Стековое кодирование

Автор задачи и разработчик: Даниил Орешников

Сразу отметим, что во всех подзадачах, в которых $\gamma = 1$, было возможно найти точный оптимальный ответ. В остальных подзадачах решением, укладывающимся в ограничения по времени и памяти, можно было найти некоторое приближение ответа.

Основная идея решения — динамическое программирование. Пусть $\texttt{dp}[j][i]$ — минимальное количество действий, необходимое, чтобы получить префикс массива a длины i, если последним действием `print` был выписан полуинтервал $[j, i)$. Переберем k — начало предыдущего выписанного отрезка. Тогда нам необходимо знать $\texttt{lcp}(k, j)$ — наибольшую общую последовательность, начинающуюся в позициях k и j.

Действительно, если до этого в стеке лежала последовательность элементов с полуинтервала $[k, j)$, а теперь лежит $[j, i)$, то надо сначала удалить из стека лишние элементы, а потом добавить не хватающие, оставив общие на месте. То есть на такое преобразование стека тратится $(j-k)-\texttt{common}$ удалений и $(i-j)-\texttt{common}$ добавлений, где $\texttt{common} = \min(j-k, i-j, \texttt{lcp}(k,j))$ — общий префикс этих двух отрезков.

Таким образом, надо обновить $\texttt{dp}[j][i]$ через $\texttt{dp}[k][j] + (i - k - 2 \cdot \texttt{common}_{k,j,i})$ по всем $k < j$. Если каждый раз считать `lcp` за линейное время, весь пересчет динамики будет работать за $\mathcal{O}(n^4)$, если же его предподсчитать за $\mathcal{O}(n^2)$, то динамика посчитается за $\mathcal{O}(n^3)$. Восстановить ответ можно стандартным образом: возвращаясь «назад» по динамике по одному отрезку за шаг.

Точные решения

В **первой и второй подзадачах** решение, отвечающее на запрос явным «извлечением» отрезка и подсчетом динамики для него с нуля, проходило на полный балл, потому что позволяло найти точное решение. Поскольку различных отрезков может быть не больше $\mathcal{O}(n^2)$, то эти две подзадачи проходились динамикой даже за $\mathcal{O}(n^4)$.

Уже соптимизированная до $\mathcal{O}(n^3)$ динамика позволяла также получить точный ответ в **третьей подзадаче**.

Для **подзадач с четвертой по седьмую** требовалось важное наблюдение, которое позволяло соптимизировать подсчет этой динамики до $\mathcal{O}(n^2)$. Вернемся к алгоритму пересчета динамики и поменяем порядок перебора индексов: будем перебирать j и k, а затем будем перебирать i. Теперь заметим, что обновления, которые мы делаем, выглядят следующим образом:

- для $i_0 = j + \min(j-k, \texttt{lcp}(k,j))$ мы обновляем $\texttt{dp}[j][i_0]$ через $\texttt{dp}[k][j] + ((j-k) - (i_0 - j))$;
- для $i < i_0$ с каждым следующим (в порядке уменьшения) i значение правой части увеличивается на 1, так как требует лишнего удаления из стека;
- для $i > i_0$ с каждым следующим (в порядке увеличения) i значение правой части увеличивается на 1, так как требует лишнего добавления в стек.

То есть на самом деле можно сначала для фиксированного j перебрать k и обновить только $\texttt{dp}[j][i_0]$, а затем, вне перебора k, перебрать i и обновить $\texttt{dp}[j][i]$ через $\min(\texttt{dp}[j][i+1], \texttt{dp}[j][i-1]) + 1$.

Это позволяет посчитать ту же самую динамику за $\mathcal{O}(n^2)$, что дает возможность

- в **в четвертой подзадаче** посчитать ее для всего массива a;
- в **пятой подзадаче** посчитать ее для каждого суффикса массива a (суммарно за $\mathcal{O}(n^3)$), после чего для ответа на запрос $[l, r)$ использовать $\texttt{dp}_l[\text{opt}][r-l]$;
- в **шестой и седьмой подзадачах** посчитать ее для всего массива a и вместо запросов $[l, r)$ отвечать на $[0, r-l)$, так как все отрезки равны.

При этом в **шестой и седьмой подзадачах** работает конструктивное решение. Перебрем максимальный размер стека за весь процесс кодирования m. Поскольку мы начинаем с пустого стека, то его длина будет принимать все значения от 0 до m, а значит за k операций вывода можно получить

любую длину массива от 0 до $k \cdot m$. Таким образом, можно для каждого m посчитать величину $m + \left\lceil \frac{r-l}{m} \right\rceil$ и выбрать среди них минимальную.

Неточные решения

Наивным решением можно было набрать определенное ненулевое количество баллов: для каждого отрезка $[l, r)$ просто $r - l$ раз сделаем `push` соответствующего элемента, после чего один раз сделаем `print`.

Можно было решать задачу **жадным алгоритмом**: в общем случае можно было посчитать z-функцию для каждого суффикса массива, после чего с ее помощью разными эвристиками стараться разбить отрезок запроса на как можно более похожие друг на друга. В подзадачах с $a_i = 1$ можно было стараться набрать длину стека, наиболее близкую к $\sqrt{r-l}$, потому что при $n = k^2$ оптимум достигается на $k \times \texttt{push}(1) + k \times \texttt{print}$.

Для решения **восьмой и девятой подзадач** можно было заметить, что оптимальный код выглядит следующим образом:

1. берется какой-то код для «левой части» (от левой границы до 2);
2. последний вывод заменяется на `push`(2), `print` и `pop`;
3. после берется оптимальный код для «правой части» с учетом текущего размера стека.

Одним из способов приблизить ответ был следующий алгоритм:

1. переберем длину стека в конце вывода левой части m в некоторой окрестности корня из ее длины;
2. возьмем соответствующее значение `dp`, предпосчитанного за $\mathcal{O}(n^2)$ на массиве из всех единиц;
3. добавим вывод числа 2 в конец;
4. для правой части найдем такое ближайшее к m число m_2, что длина правой части раскладывается в сумму слагаемых от m до m_2 включительно;
5. упорядочив слагаемые в этом разложении, получим в точности последовательность длин отрезков, выводимых командной `print` — останется только в нужных местах добавить команды `push` или `pop`.

Десятая и одиннадцатая подзадачи требовали подхода, основанного на корневой декомпозиции. Посчитать динамику на всех суффиксах массива не хватит времени и памяти. Но можно посчитать динамику на всех суффиксах, начинающихся с позиций, делящихся на некоторый s.

Иными словами, посчитаем $\texttt{dp}_x[j][i]$ — значения динамики для суффикса массива a, начинающегося в позиции $x \cdot s$. Теперь, для ответа на запрос, если $r - l < s$, посчитаем для него отдельную динамику за $\mathcal{O}(s^2)$, а иначе:

1. найдем ближайшее справа к l число m, делящееся на s;
2. для ответа в правой части возьмем $\texttt{dp}_{\frac{m}{s}}[\text{opt}][r - m]$;
3. для ответа в левой части посчитаем динамику с нуля на отрезке $[l, m)$ за $\mathcal{O}(s^2)$;
4. восстановим ответы и «объединим» их, преобразовав последнее состояние стека для левой части в первое для правой удалениями и добавлениями соответствующих элементов.

Такое решение работает за $\mathcal{O}\left(\frac{n^3}{s}\right)$ на предподсчет и $\mathcal{O}(s^2 + n)$ для ответа на запрос, что при $s \approx \sqrt{n}$ дает $\mathcal{O}\left(n^{\frac{5}{2}} + qn\right)$ времени работы и дает достаточно хорошее приближение ответа в большинстве случаев.