Республиканская олимпиада - 2022. Разбор.

Задача 1 - 1. Отель в Олимп-Сити

Идея задачи: Александр Керножицкий

В первую очередь, нам необходимо понять, что такое комната. Утверждение следующее: комната — это непрерывный подотрезок из окон такой, что:

- Существует такое состояние, что его самое левое окно отличается от окна, предыдущего от левого.
- Для остальных окон верно, что для каждого из них окно слева во всех состояниях было таким же.

Это определение окна позволяет сдать эту задачу на полный балл. Для этого давайте посчитаем все окна, для которых выполняется первая часть определения: те окна, для которых окно слева в каком-то из состояний не совпадало.

Получается, что такие окна это своеобразные «разделители» между комнатами, ведь остальные окна, «зажатые» между двумя такими окнами, образуют комнату.

Теперь ясно, что ответ это $w + n$, где w — это и есть то количество окон, которое мы насчитали (прибавляем n, потому что x разделителей образуют $x - 1$ комнату).

Итоговая сложность: $O(knm)$.

Задача 1 - 2. Игра с цифрами

Идея задачи: Антон Керножицкий

Для решения задачи нам пригодится структура данных, которая умеет обрабатывать следующие запросы над бинарной строкой:

- Инвертровать подотрезок бинарной строки. Делать это надо за $O(\log n)$.
- Для подотрезка сказать, сколько в нем единиц. Делать это надо за $O(\log n)$.

Такие запросы умеет выполнять несколько модифицированное дерево отрезков с ленивым проталкиванием. Напомним, что главное в модификациях — это написать метод для ленивого проталкивания. Разберем ключевые моменты в реализации такой структуры данных:

- «Меток» для проталкивания будем делать две: первая метка говорит нам, что необходимо инвертировать подотрезок, а вторая — что необходимо присвоить какое-то значение на подотрезке.
- В дереве будем хранить количество единиц на отрезке.
- Будем поддерживать «метки» для проталкивания таким образом, чтобы в любой момент времени в каждой вершине было не более одной метки. Сделать это не так уж и сложно: разберем случаи:

  1) Пришла метка инвертирования. Тогда просто инвертируем вершину: если была метка присвоения, просто переворачиваем эту метку присвоения, иначе, если метка переворота уже была, то ничего не делаем, иначе добавляем метку переворота.

  2) Пришла метка присвоения. Такая метка просто стирает все предыдущие метки.

В любом из этих случаев потом пересчитываем количество единиц на отрезке, как и в обычном ДО.

Итак, мы построили необходимую структуру данных. Далее задача решается бинарным поиском: мы находим первую такую позицию m, что количество единиц на отрезке $[l, m]$ плюс количество нулей на отрезке $[r - m + 1, r]$ не меньше, чем k. Если такая позиция m существует, и она находится левее, чем середина отрезка, то понятен ответ: присваиваем на отрезке $[l, m]$ все единицы, а на отрезке $[r - m + 1, r]$ все нули.

Если же позиция m находится правее границы отрезка или не существует, то строка входит в т.н. «бесконечный цикл». Заметим, что четность количества единиц изменится на четность числа k после выполнения всей операции переворачивания, так как за один шаг один меняется только один бит.

Посмотрим, как ведет себя этот «бесконечный цикл» в зависимости от четности строки:

Для нечетной строки: `11100` — `11000` — `11100` ...

Для четной строки: `111000` — `110000` — `111000` ...

Теперь ясно, что, для обработки бесконечного цикла, достаточно учитывать четность строки и четность числа k.

Итоговая сложность: $O(n \log^2 n)$.

Испытание: можете ли вы решить эту задачу за $O(n \log n)$?

Задача 1 - 3. Научные эксперименты

Идея задачи: Антон Керножицкий, Андрей Костяной

Есть два решения этой задачи. Приведем первое, более простое решение, а о втором вкратце упомянем в конце.

По сути, частица, которая останется в конце, это самая правая частица в отрезке, которая больше, чем сумма всех предыдущих чисел в отрезке. Тогда будем хранить специальный массив c, где $c_i = a_i - p_i$, где p_i это сумма первых i чисел. Теперь посмотрим, как с помощью этого массива решить задачу.

Во-первых, будем «сдвигать» отрезок на левый край. Для этого достаточно отнять от всех чисел массива p_{l-1}, и рассматривать массив, начиная с элемента l. Теперь оказывается, что необходимый нам элемент — это элемент с максимальным номером, не большим r, такой, что $c_i < 0$.

Чтобы найти его, достаточно написать дерево отрезков на минимум. Обновление одного элемента в исходном массиве есть не что иное, как прибавление числа на суффиксе в массиве c.

Рассмотрим также второе, крайне интересное решение. Рассмотрим все числа a_i в двоичном представлении. Пусть старший бит числа a_i равен b_i. Тогда ясно, что хорошим числом (тем, что мы ищем) может быть только такое число, для которого верно, что на отрезке до него число с таким же b_i встречалось не более одного раза, так как иначе префикс сумма уже была бы больше. Тогда ясно, что всего подходящих чисел на роль «кандидата» на нужное $O(n \log a_i)$. Переберем их все и проверим на выполнение условия. Это решение будет работать за $O(n \log n \log a_i)$.

Итоговая сложность: $O(n \log^2 n)$ *или* $O(n \log n \log a_i)$.

Испытание: можете ли вы решить эту задачу за $O(n \log n)$?

Задача 1 - 4. Игра в слова

Идея задачи: Александр Керножицкий

Для начала рассмотрим, как изменится ответ, если поменять местами две буквы. Если две буквы были одинаковыми, то тогда ответ не изменится. Иначе, если первая буква была меньше второй (пусть первая буква изначально стояла левее), тогда ответ увеличится, иначе — уменьшится. Таким образом получается, что мы можем сравнить две буквы, переставив их в перестановке. Для отгадывания всего слова необходимо $n-1$ сравнение, поэтому такое решение работает для $n \leq 450$.

Далее заметим, что после перестановки количество инверсий изменяется в зависимости от расстояния между обмениваемыми буквами. Т. е. если переставить местами буквы на позициях l и r, то число инверсий изменится либо на 0, либо на $|l-r|$, либо на $-|l-r|$.

Теперь расположим интересующие нас буквы $a_0, a_1, a_2, \ldots, a_k$ на позициях $1, 2, 4, 8, \ldots, 2^k$, причем пусть на позиции 2^i стоит буква a_i. Тогда сделаем следующее: сначала переставим местами a_0 с a_1, затем — a_0 с a_2, a_0 с a_3 и т. д. Здесь идет речь о перестановке букв a_i, а не букв на позиции 2^i, поэтому на всех шагах позиция a_0 может быть различной. В итоге получится, что a_0 стоит на позиции 2^k, а остальные буквы a_i стоят на позициях 2^{i-1}. Далее рассмотрим, как изменится число инверсий после таких перестановок.

Если a_0 является нулем, то в результате каждого обмена количество инверсий лишь возрастет, а если a_i является единицей, то будет убывать. Таким образом, по знаку разности (обозначим эту величину d) мы можем определить a_0: при положительном знаке $a_0 = 1$, при отрицательном — $a_0 = 0$. Если же количество инверсий не изменилось, то тогда все $a_0, a_1, \ldots, a_k$ равны. По битам числа $|d|$ можно теперь сравнить a_0 с остальными a_i: если i-й бит нулевой, то $a_0 = a_i$, иначе $a_0 \neq a_i$. Таким образом, мы сделали примерно $\log n$ сравнений за два запроса.

Мы можем дальше улучшать решение, если научимся сравнивать те же $\log n$ сравнений меньше чем за два запроса. Например, можно научиться сравнивать внутри двух наборов из $\log n$ букв за три запроса. Для этого расположим буквы первого набора на позициях $1, 2, 4, 8, \ldots, 2^{k_1}$, а буквы второго набора — на позициях $6, 7, 9, 13, \ldots, 5 + 2^{k_2}$, после чего сделать три запроса: исходная перестановка, перестановка с изменением порядка для первого набора и перестановка с изменением порядка для второго набора. Аналогично решение можно обобщить на сравнение большего числа наборов из $\log n$ каждый. Для этого необходимо разбить отрезок $[1; n]$ на непересекающиеся «цепочки», каждая из которых имеет вид $p+1, p+2, p+4, p+8, \ldots$. Тогда мы можем буквы из одного набора расположить в одной цепочке. Затем зададим в качестве запроса сначала исходную перестановку, а затем — перестановку, где переставлены буквы из i-го набора. При аккуратной реализации данное решение набирает полный балл.

Задача 2 - 1. Нумерация домов

Идея задачи: Антон Керножицкий

Давайте разберем решение этой задачи за $O(n \log a_i)$. Сразу оговорим, что мы будем решать задачу для $l = 1$, поскольку для решения задачи на отрезке достаточно решить ее на двух префиксах и вычесть один из другого.

Решение этой задачи делится на три части:

- Первое: посчитать, сколько знаков будет в числе, в котором находится r-я цифра. Пусть это k.

- Второе: посчитать, сколько **полных** чисел из k знаков мы пройдем, пока не дойдем до r-й цифры. Если r-я цифра заканчивает число, то это число мы не учтем.
- Ну и, наконец, посчитать, какой будет r-я цифра в оставшемся числе.

На каждом шаге надо уметь вычислять количество единиц на рассмотренных отрезках. Итак, решим все три шага данной задачи:

- Количество цифр во всех числах, у которых есть k знаков, это $k \cdot 2^{k-1}$. А количество единиц в этих числах это $2^{k-1} + 2^{k-2} \cdot (k-1)$ (выводится из того, что первая цифра это единица, а среди остальных поровну нулей и единиц). Отнимаем количество таких чисел от r, пока r не будет становиться отрицательным. Пусть теперь r — номер позиции внутри чисел из k знаков.
- Полное число, на котором мы остановимся, это $\lfloor \frac{n}{k} \rfloor - 1$ (если отрицательное, то полных чисел нет). Пусть номер этого числа среди чисел из k цифр m (ноль нумерация, то есть $2^k + m = r_0$), где r_0 — это изначальное r. Рассмотрим, как вычислять количество единиц в этом блоке. Действительно, в этом блоке есть m единиц. «Уберем» эти единицы, тогда необходимо посчитать количество единиц в числах от 0 до m. Делается это по такой формуле:

  Пусть i-й бит в числе m это единица. Пусть d — количество единичных битов старше этого. Тогда прибавляем к ответу $d \cdot 2^i + 2^{i-1} \cdot i$. Эта формула легко выводится, если, например, раскрыть стандартное ДП.
- Ну и третий пункт самый простой: в лоб считаем префикс этого числа и количество единиц в нем.

Итоговая сложность: $O(n \log a_i)$.

Задача 2 - 2. Лотерейные билеты

Идея задачи: Андрей Костяной, Александр Керножицкий

Для решения этой задачи применим одну формулу. Ясно, что если зафиксировать одну перестановку, то сумма по всем вторым перестановкам не будет зависеть от зафиксированной перестановки. Пусть первая перестановка это перестановка $a_i = i$.

Теперь зафиксируем позиции, на которых элементы совпадают. Таких способов зафиксировать C_k^n. Остальные элементы во второй перестановке должны образовывать т.н. беспорядок. Число беспорядков ищется по известной формуле $(n-k)! \cdot \sum\limits_{i=0}^{n-k} \frac{(-1)^i}{i!}$.

Таким образом, итоговая формула: $n! \cdot \sum\limits_{k=1}^{n} k^2 \cdot C_k^n \cdot (n-k)! \cdot \sum\limits_{i=0}^{n-k} \frac{(-1)^i}{i!}$. Предпросчитав число беспорядков, получим решение за $O(n)$.

Итоговая сложность: $O(n)$.

Задача 2 - 3. Прочная паутина

Идея задачи: Андрей Мищенко

Для начала обойдем граф поиском в глубину и построим дерево обхода. Во время поиска в глубину найдем все точки сочленения и компоненты связности, на которые распадается дерево после удаления точки сочленения.

Теперь рассмотрим дерево обхода графа поиском в глубину и заметим следующее:

- Если вершина не является точкой сочленения, то все остальные вершины образуют единую компоненту.
- Если вершина является точкой сочленения, то все компоненты образуют поддеревья в дереве поиска в глубину.

Таким образом получается, что для эффективного решения требуется уметь искать количество различных значений в поддереве (либо количество различных значений на всем массиве, кроме одного элемента). Построим эйлеров обход дерева, тогда каждое поддерево станет либо отрезков, либо объединением префикса и суффикса массива. Второй случай можно свести к первому, если записать массив эйлерова обхода дважды. Тогда любой запрос на поддереве сводится к запросу на отрезке в таком массиве. Удаление вершины, не являющейся точкой сочленения теперь также является запросом на отрезке.

Остается только научиться вычислять количество различных элементов на отрезке. Для этого будем хранить массив p_i — позиция следующего справа элемента в массиве, который равен i-му. Тогда запрос на отрезке $[l; r]$ сводится к запросу количества на отрезке $[l; r]$ таких p_i, что $p_i > r$. А эту задачу уже можно решить с помощью дерева отрезков либо за $O(n \log^2 n)$, либо оффлайн за $O(n \log n)$.

$\boxed{\textit{Итоговая сложность: } O(n \log n).}$

Задача 2 - 4. Расселение пауков

Идея задачи: Антон Керножицкий

Это задача с открытыми тестами. Можно было пробовать разные стратегии набора баллов, среди них есть перебор с отсечениями и метод имитации отжига.