

Задача А. Восстановление данных

Для решения рассмотрим разности между соседними элементами — $a_i - a_{i-1}$.

Все отрицательные разности соответствуют убыванию, а значит, все элементы, образующие эти разности, должны быть в отрезке, на котором выполняется операция (иначе после элемента вне отрезка далее будет элемент меньше по значению). Так же на отрезке не должно быть элементов, которые возрастают — иначе после применения операции они будут убывать.

Поэтому, если массив уже не отсортирован, то нужно выполнить одну операцию от минимального до максимального по индексам элементов, которые образуют отрицательные разности. При этом, если на краях есть участки с равными элементами, то их необходимо включить в отрезок, расширив его (иначе после реверса подотрезка после равного элемента окажется меньший элемент). Если после этого массив остаётся невозрастающим — то ответ -1 .

Так же задачу можно решить вторым способом — получить массив b , который соответствует отсортированному a , и затем найти все индексы i , где $a_i \neq b_i$. Тогда достаточно взять минимальный отрезок, который покроет все индексы, и сделать на нём реверс — после этого остаётся проверить, что массив стал отсортирован.

Задача В. Ожерелье

Для решения задачи необходимо найти максимальное число бусин одного цвета, которые уже идут подряд и которые могут идти подряд после удаления очередного цвета x (случай, когда есть только 1 цвет, рассматривается отдельно).

Для этого сожмём массив в пары $(color_i, cnt_i)$ — по цвету и количеству. Тогда при удалении очередного цвета x достаточно рассмотреть только соседей всех пар, для которых $color_i = x$. Далее необходимо найти среди них максимальный новый отрезок одного цвета. Это можно сделать, например, рассмотрев их по очереди и пытаясь от каждого из них собирать как можно более длинный отрезок из таких же пар, перескакивая через пары с $color_i = x$ (при этом для уже обработанных пар не нужно повторно от них запускаться).

Поскольку у каждого элемента не более 2 соседей, то для цвета x это будет работать за $\mathcal{O}(cnt_x)$, где cnt_x — число блоков бусин цвета x . Поскольку сумма cnt_x не более n , итоговое решение работает за $\mathcal{O}(n)$.

Так же есть альтернативное решение задачи. Поскольку отрезок оптимального цвета (например a) будет чередоваться с другим цветом (например b), то достаточно пройти по массиву и двумя указателями поддерживать текущий блок вида $abab...ba$, где каждая буква соответствует сжатому состоянию с некоторым числом бусин. Тогда необходимо только найти максимум длины одного из чередующихся цветов.

Задача С. Рейтинг

Решим задачу на максимум — задача на минимум будет решаться аналогично после преобразования $x = n + 1 - x$.

Будем называть читателей типа D — динамичными, а типа S — статичными.

Заметим, что если текущая сумма это S и было сделано k голосов, то динамический читатель ставит оценку $\min(n, \max(1, x \cdot (k + 1) - S))$.

Обозначим $\text{clamp}(x, l, r) = \min(r, \max(l, x))$. Тогда после оценки динамического читателя итоговый рейтинг равен $S + \text{clamp}(x \cdot (k + 1) - S, 1, n) = \text{clamp}(x \cdot (k + 1), 1 + S, n + S)$.

Покажем, что в оптимальной процедуре оценивания читатели голосуют в порядке увеличения их оценок. Для этого достаточно показать, что в каждом из четырёх вариантов расположения соседних читателей (S-S, S-D, D-S, D-D) не хуже сначала ставить того, у кого меньшее x . Для этого покажем, что сумма их оценок будет не ниже в данном порядке — поскольку при всех переходах оценка это неубывающая функция по S , то оптимально переставлять читателей так, чтобы увеличивать эту оценку.

1) Для случая двух статичных читателей нет разницы, в каком порядке они ставят оценки, поэтому можно переупорядочить их как угодно.

2) Для случая двух разных читателей с оценками s статического и d динамического получим для порядка S-D: $\text{clamp}((k+2) \cdot d, S+s+1, S+s+n)$, для порядка D-S: $\text{clamp}((k+1) \cdot d + s, S+s+1, S+s+n)$.

Границы одинаковы, но само значение выше в том случае, где большее число s или d добавляется последним.

3) Для случая двух динамических читателей со значениями $d_1 \leq d_2$ получаем для порядка D1-D2 $\text{clamp}((k+2) \cdot d_2, 1+r_1, n+r_1)$, а для порядка D2-D1: $\text{clamp}((k+2) \cdot d_1, 1+r_2, n+r_2)$, где $r_1 = \text{clamp}((k+1) \cdot d_1, S+1, S+n)$, $r_2 = \text{clamp}((k+1) \cdot d_2, S+1, S+n)$.

Очевидно, $r_1 \leq r_2$.

Если $r_1 = r_2$, то границы в обоих выражениях одинаковы, а первый аргумент в порядке D1-D2 не меньше. Значит, порядок D1-D2 не хуже.

Пусть теперь $r_1 < r_2$. Покажем, что итоговая сумма в порядке D1-D2 не меньше $r_2 + 1$, и $(k+2) \cdot d_1$. Тогда $r_1 + n \geq r_2 + 1$, так как оба значения r_1 и r_2 лежат на отрезке $[S+1; S+n]$. Кроме того, r_2 не может быть равно нижней границе $S+1$, так как $r_1 < r_2$. Поэтому $(k+2) \cdot d_2 \geq r_2 + 1$. Следовательно, после D1-D2 итоговая сумма не меньше $r_2 + 1$.

Также $(k+2) \cdot d_1 \leq r_1 + n$: если $r_1 = (k+1) \cdot d_1$, это следует из $d_1 \leq n$; если $r_1 = S+1$, то $(k+1) \cdot d_1 \leq S+1$, откуда $(k+2) \cdot d_1 \leq S+1+n = r_1 + n$.

Значит, итоговая сумма после D1-D2 не меньше $(k+2) \cdot d_1$. С другой стороны, итоговая сумма после D2-D1 не превосходит $\max(r_2 + 1, (k+2) \cdot d_1)$. Следовательно, порядок D1-D2 всегда не хуже порядка D2-D1.

Таким образом, для решения достаточно отсортировать все значения x по возрастанию и применить либо добавляя оценку x для статического читателя, либо применяя формулу для динамического читателя.

Задача D. Коллекционные карточки

Решим задачу, используя динамическое программирование. Пусть $dp[i]$ — количество способов разбить префикс до индекса i включительно на подотрезки, соответствующие условию задачи. Тогда ответ это $dp[n]$.

В $dp[i]$ можно перейти из всех состояний j ($j < i$), таких что MEX подотрезка от $j+1$ до i лежит в нужном диапазоне. При уменьшении j MEX может только увеличиться, а значит, допустимые j для перехода сами образуют подотрезок (возможно, пустой). Тогда, зная его границы, можно пересчитывать очередное значение $dp[i]$, используя префиксные суммы, построенные на динамике.

Для нахождения границ отрезков переходов достаточно поддерживать их указателями: один указатель для максимального отрезка, где MEX не более R , а другой — для минимального отрезка, где MEX не менее L .

Для поддержания указателей достаточно для левого указателя следить, что не все числа от 0 до R встречаются хотя бы один раз, а для правого что нельзя его сдвинуть далее, не удалив все вхождения одного из чисел от 0 до $L-1$. Решение работает за $\mathcal{O}(n)$.

Задача E. Подготовка к олимпиаде

Математическое ожидание итогового опыта складывается из опыта за текущую задачу (это всегда в точности n единиц) и из опыта за решённые до этого задачи — посчитаем эту вторую часть опыта.

Из линейности математического ожидания можно отдельно посчитать, сколько опыта отдельно добавляет каждая пара.

Рассмотрим пару a_i, a_j ($a_i \leq a_j$), идущую в порядке i , затем j ($i \neq j$), и рассчитаем её вклад в математическое ожидание. Пусть $q_i = 1 - p_i$.

Есть всего $n(n-1)/2$ вариантов их расположения (и данный порядок имеет место с вероятностью $1/2$), причем в $n-1$ варианте они идут сразу друг за другом — тогда вклад в математическое ожидание $+1$, в $n-2$ вариантах между ними есть 1 момент, когда задача может быть забыта — то есть вклад $+q_i$. Если между ними k чисел, то таких ситуаций $n-k-1$, а вклад $+q_i^k$. Тогда суммарный вклад равен $\frac{1}{n(n-1)} \sum_{k=0}^{n-2} (n-k-1) q_i^k$.

Данную сумму можно эффективно считать, используя формулы: $\sum_{k=0}^m q^k = \frac{1-q^{m+1}}{1-q}$ и $\sum_{k=0}^m k q^k = \frac{q-(m+1)q^{m+1}+mq^{m+2}}{(1-q)^2}$.

Однако вместо расчёта по формулам также можно было воспользоваться тем, что p_i принимают не более $m = 101$ значений, и для каждого из них рассчитать сумму отдельно за линейное время.

При этом для каждого элемента a_i он даёт один и тот же вклад с любым элементом не меньшим его, если идёт до него — это означает, что достаточно для каждого элемента a_i посчитать число элементов $a_j \geq a_i$ — и столько раз прибавить вклад одного элемента.

Для этого достаточно отсортировать элементы и аккуратно их обработать по порядку. Итоговая асимптотика решения — $\mathcal{O}(n \log n)$ или $\mathcal{O}(n \log n + nm)$ в зависимости от реализации.

Задача F2. Экспедиция (сложная версия)

Для решения задачи рассмотрим следующую динамику на дереве. Пусть $dp_0[i]$ это минимальное количество энергии, которое нужно, чтобы из вершины i обойти всё поддерево и вернуться в вершину i , а $dp_1[i]$ — минимальное количество энергии, чтобы начать путь в вершине i , обойти всё её поддерево и остановиться где-то. Тогда ответом на задачу будет $dp_1[1]$. (При этом для задачи F1 ответом будет $dp_0[1]$ и достаточно считать только его).

Для начала рассмотрим, как пересчитывается $dp_0[i]$. Также в каждой вершине будем поддерживать $delta[i]$ — изменение энергии, если полностью обойти это поддерево.

Тогда рассмотрим всех детей u_j вершины i и их $dp_0[u_j]$ и $delta[u_j]$. Пересчитаем $dp_0[u_j]$ и $delta[u_j]$ с учётом необходимой энергии на рёбрах и получим массив пар $(dp_0[j], delta[j])$ для пересчёта динамики.

Если зафиксировать порядок, в котором будут посещены вершины $j_1, \dots, j_k$, то итоговая минимальная энергия будет равна $\max(dp_0[j_1], dp_0[j_2] - delta[j_1], dp_0[j_3] - delta[j_1] - delta[j_2], \dots, dp_0[j_k] - delta[j_1] - \dots - delta[j_{k-1}])$, поскольку после посещения очередного поддерева j_r текущая энергия увеличивается на соответствующую $delta[j_r]$.

Поймём, какой порядок является оптимальным. Если начинать процесс обхода и всё ещё есть поддерева с $delta[j] \geq 0$, то выгодно их сначала обойти, так как при этом текущая энергия не уменьшается. При этом выгодно обходить их по увеличению $dp_0[j]$, поскольку в каждый момент времени либо остаются ещё поддерева с положительным изменением энергии, либо нет — но если остаются, то всегда не хуже начать с того, где $dp_0[j]$ минимально, так как если текущей энергии не хватит на обход, то значит, изначальной энергии не хватало.

Теперь остаётся рассмотреть порядок тех поддерева, где $delta[j] < 0$.

Для них выгодно рассмотреть все поддерева в порядке убывания $dp_0[j] + delta[j]$. Для этого докажем, что для двух подряд идущих поддерева в обходе оптимально сначала зайти в то, где $dp_0[j] + delta[j]$ выше — тогда из формулы расчёта минимальной энергии будет следовать, что это оптимальный порядок для любого числа поддерева (с отрицательным $delta$).

Пусть есть $d_1, delta_1$ и $d_2, delta_2$, причём $d_1 + delta_1 \geq d_2 + delta_2$. Тогда для порядка 1, 2 получаем энергию $\max(d_1, d_2 - delta_1)$, а для порядка 2, 1 — $\max(d_2, d_1 - delta_2)$.

При этом $d_1 - delta_2 > d_1$ (из $delta_2 < 0$) и $d_1 - delta_2 \geq d_2 - delta_1$ из $d_1 + delta_1 \geq d_2 + delta_2$. Это и означает, что второй вариант (2-1) не лучше первого.

Таким образом, можно получить оптимальный порядок $j_1, \dots, j_k$ и рассчитать $dp_0[i]$.

Для расчёта $dp_1[i]$ требуется выбрать одно поддерево, где маршрут будет завершён. Эту вершину можно перебрать, а для остальных расположить в том же порядке и считать их префикс и суффикс вкладов без одного поддерева (учитывая, что оно идёт последним) — остаётся только выбрать минимум и корректно учесть a_i из текущей вершины — если текущая рассчитанная энергия это E , то в итоге нужно как минимум $\max(0, E - a_i)$ энергии.

Итоговая асимптотика решения $\mathcal{O}(n \log n)$.