Разбор районной олимпиады по информатике

14 ноября 2020 г.

Задача A. Никакой удачи

Первая подзадача

Перебираем какой жетон выпадет из первого мешка(жетон из отрезка $[a, b]$), а какой из второго(жетон из отрезка $[c, d]$), и считаем количество пар с суммой n.

Сложность решения: $O(n^2)$

Полное решение

Можно заметить, что при фиксированном x, уравнение $x + y = n$ имеет ровно одно решение относительно переменной y: $y = n - x$. Тогда, если зафиксировать жетон, который выпадет из первого мешка, мы можем найти жетон, который должен выпасть из второго, чтобы в итоге получить сумму n. После этого, в зависимости от того, лежит ли жетон y во втором мешке (т.е. $y \in [c, d]$), нужно добавить к ответу единицу.

Сложность решения: $O(n)$

Пример решения на языке C++

Задача B. Палиндромов много не бывает!

Первая и вторая подзадачи

Будем обрабатывать запросы по очереди. Рассмотрим очередной запрос: сначала поменяем местами буквы s_x и s_y. (но нужно не забыть после обработки запроса вернуть всё обратно, т.к. изменения **не сохраняются** между запросами). Осталось проверить является ли строка палиндромом.
Строка t является палиндромом, если $t = t'$, где t' — развёрнутая строка t. $t = t'$ означает, что $t_1 = t'_1, t_2 = t'_2, \ldots, t_i = t'_i, \ldots$. Взяв во внимание то, как переходят символы при развороте строки (в 1-индексации $t'_i = t_{n-i+1}$), получим критерий палиндромности строки: $t_1 = t_n, t_2 = t_{n-1}, \ldots, t_n = t_1$. Таким образом мы научились проверять строку на палиндромность за $O(n)$.

Сложность решения: $O(qn)$

Задача B. Палиндромов много не бывает!

Полное решение

Рассмотрим критерий палиндромости, описанный выше, и назовём индекс i *интересным*, если $t_i \neq t_{n-i+1}$.

Утверждение: Если в исходной строке интересных индексов больше 4, то ответ на все запросы `No`.

Показать это просто: изменеие символов в позициях x и y может повлияеть на интресность не более чем 4-х позиций: x, $n - x + 1$, y, $n - y + 1$, поэтому, если в исходной строке есть более 4-х интересных позиций, то после изменения двух символов останется хотя бы одна из них, следовательно строка не будет палиндромом.

Задача B. Палиндромов много не бывает!

Осталось научиться решать задачу, если количество интересных позиций не превосходит 4. В таком случае, чтобы проверить строку на палиндромность после изменения символов в двух позициях, нужно лишь проверить интересные позиции исходной строки, а также позиции x, $n - x + 1$, y, $n - y + 1$ (суммарно не более 8 позиций). В остальных позициях равенство будет гарантированно выполняться.

Вы могли заметить, что в описанном алгоритме присутствуют лишние проверки, но это никак не влияет на его корректность.

Сложность решения: $O(n + q)$
Пример решения на языке C++

Задача C. Сосчитать их все

Первая подзадача

Любая подматрица однозначно задаётся своим левым верхним и правым нижним углом. Переберём оба угла и посчитаем сумму в образованной подматрице.

Сложность решения: $O((nm)^3)$

Вторая подзадача

Также зафиксируем левый верхний и правый нижний углы, но сумму в подматрице будем считать при помощи префиксных сумм.
Подробнее о префиксных суммах

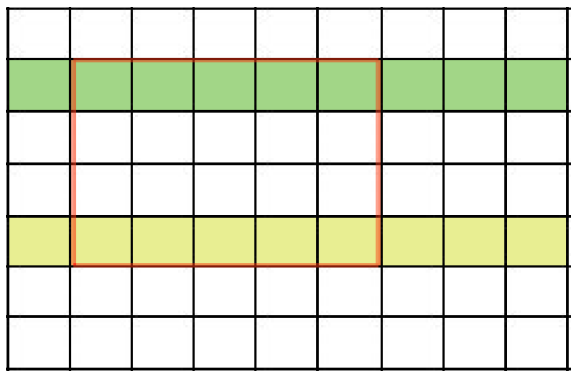
Сложность решения: $O((nm)^2)$

Третья подзадача

Подсказка: Попробуйте зафиксировать строки, где будут находится левый верхний и правый нижний уголы, а затем применить метод «двух указателей».

Задача C. Сосчитать их все

Зафиксируем две строки, в которых будут находится левый верхний и правый нижний углы подматрицы.



В таких условиях каждый столбец можно рассматривать как один элемент (со значением равным сумме значений в соответствующих ячейках столбца). Задача была сведена к аналогичной задаче на массиве.

Задача C. Сосчитать их все

Посчитаем на полученном массиве префиксные суммы. Теперь сумма на любом отрезке $[l, r]$ выражается как $pref_r - pref_{l-1}$.
Давайте зафиксируем правую границу отрезка r. Посчитать количество подотрезков с суммой $\leq k$ равносильно тому, что $pref_r - pref_{l-1} \leq k$, т.е. $pref_{l-1} \geq pref_r - k$. Правая часть — фиксированная величина при фиксированном r.
Осталось научиться отвечать на два вида запросов: добавить число в множество и сказать количество чисел больше либо равных данному в множестве. Это можно делать, например, при помощи дерева отрезков, храня в позиции p количество чисел равных p в множестве и спрашивая сумму на суффиксе.

Задача C. Сосчитать их все

Замечания:

- Для того, чтобы вместить префиксные суммы в дерево, необходимо предварительно их «сжать». Это необходимо пределывать для каждой пары фиксированных строк (картинка двумя слайдами ранее).
- Полученное решение работает за $O(n^2 m \log m)$, чего всё ещё недостаточно. Можно заметить, что асимптотика куда более значительно зависит от числа строк, чем от числа столбцов. По условию $n \cdot m \leq 10^5$, это значит, что $\min(n, m) \leq \sqrt{10^5} \approx 316$. Таким образом, если $n > m$ более выгодно вместо двух строк фиксировать два столбца.

Сложность решения: $O(s^{\frac{3}{2}} \log s)$, где $s = n \cdot m = 10^5$
Пример решения на языке C++

Задача D. Вопросики

Первая-Третья подзадачи

- Каждый раз будем заново пересчитывать НВП
- В зависимости от использованного алгоритма поиска НВП, решение набирает различное число баллов
- Перебор подмножеств за $O(n \cdot 2^n)$ на запрос — 11 баллов
- Динамика за $O(n^2)$ на запрос — 22 балла
- Динамика за $O(n \log n)$ на запрос — 41 балл

Задача D. Вопросики

Важная идея

- Поймём, чему может равняться ответ на запрос
- Обозначим за x длину НВП в исходной последовательности
- Ответ может быть равен:
  - $x + 1$, если изменённое число «встроится» в новую НВП
  - x, по умолчанию
  - $x - 1$, если все НВП проходят через наше число и они все «испортятся»

Задача D. Вопросики

Важная идея

Общая схема любого решения

- Считаем длину НВП, проходящей через новый изменённый элемент
- Если она равна $x + 1$, ответ – $x + 1$
- Определяем, лежал ли исходный элемент во всех НВП
- Если нет, то хотя бы одна из них осталась, и — x
- Иначе ответ — $x - 1$

Задача D. Вопросики

Четвёртая подзадача

- Для каждого числа посчитаем наибольшую длину возрастающей подпоследовательности, начинающейся и заканчивающейся в нём, а также количество таких и таких подпоследовательностей
- В частности, мы знаем длину и количество возрастающих подпоследовательностей во всей последовательности
- Определяем, лежал ли исходный элемент во всех НВП
- Если нет, то хотя бы одна из них осталась, и — x
- Иначе ответ — $x - 1$

Задача D. Вопросики

Как проверить, что числов входит во все НВП в исходном массиве:

- Найдём количество НВП, содержащих это число
- Если через него не проходит НВП ($l[i].length + r[i].length - 1 < lis.length$), то их 0
- Иначе их $l[i].count \cdot r[i].count$
- Если их число равно $lis.count$, то число принадлежит всем НВП
- Чтобы оперировать столь большими числами, можно считать их по какому-нибудь большоу модулю. Вероятность ложного совпадения по большому модулю мала.

Задача D. Вопросики

- За w обозначим наибольшую длину возрастающей подпоследовательности, содержащей x_i-е число в обновлённом массиве.
- Тогда $w = \max(l[i].length : i < x_i, a_i < y_i) + 1 + \max(r[i].length : i > x_i, a_i > y_i)$
- Решение работает за $O(n^2 + qn)$ и набирает 63 балла.

Задача D. Вопросики

Пятая подзадача

- Сожмём числа в исходном массиве.
- Считаем l и r с помощью дерева отрезков
- Находить w для запроса можно с помощью двумерного дерева отрезков

Сложность решения $O(n \log n + q \log^2 n)$ и оно набирает 80+ баллов.

Задача D. Вопросики

Полное решение

Вместо двумерного дерева можно отвечать на запросы offline в два прохода (сначала для всех запросов посчитаем длину НВП слева от изменённого числа, а затем справа).

Сложность решения $O(n \log n + q \log n)$.

Пример решения на языке C++