Задача A. Допрос подозреваемых

Автор задачи и разработчик: Даниил Голов

Частичные решения по задаче включают полный перебор в первой подгруппе или расположение свидетелей в произвольном порядке во второй и третьей подгруппах. Ниже приведем полное решение задачи.

На самом деле достаточно посмотреть на момент времени, в который скучность дела максимальна. Все критические точки, которые будут пройдены, будут пройдены в этот момент, если не раньше. Поскольку скучность больше не увеличится, новые критические точки пройдены не будут. Таким образом, достаточно минимизировать максимальную из скучностей дела.

Скучность дела является префиксной суммой скучностей допрашиваемых. Заметим, что префиксная сумма $a_1 + \ldots + a_n$ точно встретится в конце, поэтому получить максимум меньше ее не получится. А добиться того, чтобы она была максимальной, можно, расположив сначала всех подозреваемых с отрицательной скучностью, а после — всех с положительной.

Таким образом, самым простым решением было отсортировать всех подозреваемых по скучности и вывести в таком порядке, либо же за два прохода по массиву сначала вывести всех с отрицательной скучностью, а за второй проход — с положительной. Такое решение будет работать за время $\mathcal{O}(n+m)$. Количество пройденных критических точек также можно искать линейным проходом по массиву b_i.

Задача B. Исследование улик

Автор задачи: Александр Гордеев, разработчик: Даниил Орешников

По условию был дан массив, и требовалось для некоторых элементов найти, где остановится указатель, стартующий с них. Указатель двигался влево на числа $\leqslant$ текущего, но мог сделать не более k перемещений на равное число.

Первую подгруппу можно было решить любым перебором. В подгруппах с третьей по четвертую работала аккуратно написанная симуляция процесса. Для простоты можно было найти ответ для каждого элемента массива, после чего вывести ответы для тех элементах, которые запрашивались в тесте.

В пятой подгруппе запрещены перемещения между равными числами, таким образом указатель всегда перемещается на строго меньшее число. В таком случае ответ легко найти. Посмотрим на a_i. Если $a_{i-1} < a_i$, то $\texttt{answer}[i] = \texttt{answer}[i-1]$, так как указатель смещается на 1 влево, а дальше повторяет тот же путь. Если же $a_{i-1} \geqslant a_i$, то $\texttt{answer}[i] = i$, так как указатель вообще не двигается.

Чтобы получить полное решение, улучшим решение предыдущей подгруппы. Будем поддерживать количество шагов между одинаковыми числами на пути от i до $\texttt{answer}[i]$. Основная идея такая же, как и была раньше. Однако, если $a_{i-1} = a_i$, указатель по пути сделает на один шаг между равными числами больше, чем если бы он стартовал с a_{i-1}. Если с учетом этого количество шагов между равными числами стало больше k, начнем с ответа $\texttt{answer}[i] = \texttt{answer}[i-1]$, и будем увеличивать его, пока не «избавимся» от лишнего последнего шага по одинаковым числами. Получаем решение за время $\mathcal{O}(n)$ методом двух указателей.

Задача C. Зашифрованное сообщение

Автор задачи: Владислав Власов, разработчик: Даниил Орешников

Для решения первых подзадач можно воспользоваться методом динамического программирования, $\texttt{dp}[i]$ — минимальное число слов, на которые указанным образом можно разбить префикс и суффикс длины i. Пересчет достаточно стандартный: для текущего i перебираем начало последнего слова $j < i$, после чего

1. проверяем, что $s[j:i] = s[|s|-i:|s|-j]$;
2. если они равны, обновляем $\texttt{dp}[i]$ через $\texttt{dp}[j]$.

Проверять строки на равенство можно перебором символов за время $\mathcal{O}(|s|)$, а можно с помощью хешей за $\mathcal{O}(1)$. Эти два решения проходили разный набор подгрупп.

При условии, что исходная строка состоит только из букв ‘a’ и ‘b’ решение не сильно отличается от полного, однако на этой подгруппе может быть проще доказать, что жадный алгоритм действительно приводит к оптимальному результату.

Действительно, решим задачу жадно: будем каждый раз брать минимально возможные по длине равные префикс и суффикс и отрезать их с двух сторон от строки. Докажем, что это приводит к оптимальному ответу.

Пусть это не так, тогда в очередной раз лучший ответ получается взятием большей по длине подстроки с концов строки s. В таком случае s представима как $s = t_1 + q_1 + t_1$ и $s = t_2 + q_2 + t_2$, где $|t_1| > |t_2|$. Но при таком представлении s несложно заметить, что t_2 является как префиксом, так и суффиксом t_1, то есть является ее бордером. Минимальный бордер строки не превышает половины ее длины, поэтому $t_1 = t_2 + r + t_2$, и тогда s можно было представить как $t_2 + r + t_2 + q_2 + t_2 + r + t_2$, что является ее разбиением на еще большее число слов, что и требовалось доказать.

Таким образом, для полного решения достаточно было посчитать хеши префиксов строки s, после чего жадно набирать слова в ее разбиении, проверяя суффикс и префикс на равенство с помощью этих хешей. Общее время решения — $\mathcal{O}(|s|)$.

Задача D. Преступная сеть

Авторы задачи: Константин Бац и Даниил Орешников, разработчик: Даниил Орешников

Переформулируем задачу более простым образом. Дано подвешенное дерево, у каждой вершины есть вес, у каждого ребра есть длина. Требуется найти поддерево максимального веса, достижимое из своего корня за время T.

В первой подгруппе достаточно написать квадратичное решение — запустить обычный `dfs` из каждой вершины вниз по дереву и посчитать сумму весов вершин, достижимых за доступное время.

Во второй подгруппе гарантируется, что дерево является бамбуком, то есть у каждой вершины ровно один ребенок. В таком случае задача на дереве сводится к задаче на массиве — найти отрезок длины не более T с максимальной суммой. Такая задача решается с помощью префиксных сумм и метода двух указателей за линейное время.

Подгруппа с $T \leqslant 10$ и $t_i = 1$ решается подсчетом количества детей у каждой вершины на каждом уровне. Действительно, вершины на расстоянии t_* — это просто потомки вершины на t_* уровней дерева ниже. Находить количество детей на разных уровнях можно было с помощью эйлерова обхода или сливанием результатов, посчитанных для детей.

Следующие две подгруппы расчитаны на упрощенные вариации полного решения. Жюри же известно два полных решения данной задачи. Авторское решение заключается в следующем. Запустим один `dfs` и посчитаем для каждой вершины глубину h — расстояние от нее до корня дерева. Тогда вершина u и ее предок v находятся на расстоянии не более T тогда и только тогда, когда $h[u] - h[v] \leqslant T$.

Представим, что у нас есть декартово дерево всех детей вершины v с высотой вершин в качестве их ключей. Рассмотреть все вершины на расстоянии не более T от v — сделать `split` по ключу $h[v] + T$. Если поддерживать в декартовом дереве для каждой вершины сумму весов поддерева, то ответ на задачу при выборе стартовой вершины v получается как значение этой величины в левой части ДД после `split`.

Осталось понять как получать такое ДД. Будем идти от детей к родителям и сливать результаты для детей. При «перекладывании» элементов из меньшего дерева в большее каждый элемент будет переложен не более $\log n$ раз, асимптотика такого решения — $\mathcal{O}(n \log^2 n)$.

Альтернативное решение: можно заметить, что v попадет в ответ только для тех предков, для которых $h[u] \geqslant h[v] + T$. Множество таких u образует вертикальный путь в дереве, и вершину этого пути можно найти с помощью двоичных подъемов и бинпоиска. Таким образом, если вершина v попадает в ответ для всех вершин на пути $u_1 \to u_2$, то достаточно применить операцию $\texttt{answer}_u \leftarrow \texttt{answer}_u + (a_v + 1)$ для всех $u \in [u_1, u_2]$.

Операцию прибавления на вертикальном пути можно сделать отложенной, прибавив значение к u_2 и вычев его из p_{u_1}. После чего настоящие значения можно посчитать как сумму отложенных

значений в поддереве. В конце надо выбрать вершину v с минимальным $\texttt{answer}_v$. Общее время работы такого решения равно $\mathcal{O}(n \log n)$.