Всероссийская олимпиада школьников по информатике

Архангельск

5–11 апреля 2015 года

Задача «Автоматические друзья»

Задача «Автоматические друзья»

Задача «Автоматические друзья»

Задача «Автоматические друзья»

- Идея задачи — Михаил Пядеркин
- Подготовка тестов — Павел Кунявский
- Разбор задачи — Павел Кунявский

Постановка задачи

- Дано n троек чисел (a_i, b_i, c_i)
- Найти количество пар индексов $i < j$ таких, что они равны по ровно одному индексу

Решение первой подзадачи

- Переберем все пары индексов
- Проверим выполнение условия
- Сложность решения — $O(n^2)$

Решение второй подзадачи

- Посчитаем количество пар индексов, для которых $a_i = a_j$.
- Для этого посчитаем количество троек с каждым из значений (за $O(n)$ в сумме)
- После этого количество пар с таким значением можно вычислить по формуле $\frac{k \cdot (k-1)}{2}$, где k количество троек с таким значением

Решение второй подзадачи

- Аналогично посчитаем пары, когда $b_i = b_j, c_i = c_j$.
- Сложим эти значения. При этом мы учли все нужные пары, но дополнительно учли пары совпадающие по двум элеметам два раза.
- Пары, совпадающие по двум элементам можно посчитать аналогично. Вычтем их два раза.

Решение второй подзадачи

- Но при этом мы вычли тройки три 6 раз, в то время, как мы их прибавили три раза на первом шаге.
- Посчитаем их аналогично, и прибавим три раза.

Вопросы?

Задача «Памятник»

Задача «Памятник»

Задача «Памятник»

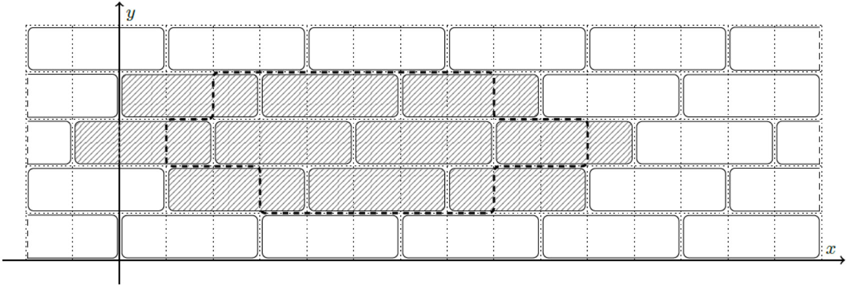
Задача «Памятник»

- Идея задачи — Георгий Корнеев
- Подготовка тестов — Никита Иоффе и Павел Маврин
- Разбор задачи — Никита Иоффе

Постановка задачи

- Требовалось разместить клетчато-выпуклый многоугольник на поле так, чтобы он занимал как можно меньше плиток размерами $1 \times k$;
- Плитки, которые покрываются не полностью, следует учитывать.

Постановка задачи



Решение на 32 балла

- Переберём сдвиг многоугольника по оси x и по оси y
- Для каждой y-координаты посчитаем сколько плиток с такой координатой пересекают наш многоугольник
- Среди всех сдвигов выбрать тот, который даёт минимальный ответ

Решение на 69 баллов

- Заметим, что сдвиг на один по y-координате эквивалентен сдвигу на один по x-координате
- Будем перебирать сдвиги только по одной из координат

Полное решение

- Для каждой y-координаты посчитаем величину $left_y$ — x-координату самой левой точки многоугольника с такой y-координатой
- Аналогично посчитаем $right_y$
- Для каждой y-координаты посчитаем количество плиток, которые точно будут накрыты, с такой y-координатой
- Осталось посчитать те плитки, которые будут накрыты не полностью

Полное решение

- В каждой y-координате не более двух плиток пересекаются с многоугольником не полностью.
- Когда x-координата левой границы прямоугольника станет кратна k, то пересечение слева исчезнет
- Аналогично, когда x-координата правой границы прямоугольника сравнима с единицей по модулю k, то появляется новое пересечение.

Полное решение

- Для каждого из k, суммарно за k, посчитаем сколько плиток пересекаются с прямоугольником

Вопросы?

Задача «Вышивка жемчугом»

Задача «Вышивка жемчугом»

Задача «Вышивка жемчугом»

Задача «Вышивка жемчугом»

- Идея задачи — Максим Ахмедов
- Подготовка тестов — Максим Ахмедов, Глеб Евстропов
- Разбор задачи — Максим Ахмедов

Постановка задачи

- Дано изображение дерева из n вершин на прямоугольной сетке
- Каждое ребро — либо вертикальный, либо горизонтальный отрезок длины 1
- Дано q запросов, каждый имеет вид «сколько компонент связности образуется при вырезании данного прямоугольного фрагмента»

Решение первой подзадачи

- Переберём все вершины сетки внутри прямоугольника запроса
- Запустим из каждой обход дерева, не выходя за пределы прямоугольника
- Посчитаем компоненты связности
- Итоговая сложность — $O(qwh)$.

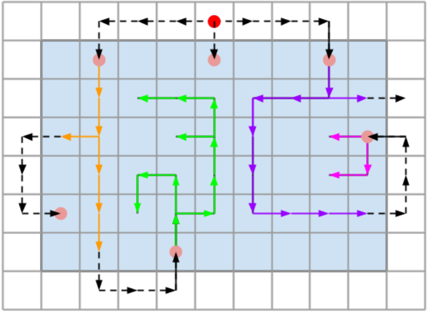
Решение второй подзадачи

- Будем запускать обход только из тех вершин, которые попадают в прямоугольник запроса
- Так как вершин всего n, то это решение будет работать за $O(qn)$

Первый способ решения

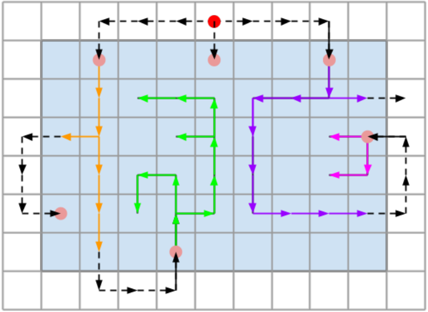
- Нам нужен способ однозначно идентифицировать каждую компоненту
- В каждой компоненте связности надо выделить её представителя
- Какую вершину в компоненте связности можно назвать «особенной»?

Первый способ решения



- Подвесим дерево за произвольную вершину и ориентируем рёбра от неё

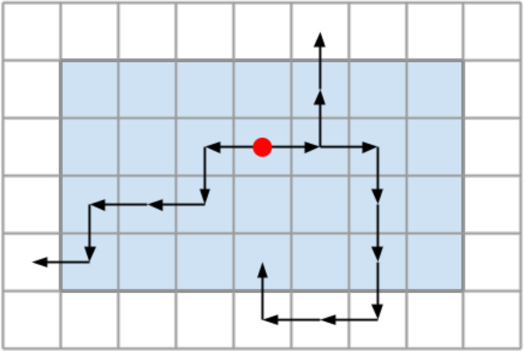
Первый способ решения



- Пусть «особенная» вершина — самая верхняя по дереву в каждой компоненте. В неё либо входит ребро снаружи, либо она — корень дерева

Первый способ решения

Пример, когда корень попадает внутрь прямоугольника:



Решение третьей подзадачи

- Для каждой строки выпишем массив с единицами на позициях с рёбрами, направленными вверх
- Количество нужных нам рёбер на нижней границе прямоугольника — сумма на подотрезке этого массива, которую можно найти за $O(1)$ с помощью частичных сумм
- Аналогично с рёбрами, направленными влево, вверх и вправо
- Сложность такого решения — $O(wh + q)$

Решение четвёртой подзадачи

- Если w и h большие, воспользоваться частичными суммами нельзя
- Для каждой строки выпишем массив с позициями рёбер, направленных вверх, и отсортируем
- Количество нужных нам рёбер на нижней границе прямоугольника можно найти с помощью двух бинарный поисков
- Сложность такого решения — $O((q+n)\log n)$

Второй способ решения

- В каждой компоненте связности рёбер на единицу меньше, чем вершин
- Значит, количество компонент связности это количество рёбер внутри прямоугольника минус количество вершин внутри прямоугольника

Второй способ решения

- Задаём горизонтальное ребро координатами левого конца, вертикальное координатами нижнего конца
- x_1, y_1, x_2, y_2 — координаты прямоугольника запроса
- $A =$ *количество вершин* (x, y), *таких что* $x_1 \le x \le x_2$ *и* $y_1 \le y \le y_2$
- $B =$ *количество горизонтальных рёбер* (x, y), *таких что* $x_1 \le x \le x_2 - 1$ *и* $y_1 \le y \le y_2$
- $C =$ *количество вертикальных рёбер* (x, y), *таких что* $x_1 \le x \le x_2$ *и* $y_1 \le y \le y_2 - 1$
- Ответ на запрос это $A - B - C$

Решение третьей подзадачи

- Подсчёт величин A, B и C — двумерная задача количества точек в прямоугольнике
- Научимся считать A. Поставим в двумерном массиве $F[\cdot][\cdot]$ единицы на позициях, соответствующих вершинам
- $A = \mathrm{sum}(F[x_1 \ldots x_2][y_1 \ldots y_2])$
- Такие суммы тоже можно вычислять за $O(1)$, предпосчитав суммы на префиксах $S[x][y] = \mathrm{sum}(F[1 \ldots x][1 \ldots y])$
- Тогда $A = S[x_2][y_2] - S[x_1 - 1][y_2] - S[x_2][y_1 - 1] + S[x_1 - 1][y_1 - 1]$

Решение третьей подзадачи

- Посчитаем таким образом A, B и C за время $O(wh)$
- Получим решение за $O(wh + q)$

Дальнейшее улучшение

- Подсчёт количества точек в прямоугольнике можно производить с помощью двумерной структуры данных, например, двумерного дерева отрезков
- Базовая реализация двумерного дерева отрезков работает за $O(\log^2 n)$ на запрос
- Этого пока слишком медленно для прохождения четвёртой подзадачи

Решение четвёртой подзадачи

- Можно улучшить двумерную структуру данных до ответа на запрос за время $O(\log n)$
- Можно воспользоваться методом сканирующей прямой для сведения задачи к одномерной в off-line
- Получается решение за $O((n+q)\log n)$

Вопросы?

Задача «Пингвиноведение»

Задача «Пингвиноведение»

Задача «Пингвиноведение»

Задача «Пингвиноведение»

- Идея задачи — Максим Ахмедов
- Подготовка тестов — Нияз Нигматуллин, Михаил Пядеркин
- Разбор задачи — Михаил Пядеркин

Постановка задачи

- Дана строка из нулей и единиц и число k.
- Требуется найти строку, которую можно представить в виде не более, чем k блоков одинаковых цифр.
- Среди таких необходимо выбрать ту, которая отличается от исходной в минимальном количестве позиций.

Простейшее решение на 11 баллов

- Отметим, что если длина исходной строки невелика, то мы можем за $O(2^n)$ перебрать все возможные строки, проверить, разбиваются ли они на нужное число блоков.
- Среди них необходимо выбрать ту, которая отличается от исходной в минимальном количестве позиций.
- Количество символов, в котором строка отличается от исходной, будем называеть стоимостью строки.

Динамическое программирование

- Пусть $ans[i][j]$ обозначает оптимальный ответ для префикса исходной строки длины i, если бы мы разбивали его на j блоков.
- Инициализация: пустой префикс можно разбить на любое число блоков, стоимость любого разбиения 0.

Динамическое программирование

- Переход: переберем $x \leqslant i$ — место начала последнего блока, тогда $ans[i][j] = \min_{x=1}^{i}(ans[x-1][j-1] + cost[x][i])$, где $cost[x][i]$ — стоимость оптимального разбиения подстроки $[x, i]$ на один блок.
- $cost[x][i] = \min(zeroes, ones)$, где $zeroes$ обозначает число нулей, а $ones$ — число единиц на отрезке $[x, i]$.
- Ответ задачи содержится в $ans[n][k]$.
- 35 баллов.

Динамическое программирование

- Пусть $ans[i][j][c]$ обозначает оптимальный ответ, если мы разбиваем префикс длины i на j блоков, причем последний блок состоит из символов c.
- Переход: очередной символ может либо начать новый блок, либо продолжать старый, то есть

$$ans[i][j][c] = (s_i \neq c) +$$

$$+ \min(ans[i-1][j][c], ans[i][j-1][0], ans[i][j-1][1])$$

- 59 баллов.

Сведение задачи к стандартной

- Для начала приблизим исходную строку с помощью одного блока из нулей.
- После этого необходимо выбрать какие-то блоки, которые будут состоять из 1.
- При этом, если мы выбираем какой-либо блок и инвертируем его, то стоимость разбиения изменяется на $-ones + zeroes$.

Сведение задачи к стандартной

- Таким образом, если заменить исходную строку из 0 и 1 на последовательность из -1 и 1, то необходимо решить следующую задачу: выбрать в массиве $k/2$ непересекающихся отрезков с максимальной суммой чисел.

Решение стандартной задачи

- Пусть нам необходимо выбрать в массиве k непересекающихся отрезков с максимальной суммой.
- Для $k = 1$ необходимо найти один отрезок с максимальной суммой, это — стандартная задача.

Решение стандартной задачи

- Будем строить ответ инкрементально: пусть мы уже построили k отрезков, научимся строить $k+1$.
- Утверждается, что выгодно либо добавить отрезок, который не пересекается ни с одним из уже выбранных, либо следует разбить один из выбранных отрезков на два, «вырезав» из него некоторый подотрезок.

Решение стандартной задачи

- Необходимо на отрезке уметь искать подотрезок с максимальной суммой, это — стандартная задача на дерево отрезков.
- Так на каждом шаге есть $O(k)$ вариантов действий, то мы уже получили решение за $O(k^2 + n \log n)$.
- 80 баллов.

Решение стандартной задачи

- Для оптимизации решения следует заметить, что наши возможности слабо изменяются при перестроении ответа: у нас, возможно, добавляются три новых возможных отрезка, и удаляется один из возможных.
- Таким образом, мы можем использовать `PriorityQueue` или `Set` для выбора оптимального действия на каждом шаге.
- 100 баллов.

Задача «Поможем дикой природе»

Задача «Поможем дикой природе»

Задача «Поможем дикой природе»

Задача «Поможем дикой природе»

- Идея задачи — Николай Ведерников
- Подготовка тестов — Павел Кунявский
- Разбор задачи — Елена Андреева

Формальная постановка задачи

- Дано число n
- Найти три целых неотрицательных числа a, b, c такие, что $a + b + c = n$ и значение $a \mathbin{\&} b \mathbin{\&} c$ максимально, где $\&$ — побитовое "И"

Двоичное представление суммы грантов, выданных организации, однозначно определяет, какие именно гранты ей выданы. Выражение $a \mathbin{\&} b \mathbin{\&} c$ учитывает только целевые гранты.

Решение на 49 баллов

- Переберем все возможные числа a, b, такие что $0 \le a \le n$; $0 \le b \le n - a$.
- Тогда $c = n - a - b$.
- Из выражений a & b & c выберем максимальное.
- Сложность решения — $O(n^2)$

Решение на 66 баллов. Часть 1

Заметим, что $a \mathbin{\&} b \mathbin{\&} c \le n/3$

- Если $n \mod 3 = 0$, то $a = b = c = n/3$ (все выданные гранты — целевые)
- Если $[n/3] \mod 2 = 0$, то $a = [n/3]$, $b = [n/3] + 1$, $c = n - a - b$, $a \mathbin{\&} b \mathbin{\&} c = a$
- ...

Решение на 66 баллов. Часть 2

- ...
- Если n нечётно, то либо a, b и c нечётны, либо a нечётно, b и c чётны.
- Аналогично для чётного n.

Подберём остальные биты чисел a, b и c рекурсивно.

Решение на 100 баллов

- В предыдущем решении сохранять промежуточные результаты в ассоциативный массив (map, словарь).

Решение на 100 баллов

- Переберём все биты числа n слева направо. Для i-го бита:
  - Если $3 \cdot 2^i \leq n$, то выдадим всем организациям грант размера 2^i.
  - Если $3 \cdot (2^i - 1) > n$, то выдадим грант размера 2^i первой организации.
  - Уменшим n на сумму выданных грантов.
- Сложность решения — $O(\log n)$.

Вопросы?

Задача «Подводная лодка»

Задача «Подводная лодка»

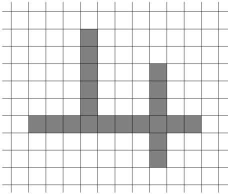
Задача «Подводная лодка»

Задача «Подводная лодка»

- Идея задачи — Михаил Пядеркин
- Подготовка тестов — Павел Маврин и Сергей Мельников
- Разбор задачи — Павел Маврин

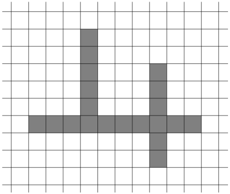
Постановка задачи

- Дан снимок местности с отметками высот точек
- Требуется выделить фигуру в форме подводной лодки, такую, что сумма высот входящих в нее клеток максимальна



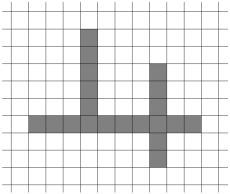
Решение за $O(n^9)$

- Решение за $O(n^9)$
- Получало 32 балла
- Идея: перебрать все возможные позиции
- Позиция задается восьмеркой чисел x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4.



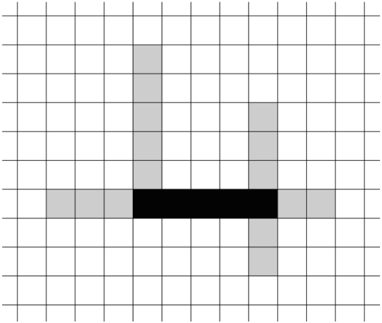
Решение за $O(n^4)$

- Решение за $O(n^4)$
- Получало 54 балла
- Идея: независимо выбирать максимумы частей



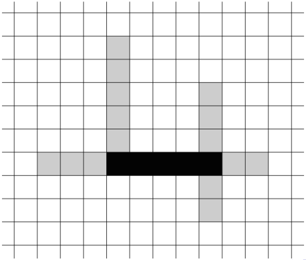
Разделение фигуры на части

- Переберем все варианты расположения черного отрезка, их $O(n^3)$



Независимый выбор максимумов

- Если черный отрезок зафиксирован, то серые отрезки можно максимизиоровать независимо.
- Для каждого из серых отрезков есть $O(n)$ вариантов.

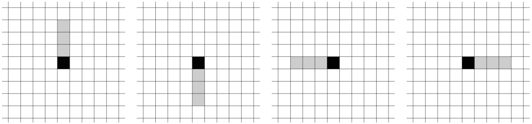


Решение за $O(n^3)$

- Решение за $O(n^3)$
- Получало 77 баллов (или больше, в зависимости от реализации)
- Идея: предподсчет отрезков с максимальной суммой

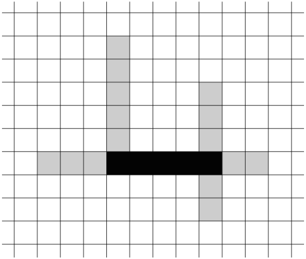
Предподсчет отрезков

- Для каждой клетки и каждого из четырех направлений найдем максимальный серый отрезок в этом направлении
- Запишем суммы на этих отрезках в массивы $up[i][j]$, $down[i][j]$, $left[i][j]$ и $right[i][j]$



Решение за $O(n^3)$

- Теперь, если зафиксирован черный отрезок, то максимальную сумму можно найти за $O(1)$

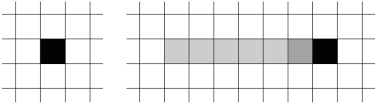


Решение за $O(n^3)$

- Решение за $O(n^2)$
- Получало 100 баллов
- Идея: динамическое программирование

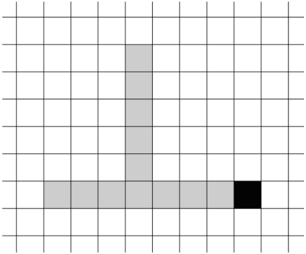
Нахождение максимальных отрезков

- Применим метод динамического программирования
- Сначала научимся считать значения $up[i][j]$, $down[i][j]$, $left[i][j]$ и $right[i][j]$ за $O(n^2)$
- Например, $left[i][j] = \max(0, a[i][j-1] + left[i][j-1])$



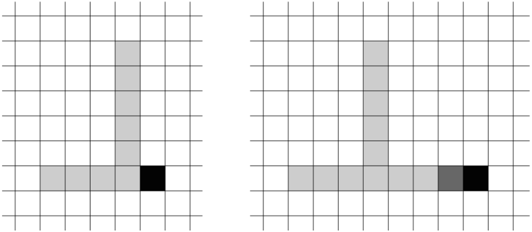
Нахождение носов с максимальной суммой

- Теперь для каждой клетки найдем «нос с палубой» с максимальной суммой. Запишем эту сумму в массив $bow[i][j]$



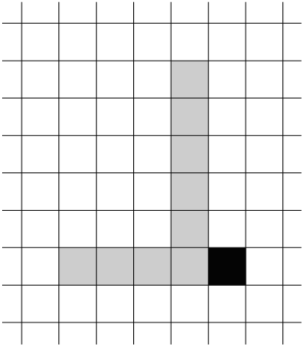
Вычисление значений $bow[i][j]$

- Применим метод динамического программирования
- Есть два варианта: развилка находится в клетке $(i, j-1)$ или левее



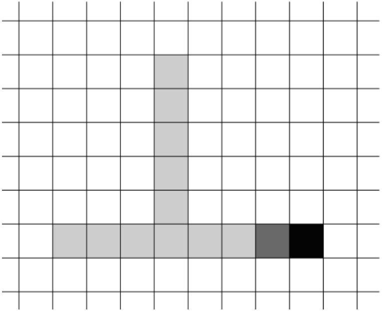
Вычисление значений $bow[i][j]$

- В первом случае максимальная сумма равна $left[i][j-1] + up[i][j-1] + a[i][j-1]$



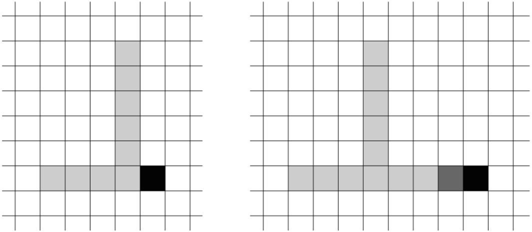
Вычисление значений $bow[i][j]$

- Во втором случае максимальная сумма равна $bow[i][j-1] + a[i][j-1]$



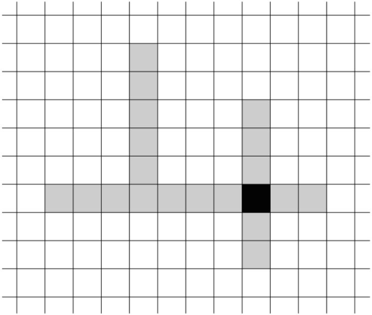
Вычисление значений $bow[i][j]$

- Выберем максимум из этих двух значений



Финальное вычисление максимума

- Переберем точку основания хвоста
- Максимальная сумма равна $bow[i][j] + up[i][j] + right[i][j] + down[i][j] + a[i][j]$



Вопросы?

Задача «Фонари»

Задача «Фонари»

Задача «Фонари»

Задача «Фонари»

- Идея задачи — Ольга Черникова, Павел Кунявский
- Подготовка тестов — Никита Иоффе, Михаил Пядеркин
- Разбор задачи — Михаил Пядеркин

Постановка задачи

- Дан массив из нулей и единиц
- Поступают запросы вида «присвоить некоторое значение на отрезке»
- После каждого запроса необходимо уметь отвечать на запрос «количество подотрезков массива, которые хоть когда-то состояли целиком из 1»

Решение на 17 баллов

- Будем хранить для каждого сегмента, является ли он исправным.
- После каждого запроса присвоения необходимо для каждого из сегментов проверить, состоит ли он после очередного события только из 1.
- Честно проверим исправность каждого сегмента.
- $O(n^2)$ сегментов, одна проверка работает за $O(n)$.
- Итоговая сложность $O(qn^3)$.

Решение на 36 баллов

- В предыдущем решении узким местом является проверка всех сегментов.
- Каждый сегмент $[l, r]$ характеризуется левым и правым концом.
- Зафиксировав левый конец, мы можем двигать правый и «на лету» поддерживать исправность отрезка.
- Итоговая сложность $O(qn^2)$.

Решение на 68+ баллов

- Зафиксируем левый конец отрезка и рассмотрим семейство сегментов $[l, l]$, $[l, l+1]$, $[l, l+2]$, $\ldots$, $[l, r]$.
- Заметим, что в этом порядке первые несколько сегментов (возможно, ноль) когда-либо были исправны, а остальные не были полностью исправны никогда.
- Таким образом, для каждого левого конца l достаточно поддерживать r_l — максимальную правую границу такую, что сегмент $[l, r_l]$ когда-либо был исправен.

Решение на 68+ баллов

- Что происходит при присваивании? Если присваивается ноль, то новых сегментов не добавляется и границы r_l не меняются.
- Если присваивается единица, то появляется большой отрезок $[x, y]$, состоящий из 1.

1 0 1 0 1 0 1 1 1 0 0

1 0 1 1 1 1 1 1 1 0 0

1 0 1 1 1 1 1 1 1 0 0

Решение на 68+ баллов

- При этом, на всем этом отрезке и только на нем, возможно, изменяются значения r_l.
- А именно, для всех $l \in [x, y]$ нужно выполнить *релаксацию* — если $r_l < y$, то необходимо присвоить $r_l = y$.

1 * 3 * 5 * 9 9 9 * *

1 0 1 1 1 1 1 1 1 0 0

1 * 9 9 9 9 9 9 9 * *

Решение на 68+ баллов

- Таким образом, после запроса присваивания 1 на сегменте $[l, r]$ нам необходимо вычислить максимальный отрезок $[x, y]$ из 1, который образовался после этого присваивания, это можно сделать с помощью цикла.
- После этого на этом отрезке нужно, возможно, обновить значения правых границ r_l.
- Количество сегментов можно вычислить, просуммировав $r_l - l$ по всем l.

Формализация решения

- После присваивания 1 необходимо уметь находить границы образовавшегося отрезка из 1, для этого необходимо уметь искать ближайший ноль слева и справа.

1 0 1 0 1 0 1 1 1 0 0

1 1 1 1

1 0 1 1 1 1 1 1 1 0 0

Формализация решения

- Затем необходимо на образовавшемся отрезке из 1 для значений r_l выполнить релаксацию на отрезке.
- Для ответов на запросы необходимо поддерживать сумму r_l по всем l.

Формализация решения

- Однако, нетрудно заметить, что r_l не убывают с ростом l: из того, что отрезок $[l, r]$ когда-либо был целиком исправным следует, что и отрезок $[l+1, r]$ когда-либо был исправен.
- Таким образом, запрос релаксации упрощается: на самом деле на некотором отрезке нужно выполнить присваивание, необходимо лишь правильно определить его границы.

Дерево отрезков

- Необходимо использовать два дерева отрезков: одно для поддержания состояния фонарей, второе для значений r_l.
- Присваивание на отрезке можно выполнять стандартным образом.

Дерево отрезков

- Поиск ближайшего ноля слева или справа можно также выполнять с помощью дерева отрезков: для этого в вершине дерева отрезков необходимо хранить самый левый ноль и самый правый ноль на отрезке.
- Для поиска отрезка релаксации можно либо воспользоваться бинарным поиском, либо использовать спуск по дереву.
- В зависимости от качества реализации и выбора компилятора такое решение получает от 80 до 100 баллов.

Корневая декомпозиция

- Можно заменить дерево отрезков на корневую декомозицию.
- Для каждого из блоков будем поддерживать, состоит он целиком из 0 или целиком из 1.
- В зависимости от качества реализации такое решение получает от 70 до 100 баллов.

Вопросы?

Задача «Сигнализация»

Задача «Сигнализация»

Задача «Сигнализация»

Задача «Сигнализация»

- Идея задачи — Максим Ахмедов
- Подготовка тестов — Максим Ахмедов, Нияз Нигматуллин, Глеб Евстропов
- Разбор задачи — Нияз Нигматуллин и Глеб Евстропов

Постановка задачи

- Дано дерево, взвешенное по вершинам d_v и рёбрам l_{vu}
- Вершина v включает u, если $\rho(v, u) \leq d_v$
- Выбирается некоторое множество вершин и включается вручную
- Происходит активация по цепочке
- Требуется найти размер минимального множества вершин, включающего весь граф

Подсчёт расстояний

- Требуется явно или неявно вычислить матрицу $\rho(i, j) \leq d_i$
- Алгоритм Флойда — $O(n^3)$
- Алгоритм Дейкстры — $O(n^2 \cdot \log n)$
- Подумать и сделать поиск в глубину — $O(n^2)$

Решение на 16 баллов

- Процесс включения будем моделировать
- Итерирование по шагам — $O(n^3)$
- Обход в ширину — $O(n^2)$
- Порядок не имеет значения — достаточно перебрать маски
- Итого: $O(2^n \cdot n^3)$ или $O(2^n \cdot n^2)$

Решение на 39 и 56 баллов

- Ориентированный граф достижимостей по неравенству $\rho(i,j) \leq d_i$
- Необходимо выбрать минимальное по размеру множество вершин, из которого достижимы все остальные
- Компоненты сильной связности за $O(v+e)$, то есть $O(n^2)$
- Граф компонент - ациклический
- Необходимо и достаточно выбрать одного представителя в каждом истоке
- Итого: 39 баллов за $O(n^3)$ и 56 за $O(n^2)$

Разделяй и властвуй

- Центроид — вершина разбивающая дерево на компоненты размером не более $\frac{n}{2}$
- Рекурсивный алгоритм: выбор и удаление центроида, запуск от поддеревьев
- Обратите внимание, что каждая вершина будет центроидом ровно один раз
- Поддерево, разбиваемое центроидом v на меньшие части, назовём компонентой данного центроида C_v
- Внутри компоненты отсортируем вершины по расстоянию до центроида за $O(C_v \cdot \log(C_v))$

Свойства центроидной декомпозиции

- Свойство 1. Для любых v и u верно ровно одно: $C_v \subset C_u$, $C_u \subset C_v$ или $C_v \cap C_u = \varnothing$
- Свойство 2. Для любой вершины v, существует не более $\log(n)$ вершин u, таких что $v \in C_u$
- Свойство 3. На пути между любой пары вершин v и u найдётся единственная вершина x, такая что $v \in C_x$ и $u \in C_x$
- Свойство 4. Асимптотика построения: $O(n \log n)$ на декомпозицию и $O(n \log^2 n)$ на сортировку.

Представление окрестности

- $R(v, r)$ — множество вершин удалённых от v не более чем на r
- Если v — центроид, то $R(v, r)$ — префикс отсортированного массива
- Любую окрестность можно представить как **объединение** не более чем $\log n$ префиксов компонент $v \in C_u$

Сжатый граф и 100 баллов

- Фиктивные вершины — префиксы отсортированного массива каждого из центроидов
- Внутри одного центроида стоится цепочка
- Каждая окрестность даёт одно ребро для каждого из центроидов её составляющих
- Суммарное число рёбер — $O(n \cdot \log(n))$
- Итого: 80 или 100 баллов в зависимости от реализации

Вопросы?