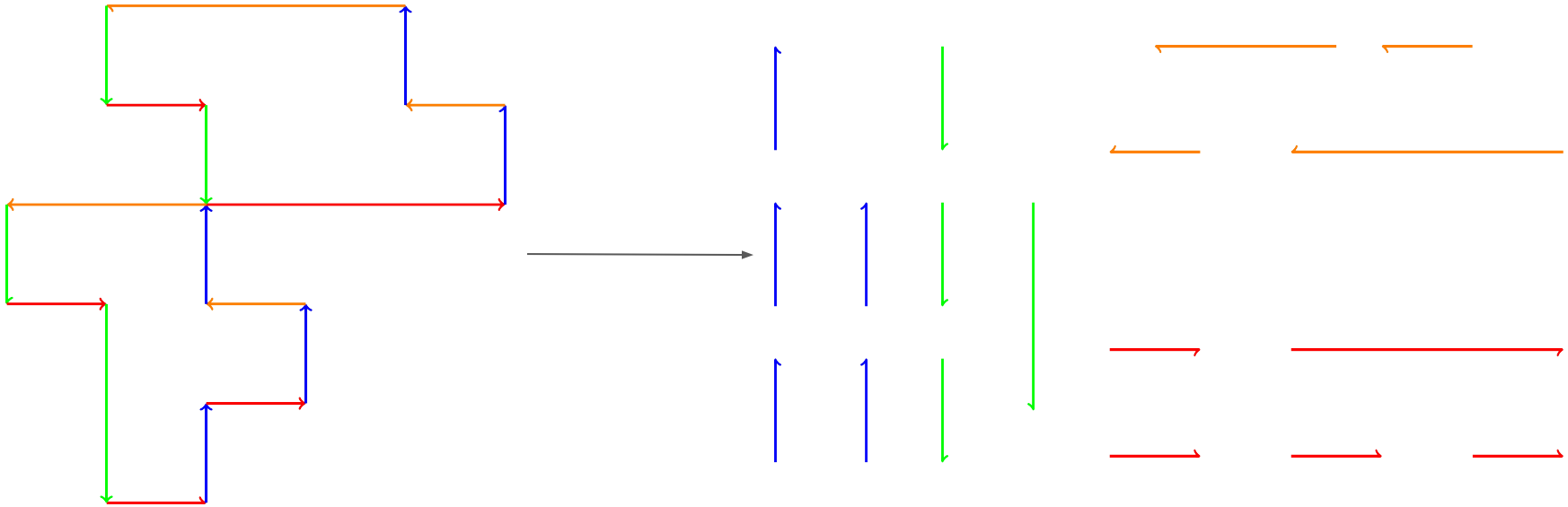
Не ок



Ок

Критерий корректности

- В корректной ломанной число вертикальных отрезков равно числу горизонтальных: $h = v$
- Горизонтальные отрезки можно разбить на идущие «вправо» и «влево»
- Вертикальные отрезки можно разбить на идущие «вверх» и вниз»:



Критерий корректности

- Очевидно, что суммарная длина отрезков «вверх» равна суммарной длине отрезков «вниз»
- Значит, множество горизонтальных длин можно разбить на два множества равной суммы
- Если разбить нельзя, то ответ *«Нет»*
- То же самое верно для вертикальных отрезков

Проверка

- Проверить, можно ли разбить множество на два множества одинаковой суммы можно с помощью решения задачи о рюкзаке методом динамического программирования:
- Сложность такой проверки будет $O(n^2C)$
    - Если разбиение какого-то из множеств не существует, ответ *Нет*
    - Иначе ответ *Да,* нужно научиться восстанавливать ответ.

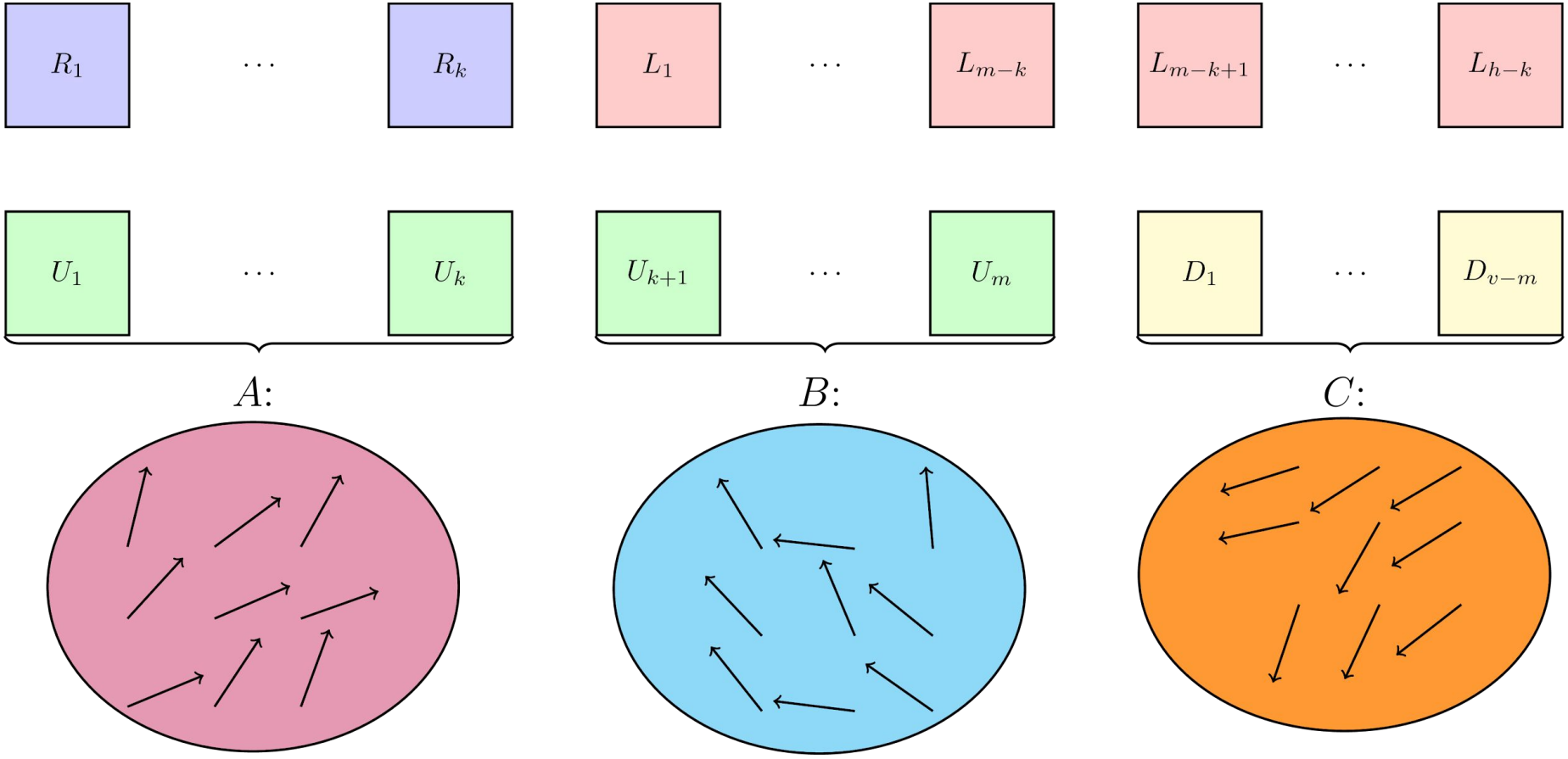
Построение ответа

- Пусть мы смогли разбить горизонтальные отрезки на два множества R и L ($|R| \le h/2 \le |L|$)
- Пусть мы смогли разбить горизонтальные отрезки на два множества D и U ($|D| \le v/2 \le |U|$)
- Тогда имеем, что $|R| \le |U|$
- Построим пример, в котором длины отрезков «вправо» будут равны множеству R, «влево»— множеству L, «вниз»— множеству D, «вверх»— множеству U
- Для этого все горизонтальные и вертикальные отрезки разобьём на пары

Построение ответа

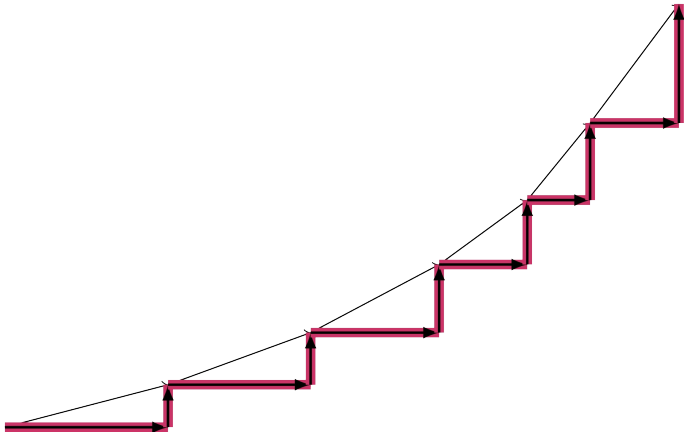
- В пару отрезку из *R* поставим какой-нибудь отрезок из *U*
- В пару отрезку из *L* поставим какой-нибудь из оставшихся отрезков (либо *U*, либо *D*)
- Из пары *(r, u)* сделаем вектор *(r, u)* — получим множество *A*
- Из пары *(l, u)* сделаем вектор *(-l, u)* — получим множество *B*
- Из пары *(l, d)* сделаем вектор *(-l, -d)* — получим множество *C*

Построение ответа



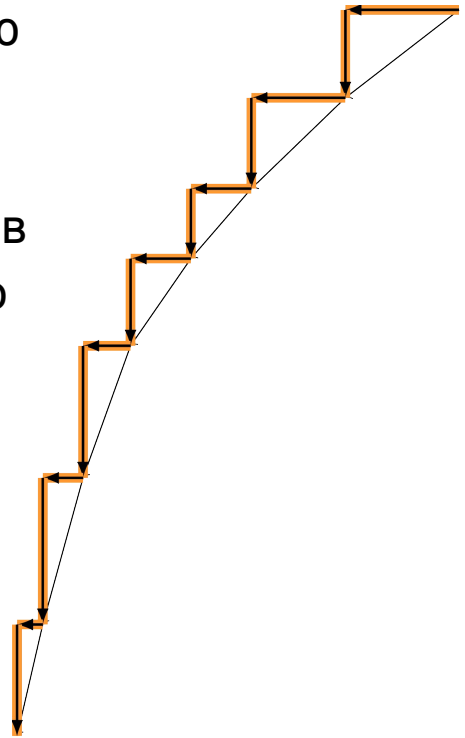
Построение ответа

- Составим из множества *A* выпуклую вниз ломаную (отсортируем вектора по полярному углу).
- Теперь заменим каждый из векторов этой ломаной на два — один вправо и один вверх:



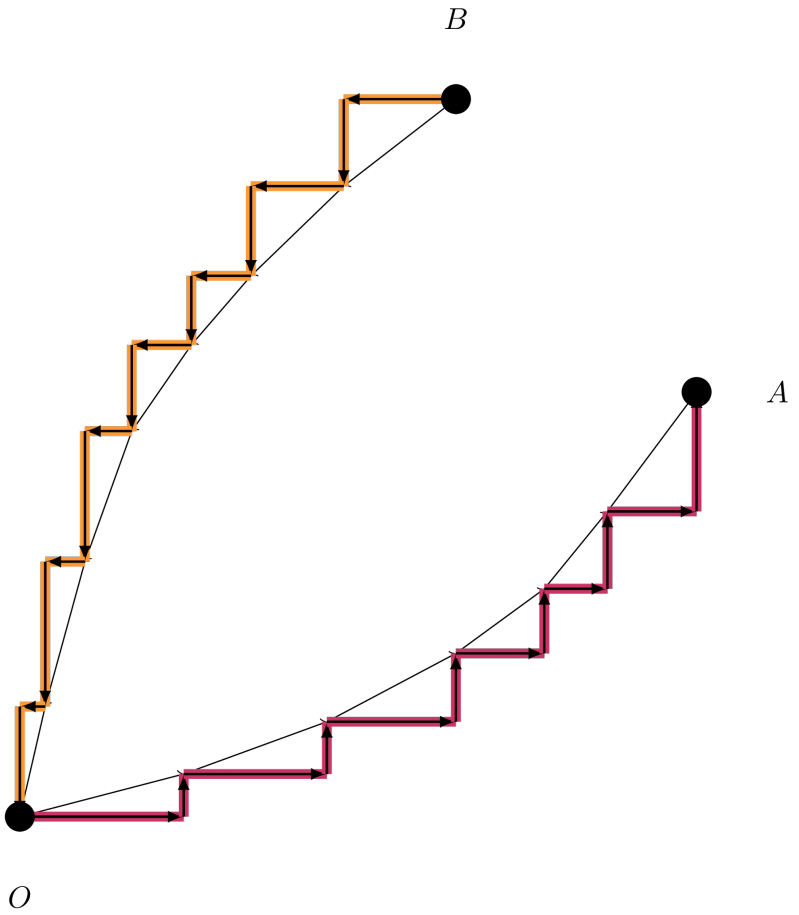
Построение ответа

- Составим из множества C выпуклую вверх ломаную (отсортируем вектора по полярному углу).
- Теперь заменим каждый из векторов этой ломаной на два — один влево и один вниз:



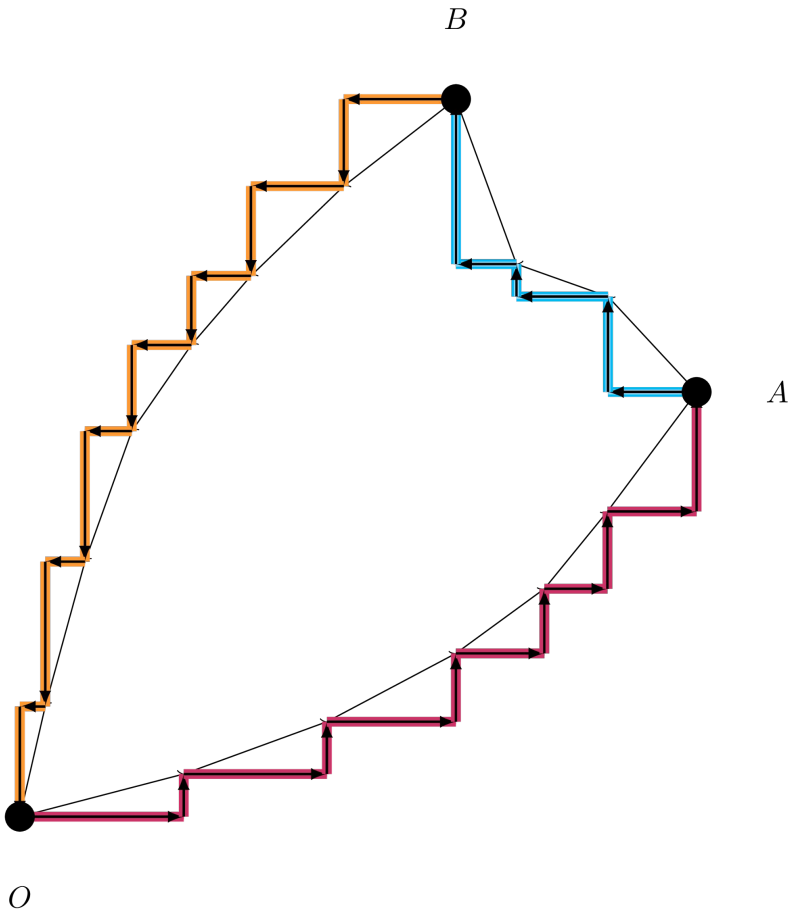
Построение ответа

- Объединим эти две ломаные, так чтобы они вторая заканчивалась там же, где начинается первая— в точке O
- Теперь осталось соединить точки A и B векторами из множества B



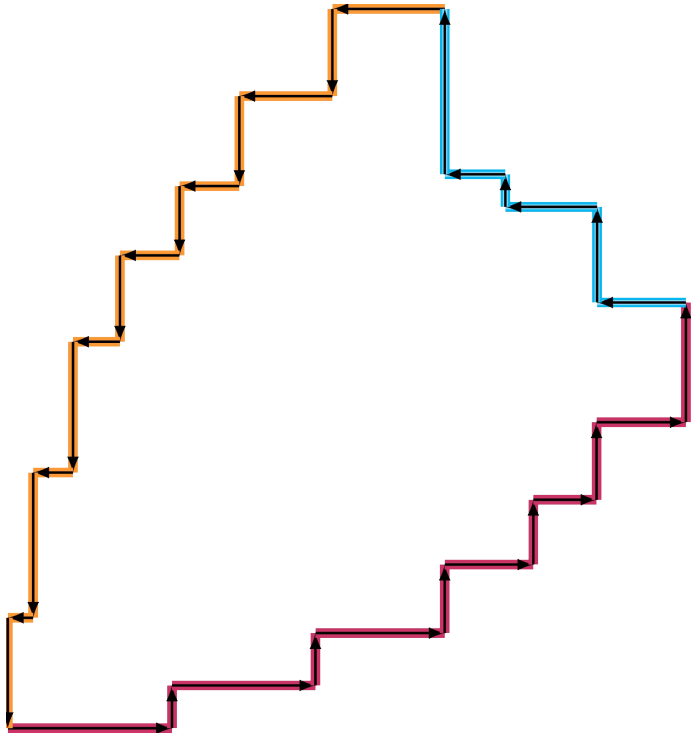
Построение ответа

- Возьмём все вектора *C* и построим из них ломаную из *A* в *B*
- Теперь заменим каждый вектор ломаной *BA* на один горизонтальный вектор влево и один вертикальный вверх



Построение ответа

- Нетрудно показать, что построенная ломаная будет удовлетворять всем условиям



Итоговое решение

- $O(n^2C)$ на разбиение множеств на две части с равной суммой
- $O(n \log n)$ на построение ответа

Задача «Сборы в поход»

Автор задачи: Григорий Резников
Разработчик: Николай Будин

Формальная постановка

- Даны m предметов, у каждого предмета известен тип и вес
- Нужно разложить предметы по минимальному количеству рюкзаков вместимостью s
- В одном рюкзаке не должно быть двух предметов одного типа
- $m \leq 23$

Решение

- Воспользуемся методом динамического программирования
- Пронумеруем предметы от 0 до $m - 1$ так, чтобы предметы одного типа шли подряд
- Состоянием будет пара из двух чисел: $mask$ и i
  - $mask$ обозначает множество набранных предметов
  - i обозначает номер текущего предмета, либо равно дополнительному значению m
- Значением будет пара (количество использованных рюкзаков, занятое место в последнем рюкзаке), его мы будем минимизировать как пару
- Обозначим $next[i]$ – номер ближайшего после i предмета с типом отличным от типа i (или m, если такого предмета нет)
- Рассмотрим следующие переходы:
  - $dp[mask][i] \longrightarrow dp[mask][i + 1]$, где $0 \le i < m$, пропускаем i-й предмет
  - $dp[mask][i] \longrightarrow dp[mask \cup \{i\}][next[i]]$, где $0 \le i < m$, $i \notin mask$, в текущий рюкзак добавляем i-й предмет
  - $dp[mask][m] \longrightarrow dp[mask \cup \{i\}][next[i]]$, где $0 \le i < m$, $i \notin mask$, добавляем новый пустой рюкзак, кладем туда i-й предмет

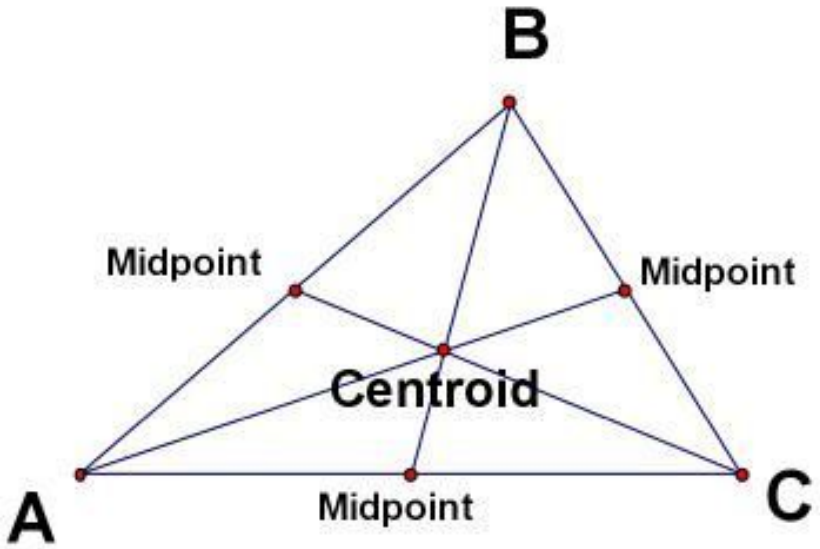
Решение

- Несложно заметить, что для любого корректного разбиения множества предметов по рюкзакам, будет существовать последовательность переходов, соответствующая такому разбиению
- Поэтому, найденный ответ будет не хуже оптимального
- С другой стороны, любая последовательность переходов соответствует корректному разбиению предметов на множества, поэтому ответ будет не лучше оптимального

Время работы и память

- Асимптотика времени работы равна $O(2^m m)$
- Занимаемая память также равна $O(2^m m)$
- Для того, чтобы решение уложилось в ограничение по времени, могло понадобиться дополнительно уменьшить используемую память
- Заранее упорядочим все множества в порядке возрастания количества предметов в них
- Тогда в каждый момент времени достаточно хранить значения для не более чем $S \cdot 2$ последовательных множеств, где S – максимальное количество различных множеств с одинаковым количеством элементов ($S = 1352078$)
- Тогда занимаемая нами память будет $O(2^m \sqrt{m})$

Задача «Найти вершину»



Автор задачи: Владислав Макеев
Разработчик: Ильдар Гайнуллин

Формальная постановка

- В дереве загадали одну из вершин
- Можно спросить про любое ребро, по какую сторону от этого ребра находится вершина
- Найти вершину за минимальное возможное число запросов

Ключевая идея

- Какое-то ребро будет спрошено первым, пометим его числом 0.
- Дальше рассмотрим аналогичные раскраски (рекурсивно) по две стороны от этого ребра, но увеличим веса этих ребер на один (так чтобы был только один ноль)
- Эта раскраска соответствует стратегии, и если k=максимальный вес ребра в ней, то стратегия соответствует поиску вершины за k+1 запрос в худшем случае
- У этой раскраски есть одно свойство которое всегда позволяет нам однозначно определить ход: на пути между любыми двумя ребрами одного цвета есть ребро меньшего цвета.
- И любая раскраска с таким свойством является корректной стратегией!

Преобразим раскраску

- Чтобы нам было проще в дальнейшем, “Инвертируем” веса в раскраске, пометим самое первое ребро цветом $k - 1$, а дальше аналогично тому, как строилась раскраска по стратегии ранее, возьмем раскраски для поддеревьев, но вычитаем один из весов двух компонент (но чтобы веса остались неотрицательными)
- Теперь нам нужно найти раскраску с минимальным максимальным весом, чтобы на пути между любыми двумя ребрами одного цвета было ребро большего веса.

Поиск оптимальной раскраски

- Будем искать эту раскраску динамикой по поддеревьям.
- Для фиксированной раскраски поддерева v, посмотрим какие цвета ‘видны’ если смотреть с v вниз.
- Скажем, что цвет видно, если есть ребро такого цвета, что на пути от этого ребра до v нет ребер большего цвета.
- Введем ‘потенциал’ раскраски: минимальная сумма 2^(c_1) + 2^(c_2) + … + 2^(c_k), где c_i -- видимые цвета. Обратите внимание, что этот потенциал это длинное двоичное число! Так как ответ может быть большим.
- **Утверждение**: от поддерева интересует только раскраска с минимальным потенциалом.

Поиск оптимальной раскраски

- Пусть поддеревья фиксированной вершины v уже покрашены в раскраски с минимальными потенциалами.
- Тогда нам нужно выбрать веса ребер исходящих вниз из v, чтобы после привешивания этих ребер к поддеревьям вершины v, разные поддеревья не имели общих видимых цветов.
- Можно покрасить ребро из вершины v в ее сына u в цвет c, если цвет c не является видимым из вершины u. После этого из множества видимых цветов пропадут все веса меньше c, но добавится цвет c
- Нам нужно поменять раскраски всех поддеревьев, чтобы при сложении потенциалов не возникло переносов (это будет соответствовать тому, что в разных поддеревьях нет общих видимых цветов). При этом условии нужно минимизировать итоговую сумму - потенциал v.

Сложение без переносов

- Из такой постановки ‘доказательство’ предыдущего утверждения становится понятным: при большем потенциале возможные ходы строго не лучше.
- Нужно решить задачу: дан массива длинных двоичных чисел a_1, a_2, .., a_k. Нужно найти массив с минимальной суммой b_1, b_2, …, b_k, что b_i > a_i, и при сложении b_1 + b_2 + … + b_k не возникнет переносов.
- *Легендарная* задача с РОИ 2018! Можно решить даже за nlogn. Но в задаче достаточно было несложного решения за n^3. Мы не будем здесь приводить детали решений этой подзадачи, чтобы не спойлерить :)
- Итого нам нужно решить эту задачу n раз, чтобы найти веса исходящих ребер из очередной вершины, и мы получаем асимптотику: O(n * T(n)) = O(n^2 log n ... n^4).