Областная олимпиада по информатике 2021. Разбор задач

Керножицкий Александр, Денгалев Даниил

Задача 1-1: Космодром

- Даны квадраты на плоскости
- Необходимо проверить, что эти квадраты образуют рамку некоторого большого квадрата

Задача 1-1: Космодром

- Найдем квадраты с минимальными и максимальнымым координатами по x и y
- Пусть это будут x_{min}, y_{min}, x_{max}, y_{max}
- Тогда квадрат с координатам (x_{min}, y_{min}) будет нижним левым углом, а (x_{max}, y_{max}) — верхним правым
- Можно вычислить p по формуле $p = x_{max}-x_{min}+1$
- Нетрудно видеть, что $x_0 = x_{min}$, $y_0 = y_{min}$

Задача 1-1: Космодром

- **Далее следует лишь проверить, что все клетки являются граничными клетками этого квадрата**
- **Стоит понимать, что граничными клетками будут те клетки, где $x = x_{min}$, $x = x_{max}$, $y = y_{min}$ или $y = y_{max}$**
- **Всего таких клеток будет 4p - 4, поскольку длина каждой стороны p, и угловые клетки будут посчитаны два раза. Исключение — случай p = 1.**
- **Нужно проверить, что количество клеток в данных будет 4p - 4 и все клетки являются граничными**
- **Если какое-то условие не выполнилось, то выводим -1**
- **Получили решение за O(n) при правильной реализации**

Задача 1-2: Игра с числами

- Два игрока играют в игру
- Игроки не обязательно ходят по порядку; они делают k ходов, для каждого из которых известно, кто ходит
- За ход один из игроков либо изменяет знак числа, либо ничего не делает
- Первый игрок стремится максимизировать сумму, второй — минимизировать
- Надо определить итоговую сумму всех чисел при оптимальной игре

Задача 1-2: Игра с числами

- Пусть сейчас ходит второй игрок
- Рассмотрим ситуацию, когда он меняет положительное число a
- Тогда он делает сумму меньше на 2a
- Если он меняет отрицательное число -a, то он делает сумму больше на 2a
- Если он ничего не делает, сумма остается неизменной

Задача 1-2: Игра с числами

- Получается, второму выгодно изменять только положительные числа, ведь он хочет минимизровать сумму
- Среди положительных чисел он поменяет знак у наибольшего
- Второй игрок не будет обратно менять знак числа на +
- Значит, поменять знак может только первый игрок. Но если он поменяет знак большего числа или второй игрок поменяет знак меньшего числа, сумма в итоге не изменится. Она сначала уменьшится на 2a ходом второго, потом увеличится на 2a ходом первого
- Однако если первый не сможет поменять знак обратно, то сумма максимально уменьшится

Задача 1-2: Игра с числами

- Аналогичными рассуждениями получаем, что для первого игрока лучшей стратегией будет менять знак минимального отрицательного числа (или пропустить ход, если отрицательных чисел нет)
- Для нахождения максимального/минимального числа можно использовать различные структуры данных (priority_queue, set, …)
- Получаем решение за O(n•log n)

Задача 1-3: Вражеские шпионы

- Дана прямоугольная матрица
- Необходимо посчитать количество ее квадратных подматриц таких, что все клетки на главной диагонали черные, а остальные — белые.

Наивное решение (18 баллов)

- Действуем просто: перебираем все подматрицы и проверяем, соответствуют ли они ограничениям
- Решение работает за $O(n^5)$ и набирает 18 баллов

Улучшенное решение (38 баллов)

- Существуют разные подходы, рассмотрим тот, который приведет нас в дальнейшем к полному решению
- Зафиксируем левый верхний угол квадрата
- Заметим, что для каждого фиксированного левого верхнего угла существует не более одной хорошей подматрицы
- Как ее найти?

Улучшенное решение (38 баллов)

- Пусть левый верхний угол — (x_0, y_0). Найдем ближайшую справа черную клетку (x_0, y)
- Тогда либо существует одна хорошая подматрица размера $y-y_0+1$, либо ее не существует вовсе
- Если мы будем проверять подматрицу за $O(n^2)$, то получим решение за $O(n^4)$, что долго

Улучшенное решение (38 баллов)

- Чтобы ускорить, посчитаем префикс-суммы для каждой строки. Проверять подматрицу будем так: проверим, что в каждой строке ровно два черных квадрата, и они стоят на нужных местах
- Тогда одна подматрица проверяется за O(n), и получаем решение за $O(n^3)$

Полное решение

- **Как ускорить решение выше?**
- **Во-первых, можно находить ближайшую справа черную клетку за O(1) с предподсчетом за O(n^2)**
- **Во-вторых, при размере s ≥ 4 можно упростить проверку:**
  - достаточно проверить только рамки квадрата (x_0,y_0) — (x_0+s-1,y_0+s-1) и то, что квадрат (x_0+1,y_0+1) — (x_0+s-2,y_0+s-2) хороший
  - первое проверяется за O(1) при помощи префикс-сумм по строкам и столбцам
  - второе проверяется за O(1), если мы уже насчитали ответ для левого верхнего угла (x_0+1,y_0+1)
- **Получаем решение за O(n^2)**

Задача 1-4: Мощный процессор

- **Дан массив из чисел от 0 до 7 и запросы следующего вида:**
  - xor: сделать на отрезке ai = ai xor x
  - seq: найти длину наибольшей возрастающей подпоследовательности на отрезке
- **Необходимо обработать все эти запросы**

Наивное решение (22 балла)

- **Запросы типа xor обрабатываются тривиально**
- **Для ответа на запросы типа seq используем ДП**
- **Например, такое: dp[x][num] — размер наибольшей возрастающей подпоследовательности, которая полностью содержится в подотрезке [l; x] и заканчивается на число num**
- **Переходы такие:**
  - если a[x] не лежит в подпоследовательности, то dp[x][num] = dp[x-1][num]
  - Иначе dp[x][a[x]] = min(dp[x-1][j]) по всем j < a[x]

Наивное решение (22 балла)

- ДП работает за O(r-l), т. е. за O(n) в худшем случае
- Получаем решение за O(nq), которое и берет 22 балла

a[i] ≤ 1 (еще 13 баллов)

- Если a[i] ≤ 1 в любой момент времени, то на запрос seq надо проверить, что самый левый ноль стоит левее самой правой единицы
- Чтобы это проверить, храним в дереве отрезков самый левый ноль, самую левую единицу, самый правый ноль и самую правую единицу
- При запросе xor у нас либо x = 0 (тогда ничего не делаем), либо x = 1 (тогда меняем местами 0 и 1 на отрезке)
- Решение работает за O((n+q)log n)

Полное решение

- Подойдем к ДП чуть по-другому
- Будем считать dp[len][x] — минимально возможный правый индекс для возрастающей подпоследовательности длины len, в которой последний элемент не превосходит x
- Как пересчитывать?

Полное решение

- **Пусть after(i, x) — первый индекс j такой, что j ≥ i и a[j] = x.**
- **Изначально:**
  - dp[1][0] = after(l, 0)
  - dp[1][x] = min(dp[1][x-1], after(l, x)
- **Пересчет:**
  - dp[i][x] = min(dp[i][x-1], after(dp[i-1][x-1], x))
- **Здесь 64 состояния, 128 переходов**
- **Остается быстро уметь считать after(i, x)**

Полное решение

- Остается быстро уметь считать after(i, x)
- Воспользуемся деревом отрезков, в котором будем хранить количество чисел 0, 1, … на отрезке
- Операция xor делается просто с помощью ленивого проталкивания
- Чтобы посчитать after(i, x), необходимо использовать двоичный поиск по дереву отрезков, который можно делать одиночным спуском за $O(\log n)$
- Получаем решение с асимптотикой $O(n + q(64 \cdot \log n))$
- Такое решение при аккуратной реализации набирает полный балл

Задача 2-1: Футбол

- В данной задаче нам давалась запись турнира
- Необходимо было решить, является ли турнир полным
- Если турнир является полным, то надо определить, кто победил.

Решение

- **В начале необходимо проверить, что количество матчей равно n(n-1)**
- **В случае, если это верно, нужно вывести YES и определить выигравшую команду**
  - Для этого нужно просто посчитать очки каждой команды
  - Среди команд с наибольшим счетом выводим наименьшую лексикографически.
- **Иначе нам надо доиграть n(n-1)-m матчей**
- **Получаем решение за O(n+m)**

Задача 2-2: Странные антинейтрино

- **Дан массив a**
- **Необходимо было найти перестановку чисел $a_1, \ldots, a_n$ такую, что $\max(a_1+a_2, a_2+a_3, \ldots, a_{n-1}+a_n)$ минимально**

Решение

- Отсортируем числа по возрастанию
- Получили массив $b_1, \ldots, b_n$
- Докажем что расстановка $b_n b_1 b_{n-1} b_2 b_{n-2} \ldots$ является оптимальной

Решение

- Докажем что расстановка $b_n b_1 b_{n-1} b_2 b_{n-2}$... является оптимальной
- Здесь мы используем самое маленькое число как соседа для двух самых больших
- Второе число по возрастанию мы используем в паре для двух оставшихся самых больших и т. д.
- Таким образом, мы подобрали наименьшего соседа из возможных для самого большого числа. Из оставшихся — наименьшего для второго по величине и т. д.
- Почему так оптимально?

Решение

- Почему так оптимально?
- Пусть мы возьмем не наименьшее число b_t, а число b_{t+k} для пары с b_i
- Тогда b_t будет использовано для пары с b_{i-z}
- Однако $\max(b_t+b_i,\ b_{t+k}+b_{i-z}) \le \max(b_i+b_{t+k},\ b_{i-k}+b_t)$
- Что и требовалось доказать
- Таким образом, получили решение за $O(n \log n)$ по времени

Задача 2-3: Быстрые вычисления

- **Необходимо посчитать количество таких пар $l \le i \le j \le r$ таких, что xor на отрезке от i до j равен s**

Наивные решения (34 балла)

- **r-$l \le 2000$ решается перебором всех пар и проверкой за $O(n^2)$**
- **Чтобы решить r-$l \le 3 \cdot 10^5$, посчитаем xor на префиксе**
  - $x[l-1] = 0$
  - $x[l] = l$
  - $x[l+1] = l \oplus (l+1)$
  - $x[l+2] = l \oplus (l+1) \oplus (l+2)$
  - …
  - $x[r] = l \oplus (l+1) \oplus (l+2) \oplus r$

Наивные решения (34 балла)

- Тогда xor на отрезке от i до j равен $x[i-1] \oplus x[j]$
- Задача сводится к тому, чтобы посчитать количество различных пар, xor которых равен s
- Это можно сделать с помощью std::map за $O(n \log n)$

На пути к полному решению (еще 16 баллов)

- **Пусть l=1. Тогда рассмотрим значения x[0], x[1], …**
  - 0, 1, 3, 0, 4, 1, 7, 0, 8, 1, 11, 0, …
- **Закономерность видна: группы вида 4k, 1, 4k+3, 0**
- **Пусть s=0. Применим соображения из предыдущей подгруппы и получим условия на i и j:**
  - либо (i-1) mod 4 = 1 и j mod 4 = 1
  - либо (i-1) mod 4 = 3 и j mod 4 = 3
  - либо i = 0 и j mod 4 = 3

На пути к полному решению (еще 16 баллов)

- **Три случая выше можно покрыть формулой**
- **Для этого придется посчитать количество чисел на отрезке с заданным остатком**
- **Также надо воспользоваться формулой арифметической прогрессии:**
  - $a + (a+1) + \ldots + b = (b-a+1) \cdot (a+b)/2$
- **Рассуждения выше верны и в случае, когда $n \neq 1$**

Полное решение

- В полном решении рассуждения аналогичны, просто надо рассматривать больше случаев
- Для этого надо проверить разные варианты для (i-1) mod 4 и j mod 4 и посмотреть, какой xor на отрезке получится
- Многие случаи довольно тривиальны
- Самый сложный случай — когда s mod 4 = 0 или s mod 4 = 3, и при этом s большое
- Тогда надо подобрать x и y такие, что (4x) xor (4y+3) = s для случая s mod 4 = 3; случай s mod 4 = 0 аналогичен

Полное решение

- Тогда надо подобрать x и y такие, что (4x) xor (4y+3) = s для случая s mod 4 = 3; случай s mod 4 = 0 аналогичен
- Как это сделать?
- Найдем количество чисел x таких, что $l \le 4x \le r$ и $l \le s \oplus 4x \le r$
- Для этого сначала научимся считать количество таких чисел x, что $0 \le 4x \le a$ и $0 \le s \oplus 4x \le b$. Пусть это число равно S(a, b)
- Тогда ответ равен S(r, r) - S(r, l-1) - S(l-1, r) + S(l-1,l-1)

Полное решение

- Осталось научиться считать S(a,b)
- Для этого разобъем отрезок 0..a на O(log a) отрезков вида $[x, x+2^q-1]$, где $x \bmod 2^q = 0$
- В таких отрезках старшие биты фиксированные, а младшие произвольные
- Тогда такой отрезок добавляет к ответу длину пересечения отрезка $[(s \oplus x)/4, (s \oplus x)/4 + 2^q - 1]$ с отрезком $[0, b/4]$

  (здесь дана примерная формула для понимания, в реальном коде границы могут различаться на ±1)

Полное решение

- Таким образом, мы решили всю задачу
- Асимптотика — $O(\log r)$

Задача 2-4: Глобальная сеть

- Дано дерево
- Необходимо удалить две вершины из него
- Связность сети — сумма квадратов размеров компонент
- Необходимо либо минимизировать связность, либо максимизировать ее

Минимизация (15 баллов)

- Здесь все просто
- Можно удалить два листа
- Ответ равен $(n-2)^2$

Наивное решение (еще 19 баллов)

- **$t = 2$, $n \le 300$**
- **Перебираем, какие две вершины удалять и смотрим размеры компонент**
- **При $n \le 3000$ нужно действовать чуть хитрее:**
  - перебираем первую вершину
  - пускаем DFS, насчитываем размеры поддеревьев после удаления вершины
  - теперь, когда мы переберем вторую вершину, мы сможем суммарно за $O(n)$ получить связность после удаления

Полное решение

- Считаем, что t = 2
- Строим центроидную декомпозицию
- Переберем центроидную компоненту такую, что обе удаляемые вершины лежат в разных поддеревьях центроида
- Насчитаем sz[v] — размеры поддеревьев и sm[v] — сумма квадратов размеров компонент, если удалить вершину v и учитывать только компоненты ниже v

Полное решение

- **Пусть мы удалили u и v в разных поддеревьях**
- **Чему тогда равна связность?**
- **Она равна $(n-sz[u]-sz[v])^2 + sm[u] + sm[v]$**
- **Преобразуем выражение**
  - $(n-sz[u]-sz[v])^2 + sm[u] + sm[v] = n^2 + (sz[u]^2 + sm[u] - 2n \cdot sz[u]) + (sz[v]^2 + sm[v] - 2n \cdot sz[v]) + sz[u] \cdot sz[v]$
- **Если считать, что u зафиксировано, то выражение имеет вид $c(u) + d(v) \cdot sz[u]$**
- **Значит, надо применять Convex Hull Trick**

Полное решение

- **Заранее построим прямые в Convex Hull Trick и отстортируем их**
- **Теперь мы умеем «объединять» два поддерева за их суммарный размер**
  - «объединять» — значит рассмотреть все варианты, когда одна вершина в первом поддереве, а вторая — во втором
- **Как объединить все?**
- **Каждый раз объединяем два минимальных в одно большое**
- **Получаем O(n log n) объединений**
- **Значит, вся центроидная компонента обрабатывается за O(n log n)**

Полное решение

- **Вся центроидная компонента обрабатывается за $O(n \log n)$**
- **Итоговая асимптотика — $O(n \log^2 n)$**
- **Надо действовать аккуратно, иначе получится решение за $O(n \log^3 n)$**
  - Например, минимумы в Convex Hull Trick надо искать с помощью двух указателей, а не бинарным поиском
- **Такое решение должно брать полный балл**

Спасибо за внимание!

Вопросы?