Задача A. Скип или не скип

Автор и разработчик задачи: Заварин Александр

Заметим, что нам нет смысла скипать задачи, для которых $b_i \leqslant i$, так как мы можем сдать задачу, получить за неё баллы и следующая задача будет выбираться среди номеров $j < i$, а если мы её скипаем, то мы не получаем баллов и следующую задачу мы выбираем из такого же набора задач или даже меньшего.

Так же заметим, что если мы находимся на задаче i и при этом максимальный номер задачи, которую нам выдавали ранее в процессе олимпиады, $j \geqslant b_i$, то нам нет смысла скипать эту задачу, ведь в следующую задачу после скипа мы могли попасть из j-й просто сдавая задачи.

При таких условиях получается, что все выданные задачи, после завершения олимпиады находятся на некотором префиксе набора задач олимпиады (т.е. существует некоторый номер i от 1 до n, что все задачи с номером $j \leqslant i$ были нам выданы, а задачи с номерами $j > i$ не выданы). Это действительно так, положим i равным максимальному номеру задачи, которую нам выдавали, тогда после выдачи этой задачи, мы не будем больше скипать задачи, так как мы уже доказали, что это не выгодно, а значит мы будем только сдавать задачи и сдадим все задачи с номерами $j < i$, которые до этого не были посещены.

Вместо того чтобы набирать максимальный суммарный балл за сданные задачи, будем стараться набрать минимальный суммарный балл за скипнутые задачи. Будем платить штраф равный a_i за скипнутую задачу и 0, если мы её сдали. Мы знаем, что ответ находится на некотором префиксе, поэтому теперь мы хотим добраться знать для каждой задачи, наименьший штраф, нужный чтобы добраться до неё.

Решим следующую подзадачу. Нам даны те же задачи, и следующие два варианта если мы находимся на i-й задаче:

- Заплатить 0 штрафа и перейти к задаче с номером $i - 1$, если такая существует;
- Заплатить a_i штрафа и перейти к задаче с номером b_i.

Теперь нам разрешается посещать каждую задачу сколько угодно раз. В таком случае, мы можем построить взвешенный ориентированный граф следующего вида:

- В графе n вершин, каждая вершина i соответствует задаче с номером i;
- Для каждого $i > 1$ проведено ребро веса 0 из вершины i в вершину $i - 1$;
- Для каждого i проведено ребро веса a_i из вершины i в вершину b_i.

Тогда наша задача сводится к поиску кратчайшего расстояния от вершины с номером 1 до каждой вершины. Вспомним, что кратчайшее расстояние нам гарантирует, что по пути до вершины i мы посетили каждую вершину максимум 1 раз, а это значит, что если мы добрались до задачи i с некоторым штрафом, то мы можем сдать все задачи на префиксе до i (включительно), так как баллы за все скипнутые задачи будут компенсированы штрафом. Так как мы уже знаем, что оптимальный ответ находится на одном из префиксов, нам нужно для каждого префикса знать сумму баллов за задачи, это легко сделать с помощью префиксных сумм. После этого из всех значений разности префиксной суммы и минимального штрафа, нужного чтобы добраться до вершины i, выберем максимум по всем префиксам i, это и будет ответом.

Кратчайшее расстояние будем искать при помощи алгоритма Дейкстры за $O(n \log n)$

Префиксные суммы считаются за $O(n)$

Итоговая ассимптотика: $O(n \log n)$

Задача B. Отгадай строку

Автор задачи: Чунаев Егор, разработчик задачи: Лазарев Никита

Если суффикс i лексикографически меньше суффикса j, это значит, что первая буква префикса i не больше первой буквы суффикса j. Поэтому давайте расставлять буквы в порядке суффиксов, который дан во входе, начиная с меньших букв.

Задача C. Холмы и ямы

Автор и разработчик задачи: Устименко Глеб

Для начала решим задачу для единственного запроса, где $l = 1$ и $r = n$.

Введём дополнительное ограничение — пусть самосвал должен обязательно стартовать на первом участке, а закончить — на последнем. Посчитаем префиксные суммы массива $a_1, a_2, \ldots, a_n$, пусть они равны $p_1, p_2, \ldots, p_n$. Если $p_n < 0$, то ответ -1, так как не хватит песка засыпать все $a_i < 0$, иначе ответ существует. Пусть x_i — количество раз, которое самосвал проехал между участками i и $i+1$ (в любом из двух направлений). Заметим, что если $p_i < 0$, то $x_i \geqslant 3$. Так как самосвал начинает левее участка $i + 1$, то ему придётся 1 раз пройти через x_i. При этом заметим, что так как $p_i < 0$, то ему придётся вернуться на префикс чтобы выровнять его (ему не хватит песка засыпать отрицательные a_i). Затем ему придётся проехать через x_i в третий раз, так как самосвал должен закончить путь правее участка i. Самосвал может проехать так, чтобы при $p_i < 0$ было $x_i = 3$, а при $p_i \geqslant 0$ было $x_i = 1$. Для этого он просто должен ехать слева-направо, и если $p_i \geqslant 0$, то он просто возвращается влево, пока едет по отрицательным префиксным суммам, зануляя отрицательные a_i. Если он доехал до $p_j \geqslant 0$, то весь префикс слева от него уже занулён (мы поддерживаем такой инвариант), и он просто возвращается вправо до p_i. Весь префикс p_i занулён. Таким алгоритмом мы получаем $x_i = 3$ при $p_i < 0$ и $x_i = 1$ иначе, что и является оптимальным ответом на задачу.

Теперь уберём ограничение на стартовую и конечную позицию самосвала. Пусть самосвал стартует на участке s, а заканчивает на участке f, при этом $s \leqslant f$. Тогда, если $i < s$, то $x_i \geqslant 2$, так как самосвал стартует и заканчивает правее участка i, то есть нужно доехать до него и потом вернуться направо. Аналогично, если $i \geqslant f$, то $x_i = 2$. Для всех $s \leqslant i < f$ выполняются ограничения на x_i, которые были описаны ранее. Как теперь достичь оптимальных x_i? Пусть самосвал доедет от s до 1, зануляя все $a_i > 0$ и собирая песок, а затем проедет вправо до s, засыпая все $a_i < 0$ (здесь мы воспользовались тем, что $p_s \geqslant 0$, доказательство этого будем дальше). От s до f самосвал едет по ранее описанному алгоритму для $s = 1$ и $f = n$. Далее он проедет от f до n, зануляя $a_i > 0$, затем вернётся в f, засыпая $a_i < 0$. Таким образом, мы получили оптимальные x_i. Осталось понять, почему $p_s \geqslant 0$. Пусть это не так, тогда, по нашим оценкам, $x_s \geqslant 3$. Но если мы увеличим s на 1, то эта оценка уменьшится до $x_s \geqslant 2$, значит, нам нет смысла рассматривать данное s. Если мы не можем увеличить s на 1, то это значит, что $s = f$. Тогда для всех i мы имеем оценку $x_i \geqslant 2$. Но тогда мы можем просто пройтись по всем участкам справа-налево и затем слева-направо, и все $x_i = 2$. Случай $s \leqslant f$ разобран. Если $s > f$, то нужно просто развернуть массив a_i и перейти к случаю $s < f$.

Попробуем получше понять, что мы на самом деле сделали, сняв ограничения на стартовую и конечную позицию самосвала. Посмотрим на массив x_i. Изначально в нём есть 1 и 3, мы можем заменить какие-то префикс и суффикс на 2. Ответ на задачу — сумма чисел получившегося массива. Для начала поймём, как пересчитывать 1 и 3. Отсортируем запросы по p_{l-1} (считаем, что $p_0 = 0$). Посмотрим на отрезок $a_l, a_{l+1}, \ldots, a_r$. Его префиксные суммы равны $p_l - p_{l-1}, p_{l+1} - p_{l-1}, \ldots, p_r - p_{l-1}$. Заметим, что при росте числа p_{l-1}, число $p_i - p_{l-1}$ будет уменьшаться. Значит, если рассматривать запросы по возрастанию p_{l-1}, то сначала $x_i = 1$ для любого i, а затем они постепенно становятся $x_i = 3$ (когда p_i становится меньше p_{l-1}). Осталось придумать структуру, которая сможет эффективно искать оптимальную замену на отрезке. Можно просто использовать Дерево Отрезков, в вершине которого будет храниться 5 значений — сумма на отрезке без замен; сумма на отрезке, если заменили всё; сумма на отрезке, если оптимально заменили префикс; сумма на отрезке, если оптимально заменили суффикс; сумма на отрезке, если оптимально заменили префикс и суффикс. Несложно придумать, как объединять 2 отрезка (эта задача очень похожа на общеизвестную задачу о поиске максимального подотрезка единиц на подотрезке бинарного массива). Для решения задачи нужно просто делать замены 1 на 3 в точке и делать запрос на отрезке в этом ДО. Данное решение нужно запустить 2 раза, для оригинального массива a_i и для его развёрнутой версии. Ответ на запрос — минимальный из двух полученных результатов.

Сложность решения — $O((n + q)log(n))$ времени и $O(n + q)$ памяти.

Задача D. Хорошие раскраски 6

Автор задачи и разработчик: Новиков Владимир

Жюри имеет большое число решений, но было выбрано число 20 как примерная граница между совсем плохими решениями и решениями, которые требуют какой-то умной идеи или качественного перебора.

Ниже будет описано решение, которое дает не более 9 повторений для клеток. Построим ответы для $c = 1, 2$ руками. Будем поддерживать инвариант, что для таблички A_c в случае, если мы припишем к ней ее же снизу, справа, сверху или слева, то все цвета останутся связанными. Научимся из таблички A_c получать ответ для $c + 2$ A_{c+2}. Пусть табличка A_c имеет размеры h, w. Построим табличку A_{c+2} с размерами $2h+1, 2w+1$. Для этого в A_{c+2} расставим в углы таблицы A_c, для которых мы попытаемся сохранить связанность крестом между ними. Для правой верхней копии A_c мы включим c-й бит, для левой нижней копии мы включим $c + 1$-й бит и для правой нижней включим c-й и $c + 1$-й биты. Теперь у нас каждого значения хотя бы 1 и остается починить связанность. Для этого для каждой клетки креста достаточно взять ближайшее ненулевое значение, которое находится либо справа от клетки либо снизу. Единственным исключением является центр креста, в который можно поставить значение соответствующее полной маске из c битов. То, что количество цветов не превосходит 9 можно показать рассмотрением случаев (это значение 9 константно начиная с некоторого c).

Задача E. Покупка колы

Автор задачи: Тихомиров Михаил, разработчик задачи: Лепешов Всеволод

Сделаем несколько простых наблюдений про оптимальную стратегию действий. Во-первых, если после нажатия на некоторую кнопку, банка колы не выпала, больше нажимать на эту кнопку нет смысла. Во-вторых, среди кнопок, для которых безуспешного нажатия ещё не было, всегда выгодно нажимать на ту кнопку, которая нажималась меньшее число раз, нестрого это можно обосновать тем, что чем меньше нажатий было сделано, тем больше шанс что следующее нажатие на эту кнопку будет успешным, так как другой информации чтобы различить эти кнопки между собой у нас нет. Из этого однозначно следует как выглядит наша стратегия: давайте отсортируем массив, пусть $a_1 \leqslant a_2 \leqslant \ldots a_n$. Первым действием нажмём на все кнопки по a_1 раз. Ясно, что все эти нажатия дадут нам банку колы, и всего мы наберём $a_1 \cdot n$ банок. Если $k \leqslant a_1 \cdot n$ больше нажатий совершать не нужно. Если же $k > a_1 \cdot n$, нам необходимо сделать ещё как минимум одно нажатие. Так как все кнопки для нас всё ещё не различимы, может получиться так, что это нажатие будет сделано на кнопку соответствующую a_1 и будет неуспешным. Далее нажмём на все оставшиеся кнопки по $a_2 - a_1$ раз, эти нажатия также гарантированно будут успешными. После чего вновь, если k не превышает уже набранного количества банок, мы завершаемся, иначе нам надо сделать как минимум одно нажатие, оно может попасть в пустую ячейку a_2. И так далее. Итого ответом на задачу будет $k + x$, где x такое минимальное число от 0 до $n-1$, что верно: $\sum_{i=0}^{x}(a_{i+1} - a_i) \cdot (n - i) \geqslant k$ (здесь мы считаем $a_0 = 0$). Асимптотика решения: $O(n \log n)$.

Задача F. НВПБП

Автор задачи: Владимир Новиков, разработчик задачи: Евгений Пахомов

Известен стандартный алгоритм поиска НВП при помощи динамического программирования с бинарным поиском. В ходе подсчета такой динамики для каждого индекса $i, 1 \leqslant i \leqslant n$ будет подсчитана длина x_i НВП на префиксе до i, оканчивающейся в позиции i. Аналогичным образом мы можем насчитать длину y_i НВП на суффиксе до i, оканчивающейся в позиции i.

Решение за квадрат. Для каждого $i, 1 \leqslant i \leqslant n$ насчитаем значение b_i — максимальное такое j, что $f(a_{[i+1,j]}) = f(a)$. Сделать такое несложно: если на префиксе до i есть НВП длины $f(a)$, то $b_i = n$. Иначе $b_i = \max\limits_{\substack{j>i, a_j>a_i, \\ x_i+y_j=f(a)}} (j-1)$. Когда $f(a) = f(\overline{a_{[l,r]}})$? Должно быть выполнено $\left(\max\limits_{i<l} b_i\right) \geqslant r$ или есть НВП длины $f(a)$ на суффиксе до $r+1$. Легко обработать эту часть за $O(1)$ на запрос: достаточно насчитать префиксные максимумы на массиве b, а также найти $\min t$, что НВП на суффиксе до t равна $f(a)$.

Значит, для полного решения задачи достаточно научиться более эффективно подсчитывать b_i.

Полное решение. Будем перебирать a_i по убыванию значения блоками из равных элементов. При переборе блока несложно посчитать b_i для всех элементов блока: для этого будем поддерживать массив $best_t = \max i | y_i = t$. Тогда $b_i = best_{f(a)-x_i} - 1$. После этого необходимо пересчитать массив $best_t$, это делается за $O(1)$ для каждого элемента блока.

Таким образом, итоговое решение работает за $O(n \log n)$ (из-за необходимости сортировки массива a).

Задача G. Склеивание массивов

Автор и разработчик задачи: Михненко Алексей

Отсортируем массивы в порядке не убывания суммы элементов. Оказывается, что всегда оптимально склеить массивы именно в таком порядке.

Для доказательства рассмотрим какой-то оптимальный ответ. Заметим, что если в итоговом порядке есть два соседних массива, что сумма элементов левого массива больше суммы элементов правого массива, то мы можем поменять их местами, и количество инверсий не увеличится. Таким образом, мы можем привести любой оптимальный ответ к нашему, меняя местами соседние массивы так, чтобы количество инверсий каждый раз не увеличивалось. Таким образом, такой порядок действительно оптимальный.

Задача H. Ч+К+С

Автор и разработчик задачи: Пархаев Андрей

Рассмотрим какой-нибудь сильносвязанный граф, в котором длины всех циклов кратны k. Заметим, что всегда можно сделать покраску этого графа в k цветов, так что любое ребро ведет из вершины цвета $color$ в вершину цвета $(color + 1) \ mod \ k$. Оказывается, что можно добавлять ребра в этот граф, только если они сохраняют описанный инвариант с цветами.

Сделаем какие-нибудь покраски исходных графов. При фиксированных покрасках довольно легко проверить, можно ли провести требуемые ребра. Для этого заведем соответствующие массивы подсчетов для каждого цвета и каждого класса вершин, после чего будем сравнивать элементы массивов согласно приведенному выше критерию. Но мы могли изначально покрасить второй граф по-другому, например, прибавив к цвету каждой вершины 1 по модулю k. Не трудно проверить, что тогда все значения массивов подсчета для второго графа сдвинуться на 1 по циклу. Аналогично, в зависимости от покраски все значения могли сдвинуться по циклу на произвольное значение. Чтобы решить эту проблему, построим изначальные массивы так, чтобы при фиксированных покрасках их требовалось бы проверить на равенство. Если при какой-то покраске достигается равенство, значит, один массив является циклическим сдвигом другого. Это условие можно проверить, например, используя алгоритм Кнута-Морриса-Пратта.

Задача I. Простая задача для любителей

Автор и разработчик задачи: Александр Понкратов

Проведем радиус-векторы во все точки. Отсортируем все векторы по полярному углу и пронумеруем их в порядке сортировки, тогда задача свелась к стандартной задаче поиска суммы на отрезке на зацикленном массиве.

Задача J. Много игр

Автор и разработчик задачи: Евтеев Тихон

Утв. 1: Пусть мы взяли хотя бы один предмет с $p_i < 100$, тогда утверждается, что сумма w_i по всем взятым эл-ам $\leqslant 200\,000 \cdot \frac{100}{99} = C$. Для доказательства предположим противное, и попробуем убрать из взятого множества любой элемент с $p_i < 100$, обозначим $q_i = \frac{p_i}{100}$.

Ответ был $(W + w_i) \cdot Q \cdot q_i$, а стал $W \cdot Q$. Ответ увеличился, если $w_i \cdot Q \cdot q_i < W \cdot Q \cdot (1 - q_i)$, то есть $w_i \cdot q_i < W \cdot (1 - q_i)$, то есть $w_i \cdot p_i < W \cdot (100 - p_i)$, что верно так как $w_i \cdot p_i \leqslant 200\,000$, а $W \cdot (100 - p_i) > 200\,000$

Выделим все предметы с $p_i == 100$, если их сумма $> C$, то мы знаем ответ, иначе для каждой суммы весов найдем максимальную вероятность, с которой можно получить такой вес, используя динамику, похожую на рюкзак.

Чтобы это сделать, уменьшим кол-во рассматриваемых предметов. Для каждого p в ответе будет какой-то префикс элементов с таким p, в порядке сортировки по убыванию w_i. Если в оптимальном ответе находится c_p эл-ов с $p_i == p$, то $c_p \cdot q^{c_p} > (c_p - 1) \cdot q^{c_p - 1}$, иначе наименьший элемент точно выгодно выкинуть. Переписав неравенство получаем $c_p \cdot q > c_p - 1$, то есть $c_p < \frac{1}{1-q}$. Таким образом среди элементов с данным p достаточно оставить для рассмотрения $\frac{100}{100-p}$ лучших, или порядка 450 $(99 \cdot \ln 99)$ предметов по всем p.

В конце достаточно пройтись по динамике и найти клетку с наибольшим ответом.

Итоговое время работы $C \cdot 99 \cdot \ln 99$

Задача K. Древо жизни

Автор задачи: Сафонов Иван, разработчик задачи: Ромашов Федор

У этой задачи существует несколько в разной степени схожих решений. Опишем одно из них.

Применим следующий жадный подход. Будем строить ответ, объединяя ответы для поддеревьев. Для этого сделаем обход в глубину, при выходе из вершины возвращается тройка $(ans, up, bonus)$, где ans — минимальное число путей, чтобы покрыть все пары соседних ребер в поддереве (учитывая ребро наверх), up — число ребер наверх, $bonus$ — число путей, которые соединены в какой то вершине поддерева, но их можно разъединить на два пути вверх, не нарушив покрытие.

Тогда если мы находимся в вершине v и получаем из сына u тройку $(ans_u, up_u, bonus_u)$, то необходимо увеличить up_u хотя бы до $deg_v - 1$ (чтобы удовлетворить покрытие). Далее как бы уменьшить его на $deg_v - 1$, подразумевая, что все такие пары мы удовлетворили. При этом пока что ans_u и $bonus_u$ суммируем, и вычитаем из ans при соединении. Увеличение делаем сначала с помощью $bonus$ $(ans_u += 1, up_u += 2, bonus_u -= 1)$, а далее просто добавляя новые пути.

После этого у нас остались лишние пути (up), ведущие в v, которые возможно можно объединить, чтобы уменьшить ответ. Это представлено набором $U = \{up_{u_1}, up_{u_2}, \ldots, up_{u_k}\}$. Если $\max(U) * 2 \leqslant sum(U)$, то можно объединить все пары (оставив максимум 1 путь наверх) в U, добавляя эти пути в $bonus_v$. Иначе, все $up_{u_i} \neq \max(U)$ увеличиваем с помощью $bonus_{u_i}$, пока не выполнено $\max(U) \cdot 2 \leqslant sum(U)$. Наконец возвращаем $up_v = (sum(U) \mod 2) + deg_v - 1$, если условие выполнено, и $up_v = 2 \cdot \max(U) - sum(U) + deg_v - 1$, а $bonus_v$ и ans_v как сумму, которая возможно менялась в процессе. Не забудьте учитывать пути, которые объединились в v.

Отдельно нужно обработать корень, так как там нет путей наверх.

Для доказательства этого решения можно рассмотреть $dp_{v,up}$ — минимальное число путей, чтобы покрыть поддерево v, если наверх выйдет up путей. Не трудно заметить, что тройка $(ans, up, bonus)$ в жадном решении описывает все оптимальные состояния динамики.

P.S. Строгие доказательства остаются в качестве упражнения читателю.

Задача L. Выгодный процент

Автор и разработчик задачи: Андреева Елена

Если $a \geqslant b$, то выгодный вклад удастся открыть на всю сумму имеющихся денег. В противном случае, некоторую сумму x придется положить на другой вклад. Тогда $a - x = b - 2x$, отсюда $x = b - a$, а максимальная сумма, на которую может быть открыт выгодный вклад — $2a - b$. Следует также учесть, что эта величина должна быть не менее 100 000 рублей. Если это не так, то ответ равен нулю.