Всероссийская олимпиада школьников по информатике

Уфа

23–29 марта 2013 года

Задача «Хоккей на Урале»

Задача «Хоккей на Урале»

Задача «Хоккей на Урале»

Задача «Хоккей на Урале»

- Идея задачи — Елена Андреева
- Подготовка тестов — Павел Кротков
- Разбор задачи — Елена Андреева

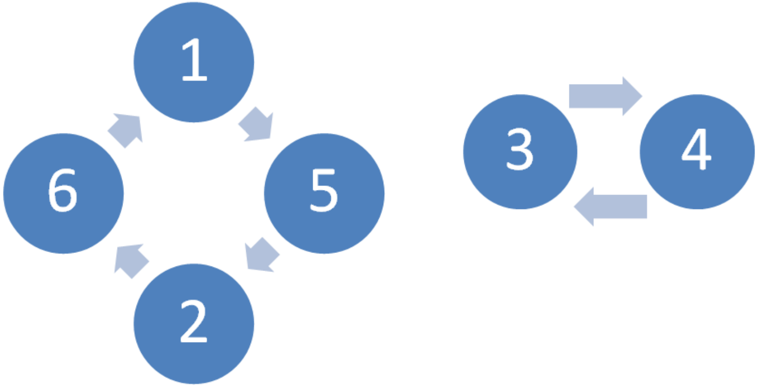
Постановка задачи

- Были проведены два тура хоккейного турнира. Каждая из N команд сыграла в каждом из туров
- Требуется выбрать K команд, никакие две из которых не встречались в рамках первых двух туров

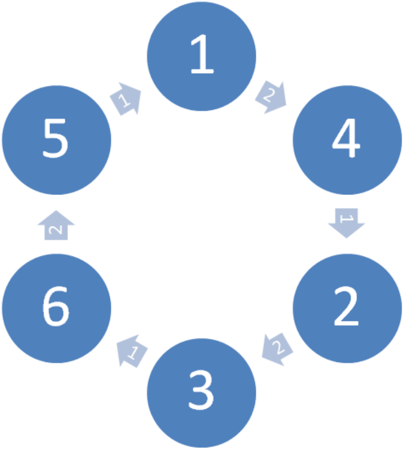
Математическая модель

- Построим граф, вершинами которого являются N команд, а ребрами — сыгранные ими партии. Тогда каждая вершина будет соединена ребрами с двумя другими
- Такое возможно, только если граф представляет собой набор циклов

Математическая модель



Все циклы чётной длины



Решение

- В каждом цикле выберем вершины (номера команд) через одну. Они нам заведомо подходят, и больше выбрать невозможно.
- Таким образом всегда можно выбрать $K \leqslant N/2$ команд.

Решение

- В качестве первой команды можно взять любую
- Если хранить граф в массиве размером $[N][2]$, в котором для каждой вершины будут храниться два её соседа, то мы быстро будем находить следующую вершину в цикле
- Обход вершин можно организовать с помощью обхода графа в глубину
- Общая сложность решения пропорциональна числу вершин в графе N

Вопросы?

Задача «Робот-сборщик»

Задача «Робот-сборщик»

Математическая формулировка

Входные данные: число K, строка $s_1 s_2 \ldots s_N$.
Результат: количество «подходящих» отрезков от L до R, на которых $s_i = s_{i+K}, i = L \ldots R - K$, $R - L \geqslant K$.

Решение: подзадача 1

Перебрать все пары L и R. Для каждой пары проверить условие $s_i = s_{i+K}$ в цикле по i. Время: $O(N^3)$. Проходит тесты для $N \leqslant 100$, набирает 30 баллов.

Решение: подзадача 1

Лишние действия: при переходе от R к $R+1$ условия $s_i = s_{i+K}$ для $i = L \dots R-K$ проверяются повторно.

Решение: подзадача 2

Перебрать все значения L.
Для каждого L перебрать все R от $L + K$ до N.
Для каждого R: если $s_{R-K} = s_R$, то отрезок $L \dots R$ считать подходящим (поскольку условия $s_i = s_{i+K}$ для $i = L \dots R-1$ проверены на предыдущих итерациях), иначе перейти к следующему L.
Время: $O(N^2)$. Проходит тесты для $N \leqslant 5000$, набирает 60 баллов.

Решение: подзадача 2

Лишние действия: при переходе от L к $L+1$ условия $s_i = s_{i+K}$ для $i = L \ldots R-K$ проверяются повторно.

Решение: подзадача 3

Установить $R \leftarrow 1$. Для каждого L от 1 до $N - K$:

- Найти максимальное значение $R' \geqslant R$, такое, что $s_{r-K} = s_r$ для всех $r = R \ldots R'$, $R \geqslant L + K$.
- Все отрезки $L \ldots r$, где $r = L + K \ldots R'$, считать подходящими (поскольку условия $s_i = s_{i+K}$ для $i = L \ldots R$ проверены на предыдущих итерациях), их количество равно $R' - L - K - 1$.
- Установить $R \leftarrow R'$

Время: $O(N)$. Проходит тесты для $N \leqslant 200\,000$

Решение: пример кода

```
int R = 0;
for (int L = 0; L < N; R++) {
    R = max(L + K, R);
    while (R < N && s[R - K] == s[K]) R++;
    ans += R - L - K;
}
```

Нумерация символов с 0, R и R' хранятся в одной переменной.

Решение: экзотика

Условие $s_i = s_{i+K}, i = L \dots R - K$, $R - L \geqslant K$ эквивалентно условию $s[L \dots R - K] = s[L + K \dots R]$, где $s[i \dots j]$ — подстрока с i-го по j-й символ.

Перебрать все значения L.

Для каждого L найти максимальное R от $L + 1$ до N, для которого $s[L \dots R - K] = s[L + K \dots R]$ двоичным поиском.

Для быстрого сравнения подстрок использовать хеш-функцию.

Время: $O(N \log N)$. Проходит тесты для $N \leq 200\,000$, набирает 100 баллов

Вопросы?

Задача «Графический редактор «Хамелеон»

Задача «Графический редактор «Хамелеон»

Задача «Графический редактор «Хамелеон»

Задача «Графический редактор «Хамелеон»

- Идея задачи — Михаил Дворкин
- Подготовка тестов — Павел Кунявский и Никита Иоффе
- Визуализатор — Георгий Корнеев
- Разбор задачи — Павел Кунявский

Формулировка

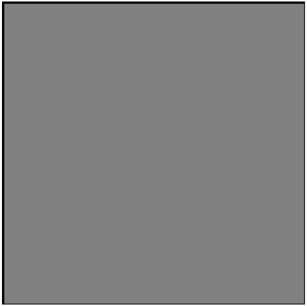
- Задано квадратное поле, которое требуется покрасить по данной картинке.
- Можно перемещаться в четырёх направлениях.
- При перемещении можно либо перекрасить клетку в цвет хамелеона, либо хамелеона в цвет клетки.

Результат:

- Последовательность ходов, приводящая к необходимой покраске.

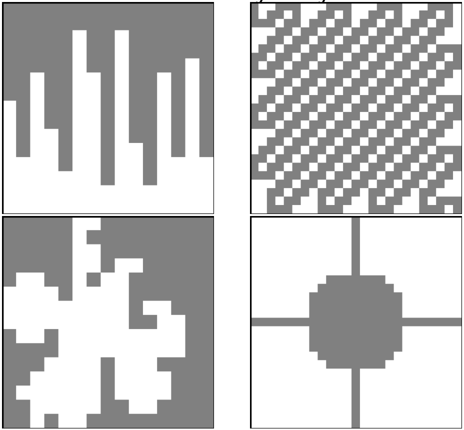
Структура тестов

- Первый тест — черный квадрат.
    - Обойти квадрат как угодно



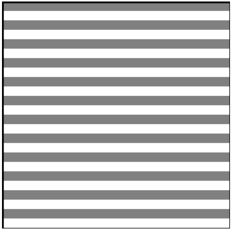
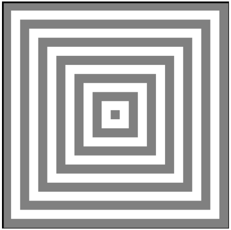
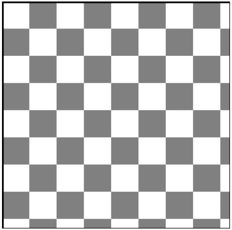
Структура тестов

- Следующие 4 теста — связные области
    - Обойти область в глубину



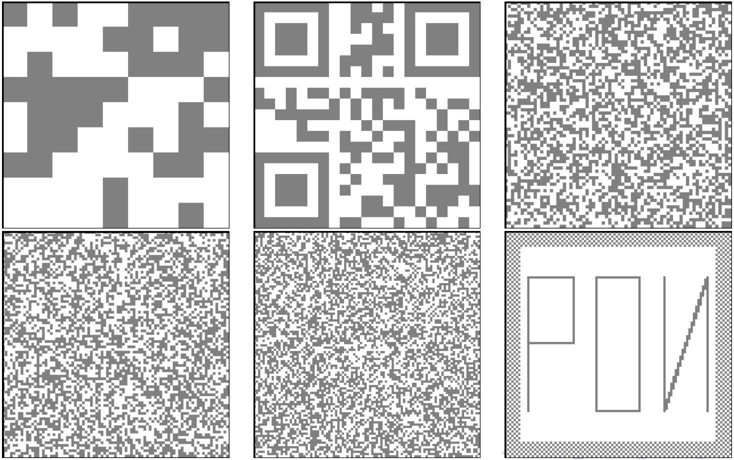
Структура тестов

- Следующие 3 теста — конструкции
  - Можно написать специальное решение

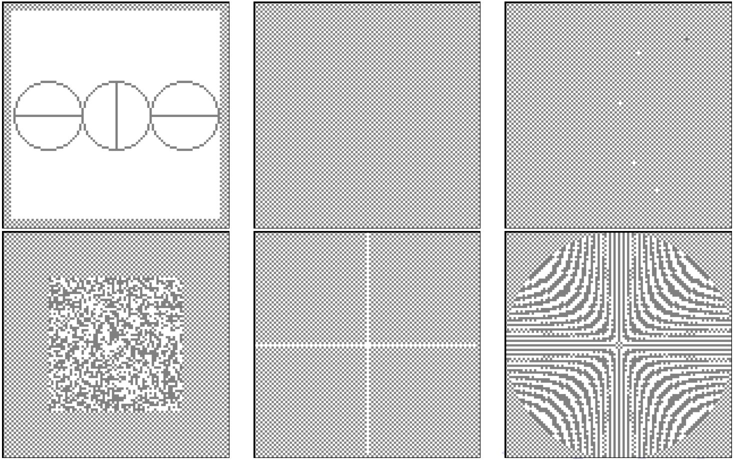
Структура тестов

Остальные тесты решаются общим методом



Структура тестов

Остальные тесты решаются общим методом



Решение

- Докажем, что покрасить шахматную доску нельзя
- Для этого посмотрим на последнее перекрашивание. Оно создает две клетки одинакового цвета рядом.
- Значит, в любой покраске такие есть, а в шахматной нет.

Решение

- Докажем, что всё остальное покрасить можно.

- Выберем две соседние клетки, которые должны быть покрашены в одинаковый цвет. Назовем выбранную пару клеток *базой*.
- Покрасим две клетки базы в разные цвета.
- Для определенности будем считать базу горизонтальной, второй случай симметричен.

Решение

- Будем красить доску отдельно снизу от базы, отдельно сверху. Рассмотрим нижнюю часть.
- Будем красить строки снизу вверх. Строку красим отдельно слева от базы, отдельно справа.
- Обе части будем красить от края к базе.
- Чтобы покрасить клетку, надо переместиться на базу в клетку с нужным цветом и вернуться, перекрашивая.

Решение

- Таким образом получится покрасить всё, кроме части вокруг базы.
- Её можно разобрать руками. Для этого может понадобиться испортить базу.
- Таким образом получено решение, делающее N^3 ходов.

Решение

- Для того, чтобы получить квадратичное решение, надо не ходить на базу для каждой клетки.
- Например для этого можно покрасить все чётные строки в один цвет, а все нечётные — в другой за один проход по полю.

Решение

- После этого можно разносить цвет, проходя на строку ближе к базе и перенося другой цвет на очередную строку.
- При этом не получится покрасить последнюю строку, но её можно покрасить старым алгоритмом за квадрат.

Оптимизации

- Такое решение уже должно укладываться в $5 \cdot N^2$ ходов.
- Строки вокруг базы можно покрасить аналогично, проходя по столбцам.

Оптимизации

- Такой алгоритм уже занимает $\frac{1}{2} \cdot N^2$ ходов на покраску клеток через строку. При проходе назад мы тратим еще ходов не более $3 \cdot N^2$.
- Применим такую оптимизацию: если несколько подряд клеток не того цвета, будем красить их за один проход. Тогда мы будем тратить три хода на не более чем половину клеток. Это уменьшит длину решения на N^2 ходов.

Оптимизации

- Тогда решение займет $2.5 \cdot N^2$ ходов. Неучтённые линейные части занимают меньше $\frac{1}{2} \cdot N^2$.

Вопросы?

Задача «Древние династии»

Задача «Древние династии»

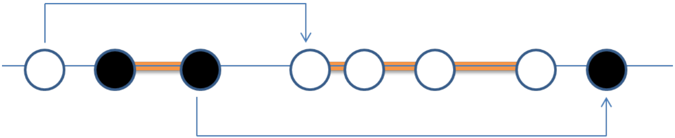
Задача «Древние династии»

Задача «Древние династии»

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

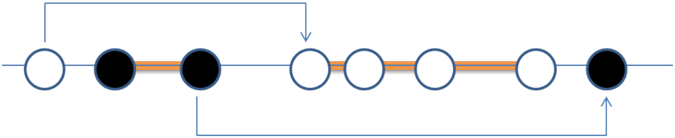
Математическая формулировка

- На прямой отмечено n различных точек.
- Необходимо покрасить их в два цвета (белый и черный) так, чтобы для каждой точки ближайшая точка того же цвета находилась на расстоянии в пределах от $[l_i, r_i]$.
- Требуется минимизировать количество пар соседних точек одного цвета



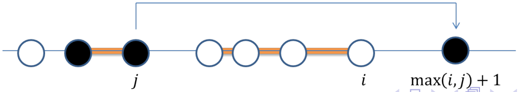
Простое решение первой подзадачи

- Будем перебирать все возможные варианты покраски точек (всего 2^n вариантов).
- Для каждого варианта нетрудно проверить, является ли он корректным.
- Среди корректных необходимо выбрать тот, который минимизирует искомую величину
- 20 баллов



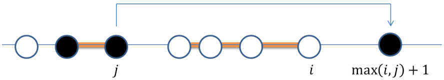
Динамическое программирование

- Необходимо знать лишь те моменты, в которые в последний раз встречался тот или иной цвет
- Пусть $ans_{i,j}$ обозначает минимальное количество пар соседних точек одного цвета в корректной раскраске отрезка $[1, max(i,j)]$, где i — последний момент, когда точка была окрашена в белый цвет, j — последняя точка, окрашенная в черный цвет



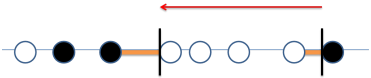
Переход

- Переход: будем выбирать, в какой именно цвет стоит покрасить элемент $max(i, j) + 1$.
- Допустим, мы хотим покрасить её в черный цвет. Необходимо проверить, что $x_{max(i,j)+1} - x_j$ лежит в пределах $[l_1, r_1]$, после этого попробовать обновить ответ для состояния $ans_{i,max(i,j)+1}$.
- $O(n^2)$ состояний, $O(n^2)$ переходов, $O(n^2)$ памяти, 40 баллов



Другое динамическое программирование

- Выберем в качестве состояния момент последний момент смены цвета
- Будем перебирать, где произошла последняя смена состояния цвета ранее этого состояния
- Проверим, что новый покрашенный отрезок действительно можно покрасить (между соседними точками отрезка подходящие для данного цвета расстояния), и что можно прыгнуть в предыдущую точку из нашей



Оптимизация перехода

- $O(n)$ состояний, $O(n^2)$ переходов, каждый переход за $O(n)$, $O(n)$ памяти
- Итоговое время работы $O(n^3)$, 20 баллов
- Перебирая переходы справа налево, можно проверять отрезок на корректность по ходу
- $O(1)$ на один переход, итоговое время работы $O(n^2)$, $O(n)$ памяти
- 60 баллов

Непрерывные отрезки

- Задача равносильна максимизации количества пар точек разного цвета
- Допустимые клетки, в которые мы можем перейти из текущей, представляют собой непрерывный отрезок
- На этом отрезке необходимо выбирать максимум
- Используя дерево интервалов, получаем $\log n$ на один переход, итоговое время работы $O(n \log n)$ и 80 баллов

Непрерывные отрезки и сдвиг

- При переходе к следующему состоянию, границы допустимого отрезка двигаются только вправо
- Границы этого отрезка можно поддерживать с помощью метода двух указателей
- Для отрезка, который двигается вправо («окно») мы можем решать задачу о поиске максимума за $O(n)$ в сумме

Минимум в «окне»

- Для этого будем поддерживать **deque**, в котором будем держать текущих кандидатов для максимума.
- При добавлении элемента в конец (сдвиге правой границы) будем выбрасывать элементы с конца, пока они меньше текущего.
- При сдвиге левой границы просто выбрасываем элемент с начала (если он там есть)
- $O(n)$ и заслуженные 100 баллов.

Вопросы?

Задача «Звездный путь»

Задача «Звездный путь»

Задача «Звездный путь»

Задача «Звездный путь»

- Идея задачи — Павел Кротков
- Подготовка тестов — Никита Иоффе, Юрий Петров
- Разбор задачи — Павел Кротков

Формулировка задачи

- Космическая экспедиция последовательно посещает N планет
- На каждой планете есть заправка с горючим определенного типа
- Заправившись на какой-то планете, экспедиция может долететь только до следующей планеты с горючим такого же типа
- Посетить все планеты за минимальное количество заправок

Начало решения

- Для каждой планеты определим `next[i]` как номер следующей планеты с горючим такого же типа
- Заметим, что на первой планете мы всегда заправляемся
- Это значит, что мы точно можем долететь до `next[1]`
- На какой-то из планет с номерами `[2, next[1]]` нам точно придется заправиться

Основная идея

- Заметим, что можно заправиться на планете, у которой значение `next` максимально
- Если заправиться на любой другой, в дальнейшем будет меньше выбора
- Из этой идеи получается решение

Формализация решения

- Каждый раз храним два последних города, в которых мы заправлялись
- Обозначим их `k` и `j`
- Следующий город, в котором мы заправимся — город с максимальным значением `next` на отрезке `[next[k], next[j]]`

Поиск значений `next`

- Пройдем по массиву с типами горючего с конца
- Для каждого типа будем хранить его самое левое встреченное вхождение
- Проходя очередную планету, будем ставить для нее правильное значение `next` и обновлять самое левое вхождение

Поиск максимумов в массиве next

- Можно реализовать за $\mathcal{O}(\log N)$ с помощью дерева отрезков
- Можно заметить, что суммарная длина всех отрезков не превышает $2 \times N$, и искать пробегом по массиву
- Решение будет работать или за $\mathcal{O}(N \times \log N)$, или за $\mathcal{O}(N)$

Частичное решение (50 баллов)

- Используем динамическое програмирование
- `d[i]` — минимальное количество заправок, за которое можно добраться до планеты i
- `d[i]` равно минимуму в массиве `d` по всем планетам `j` таким, что `j` $<$ `i`, `next[j]` $\geqslant$ `i` плюс один
- Решение работает за $\mathcal{O}(N^2)$
- Его можно доработать до правильного

Вопросы?

Задача «Морской бой»

Задача «Морской бой»

Задача «Морской бой»

Задача «Морской бой»

- Идея задачи — Жюри
- Подготовка тестов — Павел Кротков, Никита Иоффе
- Разбор задачи — Никита Иоффе

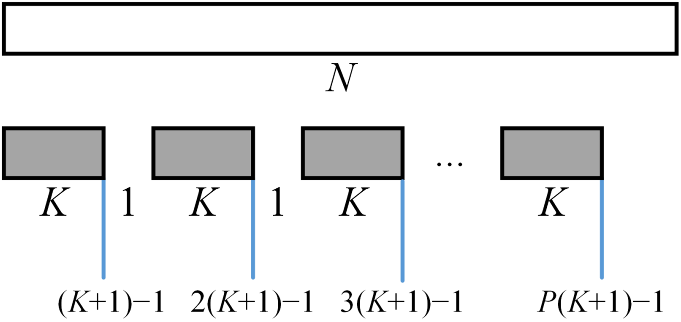
Формулировка

Входные данные:

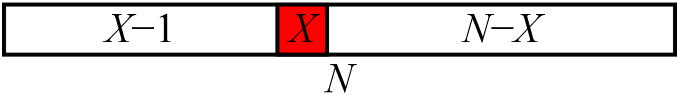
- Дано поле $1 \times N$, известно, что на этом поле можно расставить T кораблей размером $1 \times K$
- В клетки поля производятся выстрелы, ни в какую клетку не производится два выстрела
- После каждого выстрела требуется определить, можно ли на поле расставить T кораблей. Если корабли можно расставить, то следует вывести количество клеток, в которых гарантировано находится корабль.

Решение

- Рассмотрим отрезок длины N
- В отрезок помещается $P = \left\lfloor \frac{N+1}{K+1} \right\rfloor$ кораблей



Решение



- Рассмотрим выстрел в позицию S
- $A = \left\lfloor \frac{S}{K+1} \right\rfloor$ — максимальное количество кораблей слева
- $B = \left\lfloor \frac{N-S+1}{K+1} \right\rfloor$ — максимальное количество кораблей справа
- Если $A + B < P$, то в позиции S всегда находится корабль

Решение

- Количество свободных клеток $E = N - A \times K - (P - A) \times K = N - P \times K$
- Для одного корабля количество точно занятых им клеток $K - E$
- Количество точек, в которых гарантировано находится корабль — $R = P \times (K - E)$

Решение

- Будем решать задачу для каждого отрезка отдельно
- Пусть $P_{all} = \sum_{i=1}^{c} P_i$
- Пусть $R_{all} = \sum_{i=1}^{c} R_i$
- Если $P_{all} > T$, то нет клеток, в которых обязательно находится корабль
- Если $P_{all} = T$, то в текущий момент R_{all} клеток, в которых обязательно находится какой-нибудь корабль

Решение

- Требуется поддерживать множество текущих отрезков
- Одна из возможных реализаций — сбалансированное двоичное дерево (`std::set` в C++)

Частичные решения

- Переборное решение — 30 баллов.
- Решение, поддерживающее множество отрезков за $O(N)$ операций на выстрел — 60 баллов.

Альтернативные решение

- Перекрашивать меньший отрезок — 100 баллов
- Решение за $N\sqrt{N}$ — 100 баллов.

Вопросы?

Задача «Массовый прогноз»

Задача «Массовый прогноз»

Задача «Массовый прогноз»

Задача «Массовый прогноз»

- Идея задачи — Глеб Евстропов
- Подготовка тестов — Павел Кунявский, Глеб Евстропов
- Разбор задачи — Глеб Евстропов

Математическая формулировка

Входные данные:

- Количество элементов последовательности n
- Диапазон цветов k
- Последовательность цветов c_i

Результат:

- Количество подотрезков исходной последовательности, на которых какой то цвет встречается больше раз чем все остальные вместе взятые

$O(n^3 \cdot k)$

- Решим задачу отдельно для каждого подотрезка
- Отдельно проверим каждый цвет на преобладание на выбранном отрезке
- Пусть цвет встречается cnt раз на отрезке (i, j), тогда цвет преобладает на этом отрезке если $cnt \cdot 2 > (j - i + 1)$
- Итоговая сложность $O(n^3 \cdot k) \leqslant$ кол-во отрезков $\times$ кол-во цветов $\times$ максимальную длину отрезка

$O(n^3 \cdot k)$

- Решим задачу отдельно для каждого подотрезка
- Отдельно проверим каждый цвет на преобладание на выбранном отрезке
- Пусть цвет встречается cnt раз на отрезке (i, j), тогда цвет преобладает на этом отрезке если $cnt \cdot 2 > (j - i + 1)$
- Итоговая сложность $O(n^3 \cdot k) \leqslant$ кол-во отрезков $\times$ кол-во цветов $\times$ максимальную длину отрезка

$O(n^3 \cdot k)$

- Решим задачу отдельно для каждого подотрезка
- Отдельно проверим каждый цвет на преобладание на выбранном отрезке
- Пусть цвет встречается cnt раз на отрезке (i, j), тогда цвет преобладает на этом отрезке если $cnt \cdot 2 > (j - i + 1)$
- Итоговая сложность $O(n^3 \cdot k) \leqslant$ кол-во отрезков $\times$ кол-во цветов $\times$ максимальную длину отрезка

$O(n^3 \cdot k)$

- Решим задачу отдельно для каждого подотрезка
- Отдельно проверим каждый цвет на преобладание на выбранном отрезке
- Пусть цвет встречается cnt раз на отрезке (i, j), тогда цвет преобладает на этом отрезке если $cnt \cdot 2 > (j - i + 1)$
- Итоговая сложность $O(n^3 \cdot k) \leqslant$ кол-во отрезков $\times$ кол-во цветов $\times$ максимальную длину отрезка

$O(n^3 \cdot \log n)$ и $O(n^3)$

- Как и до этого решаем задачу отдельно для каждого отрезка
- Выделить самый часто встречающийся элемент можно с помощью сортировки элементов на подотрезке - $O(n^3 \log n)$, а затем проверять его на преобладание не этом отрезке
- В отдельном массиве посчитаем частоты встречаемости всех элементов на подотрезке и за линию выберем максимум, получив сложность $O(n^3)$

$O(n^3 \cdot \log n)$ и $O(n^3)$

- Как и до этого решаем задачу отдельно для каждого отрезка
- Выделить самый часто встречающийся элемент можно с помощью сортировки элементов на подотрезке - $O(n^3 \log n)$, а затем проверять его на преобладание не этом отрезке
- В отдельном массиве посчитаем частоты встречаемости всех элементов на подотрезке и за линию выберем максимум, получив сложность $O(n^3)$

$O(n^3 \cdot \log n)$ и $O(n^3)$

- Как и до этого решаем задачу отдельно для каждого отрезка
- Выделить самый часто встречающийся элемент можно с помощью сортировки элементов на подотрезке - $O(n^3 \log n)$, а затем проверять его на преобладание не этом отрезке
- В отдельном массиве посчитаем частоты встречаемости всех элементов на подотрезке и за линию выберем максимум, получив сложность $O(n^3)$

$O(n^2k)$ и $O(n^2)$

- Теперь будем фиксировать только левую границу, то есть при $i = const$ вычислять ответ для всех $j \geqslant i$
- Поддерживаем массив с количествами цветов на текущем отрезке. Очевидно, что при сдвиге на единицу вправу нужно изменить только одно значение этого массива
- Если для выбора максимума по-прежнему линейно пробегать весь массив, то получаем $O(n^2 \cdot k)$. Однако при сдвиге вправо максимум можно пересчитывать за $O(1)$ проверяя цвет нового элемента

$O(n^2 k)$ и $O(n^2)$

- Теперь будем фиксировать только левую границу, то есть при $i = const$ вычислять ответ для всех $j \geqslant i$
- Поддерживаем массив с количествами цветов на текущем отрезке. Очевидно, что при сдвиге на единицу вправу нужно изменить только одно значение этого массива
- Если для выбора максимума по-прежнему линейно пробегать весь массив, то получаем $O(n^2 \cdot k)$. Однако при сдвиге вправо максимум можно пересчитывать за $O(1)$ проверяя цвет нового элемента

$O(n^2k)$ и $O(n^2)$

- Теперь будем фиксировать только левую границу, то есть при $i = const$ вычислять ответ для всех $j \geqslant i$
- Поддерживаем массив с количествами цветов на текущем отрезке. Очевидно, что при сдвиге на единицу вправу нужно изменить только одно значение этого массива
- Если для выбора максимума по-прежнему линейно пробегать весь массив, то получаем $O(n^2 \cdot k)$. Однако при сдвиге вправо максимум можно пересчитывать за $O(1)$ проверяя цвет нового элемента

$O(n^2k)$ и $O(n^2)$

- Теперь будем фиксировать только левую границу, то есть при $i = const$ вычислять ответ для всех $j \geqslant i$
- Поддерживаем массив с количествами цветов на текущем отрезке. Очевидно, что при сдвиге на единицу вправу нужно изменить только одно значение этого массива
- Если для выбора максимума по-прежнему линейно пробегать весь массив, то получаем $O(n^2 \cdot k)$. Однако при сдвиге вправо максимум можно пересчитывать за $O(1)$ проверяя цвет нового элемента

$O(nk \log n)$

- Легко заметить, что так как условие преобладания строгое, то на отрезке не может быть двух и более преобладающих цветов.
- Из этого соображения следует, что по каждому цвету задачу можно решать независимо.
- Если зафиксировать цвет c и заменить на $+1$ все элементы данного цвета и на -1 все остальные, то ответом для данного цвета является кол-во отрезков с положительной суммой.
- Если перейти к последовательности

$O(nk \log n)$

- Легко заметить, что так как условие преобладания строгое, то на отрезке не может быть двух и более преобладающих цветов.
- Из этого соображения следует, что по каждому цвету задачу можно решать независимо.
- Если зафиксировать цвет с и заменить на $+1$ все элементы данного цвета и на -1 все остальные, то ответом для данного цвета является кол-во отрезков с положительной суммой.
- Если перейти к последовательности

$O(nk \log n)$

- Легко заметить, что так как условие преобладания строгое, то на отрезке не может быть двух и более преобладающих цветов.
- Из этого соображения следует, что по каждому цвету задачу можно решать независимо.
- Если зафиксировать цвет c и заменить на $+1$ все элементы данного цвета и на -1 все остальные, то ответом для данного цвета является кол-во отрезков с положительной суммой.
- Если перейти к последовательности

$O(nk \log n)$

- Легко заметить, что так как условие преобладания строгое, то на отрезке не может быть двух и более преобладающих цветов.
- Из этого соображения следует, что по каждому цвету задачу можно решать независимо.
- Если зафиксировать цвет c и заменить на $+1$ все элементы данного цвета и на -1 все остальные, то ответом для данного цвета является кол-во отрезков с положительной суммой.
- Если перейти к последовательности

$O(nk \log n)$

- Легко заметить, что так как условие преобладания строгое, то на отрезке не может быть двух и более преобладающих цветов.
- Из этого соображения следует, что по каждому цвету задачу можно решать независимо.
- Если зафиксировать цвет c и заменить на $+1$ все элементы данного цвета и на -1 все остальные, то ответом для данного цвета является кол-во отрезков с положительной суммой.
- Если перейти к последовательности

$O(n \cdot k)$

- Научимся решать задачу для одного цвета за $O(n)$. Как было сказано, требуется в последовательности частичных сумм для каждого элемента найти количество меньших его и расположенных левее.
- Заметим, что так как два соседних элемента отличаются ровно на 1, то между двумя соседними элементами одного значения, все промежуточные либо строго больше, либо строго меньше.
- Отсюда следует, что ответ для фиксированной позиции легко

$O(n \cdot k)$

- Научимся решать задачу для одного цвета за $O(n)$. Как было сказано, требуется в последовательности частичных сумм для каждого элемента найти количество меньших его и расположенных левее.
- Заметим, что так как два соседних элемента отличаются ровно на 1, то между двумя соседними элементами одного значения, все промежуточные либо строго больше, либо строго меньше.
- Отсюда следует, что ответ для фиксированной позиции легко

$O(n \cdot k)$

- Научимся решать задачу для одного цвета за $O(n)$. Как было сказано, требуется в последовательности частичных сумм для каждого элемента найти количество меньших его и расположенных левее.
- Заметим, что так как два соседних элемента отличаются ровно на 1, то между двумя соседними элементами одного значения, все промежуточные либо строго больше, либо строго меньше.
- Отсюда следует, что ответ для фиксированной позиции легко

$O(n^{1.5} \cdot \log n)$ и $O(n^{1.5})$

- Заметим, что если кол-во элементов цвета c равно cnt, то он может преобладать на отрезках длины не более $2cnt - 1$.
- Разделим цвета на две группы: встречаемость которых больше либо равна $n^{0.5}$ и наоборот.
- Для первой группы мы умеем решать задачу за $O(n)$ для одного цвета, но количество цветов в этой группе не превосходит $n^{0.5}$.
- Для второй группы достаточно рассмотреть все отрезки, длина которых не превосходит $2n^{0.5}$.
- Обе части, а, следовательно, и всё решение

$O(n^{1.5} \cdot \log n)$ и $O(n^{1.5})$

- Заметим, что если кол-во элементов цвета c равно cnt, то он может преобладать на отрезках длины не более $2cnt - 1$.
- Разделим цвета на две группы: встречаемость которых больше либо равна $n^{0.5}$ и наоборот.
- Для первой группы мы умеем решать задачу за $O(n)$ для одного цвета, но количество цветов в этой группе не превосходит $n^{0.5}$.
- Для второй группы достаточно рассмотреть все отрезки, длина которых не превосходит $2n^{0.5}$.
- Обе части, а следовательно и все решение

$O(n^{1.5} \cdot \log n)$ и $O(n^{1.5})$

- Заметим, что если кол-во элементов цвета c равно cnt, то он может преобладать на отрезках длины не более $2cnt - 1$.
- Разделим цвета на две группы: встречаемость которых больше либо равна $n^{0.5}$ и наоборот.
- Для первой группы мы умеем решать задачу за $O(n)$ для одного цвета, но количество цветов в этой группе не превосходит $n^{0.5}$.
- Для второй группы достаточно рассмотреть все отрезки, длина которых не превосходит $2n^{0.5}$.
- Обе части, а, следовательно, и все решение

$O(n^{1.5} \cdot \log n)$ и $O(n^{1.5})$

- Заметим, что если кол-во элементов цвета c равно cnt, то он может преобладать на отрезках длины не более $2cnt - 1$.
- Разделим цвета на две группы: встречаемость которых больше либо равна $n^{0.5}$ и наоборот.
- Для первой группы мы умеем решать задачу за $O(n)$ для одного цвета, но количество цветов в этой группе не превосходит $n^{0.5}$.
- Для второй группы достаточно рассмотреть все отрезки, длина которых не превосходит $2n^{0.5}$.
- Обе части, а, следовательно, и все решение

$O(n^{1.5} \cdot \log n)$ и $O(n^{1.5})$

- Заметим, что если кол-во элементов цвета c равно cnt, то он может преобладать на отрезках длины не более $2cnt - 1$.
- Разделим цвета на две группы: встречаемость которых больше либо равна $n^{0.5}$ и наоборот.
- Для первой группы мы умеем решать задачу за $O(n)$ для одного цвета, но количество цветов в этой группе не превосходит $n^{0.5}$.
- Для второй группы достаточно рассмотреть все отрезки, длина которых не превосходит $2n^{0.5}$.
- Обе части, а следовательно, и все решение

$O(n \log n)$ и $O(n)$

- Рассмотрим следующие три примера: все $+1$ образуют непрерывный отрезок, $+1$ расположены вперемешку с -1 на отрезке длины $2cnt$ и $+1$ образую два сильно разведенных отрезка.
- Легко заметить, что в первых двух случаях достаточно отойти не более чем на cnt вправо и на cnt влево и решить за линейное решение задачу на данном подотрезке. В третьем случае задача разбивается на 2 отрезка, оставаясь суммарно линейной по cnt

$O(n \log n)$ и $O(n)$

- Рассмотрим следующие три примера: все $+1$ образуют непрерывный отрезок, $+1$ расположены вперемешку с -1 на отрезке длины $2cnt$ и $+1$ образую два сильно разведенных отрезка.
- Легко заметить, что в первых двух случаях достаточно отойти не более чем на cnt вправо и на cnt влево и решить за линейное решение задачу на данном подотрезке. В третьем случае задача разбивается на 2 отрезка, оставаясь суммарно линейной по cnt

$O(n \log n)$ и $O(n)$

- Рассмотрим следующие три примера: все $+1$ образуют непрерывный отрезок, $+1$ расположены вперемешку с -1 на отрезке длины $2cnt$ и $+1$ образую два сильно разведенных отрезка.
- Легко заметить, что в первых двух случаях достаточно отойти не более чем на cnt вправо и на cnt влево и решить за линейное решение задачу на данном подотрезке. В третьем случае задача разбивается на 2 отрезка, оставаясь суммарно линейной по cnt

$O(n \log n)$ и $O(n)$

- Утверждение: искомое разбиение находится как объединение отрезков следующего вида: для каждого i рассмотрим минимальное j такое что сумма на отрезке (j, i) положительна, и максимальное j с таким же условием.
- Жюри располагает строгим доказательством этого факта и готово изложить его по просьбе участников, здесь же оно не приводится для экономии места и времени, а так же является простым и полезным упражнением

$O(n \log n)$ и $O(n)$

- Утверждение: искомое разбиение находится как объединение отрезков следующего вида: для каждого i рассмотрим минимальное j такое что сумма на отрезке (j, i) положительна, и максимальное j с таким же условием.
- Жюри располагает строгим доказательством этого факта и готово изложить его по просьбе участников, здесь же оно не приводится для экономии места и времени, а так же является простым и полезным упражнением

$O(n \log n)$ и $O(n)$

- Утверждение: искомое разбиение находится как объединение отрезков следующего вида: для каждого i рассмотрим минимальное j такое что сумма на отрезке (j, i) положительна, и максимальное j с таким же условием.
- Жюри располагает строгим доказательством этого факта и готово изложить его по просьбе участников, здесь же оно не приводится для экономии места и времени, а так же является простым и полезным упражнением

Вопросы?

Задача «Флешмоб»

Задача «Флешмоб»

Задача «Флешмоб»

Задача «Флешмоб»

- Идея задачи — Юрий Петров
- Подготовка тестов — Михаил Пядёркин, Сергей Мельников
- Разбор задачи — Сергей Мельников

Задача

Входные данные:

- Дано N отрезков, каждый горизонтальный или вертикальный.
- Отрезки упорядочены.

Задача:

- Найти на каждом отрезке точку, не покрытую предыдущими отрезками.

Идеи решения

- Ответ либо 0, либо N.
- Независимо решаем для горизонтальных и для вертикальных отрезков.
- Будем говорить, что отрезок номер t добавляется в момент времени t.

Сжатие координат

- Рассмотрим множество X в котором находятся координаты концы отрезков.
- Существует ответ, что для любой точки в нём либо x, либо $x - 1$ лежит в X.
- Аналогично для Y.

Сканирующая прямая

- Идём сканирующей прямой сверху вниз.
- Для каждой координаты X храним в очереди с приоритетами, какие вертикальные отрезки уже начались, но ещё не закончились.
- Минимум в этой очереди это время, в которое данная координата X на сканирующей прямой покрывается отрезком.
- Будем хранить эти минимумы в дереве отрезков T1 для горизонтали с операцией максимум.

Сканирующая прямая (2)

События:

- Начался вертикальный отрезок.
  - Добавим номер отрезка в очередь с приоритетами.
  - Обновим значение в дереве отрезков.
- Закончился вертикальный отрезок.
  - Удалим его номер из очереди с приоритетами.
  - Обновим значение в дереве отрезков.

Решение для одной горизонтали

- В дереве отрезков T1 хранится информация, в какой минимальный момент времени это точка уже покрыта. Для некоторых клеток это значение бесконечность.
- Заведём ещё одно дерево отрезков T2 с операцией максимум. В нём будем хранить информацию про горизонтальные отрезки.

Решение для одной горизонтали (2)

- Рассматриваем отрезки в порядке времени их появления.
- Делаем запрос на максимум в объединении деревьев T1 и T2, если есть точка с временем бесконечность, то ставим точку туда.
- Присваиваем в T2 на отрезке время появления этого отрезка.

Время работы

- $O(N)$ событий, суммарное время на обработку $O(N \log N)$.
- Горизонтальных отрезков не больше, чем N. Суммарное время на обработку $O(N \log N)$.
- Время работы $O(N \log N)$.

Вопросы?

EOF