

Задача Е. Сильная связность наносит ответный удар

Авторы задачи: Роман Первутинский, Денис Мустафин

Разработчик задачи: Роман Первутинский

Подгруппа 1

Для каждого подмножества рёбер проверим, поменяются ли КСС, если развернуть лежащие в нём рёбра. Хорошее подмножество не содержит подмножеств, для которых КСС не поменяются. Для каждого подмножества проверим, является ли оно хорошим, перебрав все его подмножества. С поиском КСС алгоритмом Тарьяна или Косарайю получаем решение за $O(3^m + 2^m \cdot m)$.

Подгруппа 2

Для каждого множества нам нужно проверить, содержит ли оно одно из «плохих» подмножеств. Воспользуемся ДП по подмножествам:

$$\text{is_good}[S] = \neg \text{can_reverse}[S] \wedge \bigwedge_{e \in S} \text{is_good}[S \setminus \{e\}]$$

Итого получаем решение за $O(2^m \cdot m)$.

Подгруппы 3, 4 (ациклический граф)

Рассмотрим некоторое ребро $e = (u, v)$. Если не существует другого пути из u в v , e можно развернуть, и граф останется ациклическим. Таким образом, e не может лежать в хорошем множестве.

Пусть теперь существует другой путь из u в v . Тогда мы можем восстановить направление e из направлений рёбер на этом пути, то есть направление на самом e можно стереть. Множество из всех таких рёбер e — хорошее. Это множество является единственным максимальным.

Чтобы проверить существование пути, не проходящего через ребро, удалим ребро из графа и запустим поиск в ширину. Итого получаем решение за $O(m^2)$.

Подгруппа 5 (гамильтонов цикл)

Назовём все рёбра, не лежащие на гамильтоновом цикле, *прямыми*. Поскольку при их развороте сильная связность графа не нарушается, они не могут лежать в хорошем множестве.

Прямые рёбра покрывают некоторые отрезки цикла. Рассмотрим пересечение нескольких таких отрезков (возможно, не связное, поскольку отрезки циклические). Утверждение: все рёбра цикла, принадлежащие этому пересечению, можно развернуть, и граф останется сильно связным.

Таким образом, хотя бы одно ребро цикла на пересечении отрезков должно не лежать в хорошем множестве. Заметим, что если мы можем взять ещё один отрезок в наше пересечение, и оно уменьшится, мы получим более сильное условие. При этом «минимальные» пересечения, которые больше нельзя уменьшить, не пересекаются.

Выберем на каждом минимальном пересечении некоторое ребро, и возьмём множество из всех рёбер, кроме прямых и выбранных. Почему оно будет хорошим? Поскольку пересечения минимальны, в графе их рёбра образуют цепочки между вершинами степени 2. Поскольку в итоговом графе не должно быть стоков и истоков, направления на всех рёбрах цепочки восстанавливаются однозначно.

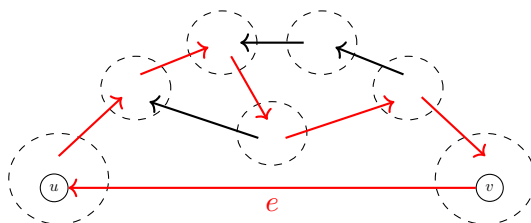
Полученное множество будет максимальным хорошим. Количество способов — произведение размеров минимальных пересечений.

Подгруппа 6 (сильно связный граф)

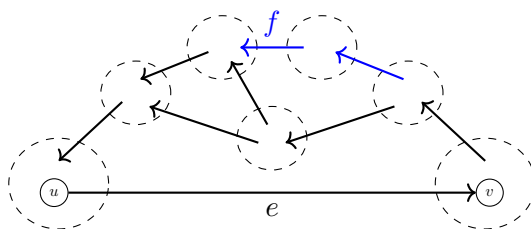
Обобщим идею «минимальных» множеств, которые не должны полностью лежать в хорошем, на общий случай.

Рассмотрим ребро $e = (u, v)$. Опять, если при его удалении граф останется сильно связным, e не может лежать в хорошем множестве. Иначе при развороте e граф распадётся на несколько КСС. Назовём множество рёбер, принадлежащих конденсации, $S(e)$ ($e \in S(e)$).

Пусть мы развернули e и хотим развернуть ещё некоторое подмножество рёбер, чтобы граф остался сильно связным. Мы обязаны развернуть некоторое подмножество $S(e)$, чтобы появился путь из u в v .



Заметим, что если $f \in S(e)$, то $S(f) \subseteq S(e)$. Если $S(f) = S(e)$ для всех $f \in S(e)$, то рёбра в $S(e)$ образуют цикл в конденсации. Назовём такое $S(e)$ *нужным* — это эквивалент «минимального» множества из прошлого решения.



Рассмотрим некоторое множество рёбер, которое можно развернуть, и граф останется сильно связным. Пусть e — ребро из этого множества с минимальным $S(e)$. Тогда $S(e)$ — нужное. Действительно, пусть это не так, тогда существует такое ребро $f \in S(e)$, что существует путь из v в u в графе конденсации, не проходящий через f . Тогда $S(f) \subset S(e)$; противоречие.

Поскольку $S(e)$ образует цикл, $S(e)$ целиком лежит в рассматриваемом множестве. Поэтому любое множество рёбер, которое можно развернуть без нарушения сильной связности, содержит какое-то нужное.

При этом можно развернуть само $S(e)$, чтобы граф остался сильно связным. Таким образом, A хорошее $\iff A$ не содержит ни одно нужное $S(e)$.

Для этого нам нужно не взять в A по одному ребру из каждого нужного множества. Поскольку они не пересекаются, сделать это можно как угодно. Максимальное хорошее множество состоит из всех рёбер, кроме не влияющих на сильную связность и выбранных из нужных множеств. Количество способов — произведение размеров нужных множеств.

Полное решение

Решим задачу отдельно для каждой КСС и для графа конденсации. В графе конденсации может быть несколько рёбер, соединяющих две КСС; из них мы должны не взять в хорошее множество одно любое. При этом домножим ответ на их количество.

Итоговое решение работает за $O(m^2)$.

Решение на частичные баллы

Из полного решения следует интересный факт: любое хорошее множество лежит в некотором максимальном хорошем. Это позволяет написать следующее жадное решение:

Пусть мы уже выбрали некоторое хорошее множество. Теперь мы хотим понять, можно ли добавить в него ещё какое-то ребро. В каком случае мы не можем этого сделать? В том, когда можно развернуть наше ребро и ещё какое-то подмножество уже стёртых, сохранив сильную связность.

Как уже сказано выше, если после разворота нашего ребра (u, v) перестанет существовать путь из u в v , то нам нужно развернуть какие-то ребра из уже взятого множества, чтобы этот путь появился. Проверить, можно ли это сделать, мы можем, ища путь из u в v в графе, где для каждого стёртого ребра мы добавили развёрнутое.

Если такого пути нет, то ребро (u, v) можно стереть. Почему, если такой путь существует, мы не можем это сделать? Утверждается, что после разворота ребра (u, v) и рёбер на найдённом пути граф останется сильно связным. Действительно, мы получили цикл из этого пути и развёрнутого ребра (v, u) . Для всех рёбер, которые мы развернули на этом пути, достижимость между концами ребра осталась, поскольку все вершины на цикле достижимы друг из друга.

Рассмотрев все рёбра графа в любом порядке, мы построим некоторое максимальное хорошее множество.

Задача F. Лучший бегун

Автор задачи: Сергей Князевский

Разработчик задачи: Алексей Дацковский

Сначала разберём подгруппы, в которых не используется основная идея из общего решения:

Подгруппа 2: Всегда выиграет первый бегун: он должен после каждой дорожки перемещаться на одну влево (пока это возможно). Так он пробежит больше всех дорожек, так как он бегал по максимально коротким дорожкам.

Подгруппа 3: Можно решить с помощью динамического программирования. $dp[i][j]$ (где i — текущая дорожка бегуна, а j — оставшееся время) равно максимальному количеству дорожек, которое можно пробежать, начиная с такой ситуации.

Подгруппа 5: Всегда выигрывает бегун, который начинает на дорожке с минимальной длиной, так как он может пробежать максимальное число раз по самой короткой дорожке.

Для остальных подгрупп нужно понять, как будет выглядеть оптимальный маршрут бегуна. Пусть он начинает на дорожке i .

- Он должен закончить на самой короткой дорожке, на которой он был (обозначим её j). Если бы он остался на самой короткой дорожке и бегал по ней до конца времени, то пробежал бы не меньше дорожек, чем в любом другом сценарии.
- Также он должен максимально быстро перейти от i до j , и потом бегать только по j . Так он пробежит максимальное число дорожек среди всех сценариев.

Пусть у нас есть бегун с начальной дорожкой i и конечной дорожкой j . Определим количество дорожек, которое он пробежит. Не умаляя общности, пусть $i < j$. Тогда он потратит $a_i + a_{i+1} + \dots + a_{j-1}$ времени, чтобы дойти до дорожки j , а потом оставшееся время будет бегать по j . Значит, он пробежит $j - i + \lfloor \frac{T - (a_i + a_{i+1} + \dots + a_{j-1})}{a_j} \rfloor$ дорожек (если время от дорожки i до дорожки j не превышает T). Если поддерживать префиксные суммы массива a , то мы сможем вычислять количество дорожек за $O(1)$.

Этого достаточно, чтобы решить **подгруппу 1**: для каждой начальной позиции перебрать все конечные позиции и найти максимальное число дорожек.

Подгруппа 4: здесь нужно перебирать в качестве конечных позиций только те дорожки, которые строго короче всех дорожек между начальной и ней. Так как длины дорожек не превосходят 20, то для каждой начальной позиции мы рассмотрим не более 20 конечных дорожек слева и справа. А чтобы быстро находить ближайшую слева (или справа) дорожку короче текущей, можно предпочитать их заранее.

Подгруппа 6: осталось применить небольшой трюк. Будем наоборот для каждой конечной позиции j находить бегуна, который пробежит максимальное число дорожек и закончит на дорожке j . Можно показать, что это будет либо ближайший бегун слева от дорожки, либо ближайший бегун

справа (или бегун, начинающий на этой дорожке, если такой есть) — другие бегуны потратят больше времени на то, чтобы добраться до j (если j — оптимальная конечная позиция). Значит, нам нужно перебрать не более 2 начальных дорожек для каждой конечной дорожки.

Задача G. Переворот карт

Автор задачи: Алексей Михненко

Разработчик задачи: Артём Агафонов

Пусть на столе осталось k карт, все из которых являются односторонними или перевёрнутыми двусторонними. Тогда игроки могут совершать только первое действие, а значит победитель определяется однозначно. Если k нечётное, то побеждает первый игрок, иначе — второй.

Пусть теперь на столе лежат односторонние и перевёрнутые карты, а также одна неперевернутая двусторонняя карта, действие с которой будет совершать текущий игрок. Заметим, что в такой позиции текущий игрок побеждает. Действительно, каждый из двух возможных ходов оставит после себя число карт разной чётности, а значит текущий игрок может совершить ход, который по прошлому замечанию однозначно будет гарантировать ему победу. Получаем, что в игре побеждает тот игрок, который сможет взять последнюю двустороннюю карту.

Прошрое замечание позволяет упростить задачу. Обозначим за x наибольшее число, записанное на лицевой стороне двусторонних карт. Тот игрок, который будет играть с картой x , побеждает. Тогда:

1. Мы можем удалить все односторонние карты с числом больше x . Их наличие никак не влияет на номер игрока, который будет совершать ход с картой x .
2. Мы можем заменить все двусторонние карты, имеющие число на обратной стороне больше x , на односторонние карты. Заметим, что вне зависимости от того, будет убрана или перевёрнута данная карта, после хода она не будет влиять на номер игрока, который возьмёт карту x . Значит мы можем считать, что это просто односторонняя карта с числом, записанным на лицевой стороне исходной двусторонней карты.
3. После двух прошлых изменений в момент, когда ход дойдёт до карты x , никаких других карт не останется, а значит текущий игрок обязательно сбросит её, чтобы обеспечить себе победу. Поэтому мы можем заменить данную карту на одностороннюю с числом x .

Данное упрощение задачи не привело к изменению победителя игры, но число двусторонних карт на столе уменьшилось хотя бы на 1.

Так как после каждого упрощения задачи число двусторонних карт уменьшается, то, применив данную операцию некоторое конечное число раз, мы получим позицию, содержащую только односторонние карты. Тогда, воспользовавшись первым замечанием, мы сможем определить победителя данной игры.

Заметим, что, чтобы упростить задачу, необходимо перебрать все карты, то есть сделать $O(n+m)$ действий. Так как после этого число двусторонних карт уменьшится хотя бы на 1, то суммарно будет не более $O(n)$ упрощений. Таким образом мы получили решение, имеющее сложность $O(n(n+m))$.

Попробуем ускорить предыдущее решение. Вместо того, чтобы честно совершать очередное упрощение, будем просто искать двустороннюю карту, которая будет последней в новой игре. Заметим, что данная карта обладает следующим свойством: на её обратной стороне записано число меньше текущего x , и среди таких карт она обладает наибольшим числом на лицевой стороне. Тогда, отсортировав двусторонние карты в порядке уменьшения числа на лицевой стороне за $O(n \log(n))$, мы будем перебирать их в таком порядке и в тот момент, когда будет выполнено условие на число, записанное с обратной стороны, будем обновлять значение x , тем самым переходя к следующей карте. Совершив такое действие, мы найдём число x — число на лицевой стороне двусторонней карты, которая будет последней в последнем упрощении. Посмотрев на чётность количества двусторонних карт, имеющих на лицевой стороне число не большее x , и односторонних карт, имеющих число меньше x , мы определим победителя игры. Сложность данного решения — $O(n \log(n) + m)$.

Используя сортировку подсчётом, можем получить асимптотику $O(n + m)$, но в данной задаче этого не требовалось.

Задача Н. Порядковая статистика

Автор и разработчик задачи: Валерий Родионов

В подзадаче 1 достаточно промоделировать процесс из условия за $(mn \log n)$, m раз сортируя массив и уменьшая на 1 последние k элементов.

В подзадачах 2-4 выполняется $k = 1$, то есть на каждой из m операций максимальный элемент в массиве уменьшается на 1. Предположим, что наибольший элемент в массиве после m операций равен x . Тогда выполняется $m \leq \max(a_1 - x, 0) + \dots + \max(a_n - x, 0) = T(x)$. Нетрудно заметить также, что наибольшее значение в массиве после m операций равно наименьшему x , удовлетворяющему этому условию. Найдем это x с помощью бинарного поиска. После $T(x)$ операций все элементы массива, которые были больше x , станут равны x , а остальные не изменятся. Мы знаем, что меньше максимум в массиве после этого уже не станет, а значит оставшиеся $m - T(x)$ операций будут уменьшать элементы, равные x . Мы знаем количество оставшихся операций, поэтому просто уменьшим на 1 любые $m - T(x)$ элементов, равных x , и получим финальный массив. Такое решение работает за $O(n \log \max a_i)$ и проходит подзадачу 2. Чтобы пройти подзадачи 3 и 4, нужно научиться быстро считать величину $T(x)$ для заданного x . В подзадаче 3 нет изменений, поэтому достаточно насчитанных префиксных сумм на отсортированном массиве a , а в подзадаче 4 для поддержания изменений использовать дерево отрезков, дерево Фенвика или декартово дерево, получая итоговую асимптотику $O(n \log m \log n)$.

В подзадаче 5 выполняется $k = 2$. Отсортируем массив a . Пока a_n больше максимального из $a_1, a_2, \dots, a_{n-1}$, задача эквивалентна случаю $k = 1$, так как на каждом шаге уменьшается a_n и максимальный из $a_1, a_2, \dots, a_{n-1}$. Найдем с помощью бинарного поиска и решения для $k = 1$ момент, когда a_n станет равен максимуму на префиксе $a_1, a_2, \dots, a_{n-1}$. Эта часть решения работает за $O(n \log^2 C)$. Снова отсортируем массив, теперь выполняется $a_n = a_{n-1}$. С этого момента отсортированный массив после каждой операции будет выглядеть следующим образом: некоторый неизменный префикс массива, после идет суффикс, состоящий из чисел x или $x + 1$ для некоторого x , причем длина суффикса не меньше 2. Для простоты скажем, что длина суффикса четная, нечетный случай разбирается более-менее аналогично. Если L — длина суффикса, и минимальное значение на суффиксе больше максимального на префиксе, то за $L/2$ операций все элементы на суффиксе уменьшатся на 1. В тот момент, когда x станет равен максимуму на префиксе, длина суффикса увеличится на 1. Всего расширений суффикса будет не больше n . Если аккуратно обработать все расширения, то эта часть решения будет работать за $O(n)$. Получаем решение за $O(n \log^2 C)$.

В подзадаче 6 выполняется $m \leq 10^6$. Научимся обрабатывать одну операцию за $O(\log n)$, при этом поддерживая массив отсортированным. Пусть массив перед операцией отсортирован, и на позиции $n - k + 1$ стоит число x . Пусть количество чисел в массиве, больших x , равно k_1 . Тогда на этой операции уменьшатся все элементы массива, большие x , а также $k - k_1$ элементов, равных x . Чтобы поддержать отсортированность массива, среди элементов, равных x , нужно выбрать $k - k_1$ из них, стоящих в массиве левее всего. Находить количество элементов, больших x , и первый элемент не меньший x , а также уменьшать все элементы на отрезке на 1 можно с помощью дерева отрезков за $O(\log n)$. Итого, получаем решение за $O(m \log n)$.

В оставшихся подзадачах так или иначе используется основная идея полного решения.

Основная идея: представим, что на каждой операции мы уменьшаем не k наибольших элементов, а любые k элементов по нашему выбору. Также будем считать, что операции применяются так, чтобы поддерживать отсортированность массива (обратите внимание, что это требование никак не ограничивает операции, если рассматривать их как операции над неупорядоченным набором элементов массива). Пусть b — массив, полученный применением m раз операции из условия, а c — массив, полученный применением произвольной последовательности из m операций, где каждая операция уменьшает k любых элементов на 1. Поймем, чем массив b , который мы хотим найти, выделяется среди всевозможных массивов c . Пусть массивы b и c отсортированы. Тогда утверждается, что $b_1 + \dots + b_i \geq c_1 + \dots + c_i$ для всех i от 1 до n , то есть операции из условия в каком-то

смысле «максимизируют» полученный массив. Отсюда также следует, что массив $b_1, b_2, \dots, b_n$ является лексикографически максимальным массивом, который можно получить такими операциями. Доказательство: рассмотрим последовательность операций, приводящую к максимальной возможной сумме $c_1 + \dots + c_t$. Посмотрим на последнюю плохую операцию (ту, которая уменьшала на 1 не k максимальных элементов). Тогда если элементы i, j , такие, что до этой операции $a_i < a_j$ ($i < j$, так как операции сохраняют порядок элементов), при этом i -й элемент был уменьшен на 1, а j -й не был. Пусть после применения всех операций массив стал равен $b_1, b_2, \dots, b_n$. Если $b_i < b_j$, поменяв в рассматриваемой плохой операции i -й и j -й элемент местами мы получим в конце не меньшее значение $b_1 + b_2 + \dots + b_t$. Если $b_i = b_j$, то после плохой операции была хотя бы одна, которая уменьшила a_j и не уменьшила a_i . Поменяем местами a_i и a_j в обеих операциях, получим тот же массив b , но при этом увеличится сумма элементов, которые были уменьшены на 1 в плохой операции. Количество операций и сумма элементов конечны, поэтому такой процесс когда-нибудь закончится, и мы получим последовательность, состоящую из хороших операций и максимизирующую сумму $c_1 + \dots + c_t$.

Мы поняли, что b максимизирует префиксные суммы по всем массивам c , получаемым с помощью t операций вычитания 1 из k элементов. Следующая задача позволяет легко описать все подходящие массивы c , по которым мы будем искать этот максимум.

Вспомним такую классическую задачу: как проверить, что в массиве $d_1, d_2, \dots, d_n \geq 0$ можно m раз уменьшить k разных (но не обязательно различных по значению), элементов, так, чтобы все элементы массива остались неотрицательными. Ответ: это возможно тогда и только тогда, когда $\min(d_1, m) + \dots + \min(d_n, m) \geq mk$. Доказательство как обычно остается в качестве упражнения.

С помощью описанного решения классической задачи можно проверять, можно ли получить из a после выполнения m операций массив c , такой, что $c_i \geq b_i$ для всех i . Тогда сделав последовательно бинарный поиск по b_1 , потом по b_2 с фиксированным b_1 и так далее мы найдем максимальный лексикографически массив, который можно получить из a за m операций, а по факту выше он и будет ответом. В зависимости от эффективности реализации получается решение за $O(n^2 \log m)$, $O(n^2)$, $O(n \log m)$ или $O(n)$, проходящее подзадачу 7, и, возможно, 8.

Научимся считать максимальную возможную префиксную сумму $a_1 + a_2 + \dots + a_i$, которую можно получить после m операций. Предположим, что мы уменьшили элемент a_i до x . Тогда элементы $a_1, a_2, \dots, a_{i-1}$ должны быть уменьшены хотя бы до x , а элементы $a_{i+1}, \dots, a_n$ могут быть уменьшены не ниже, чем до x , потому что порядок элементов в массиве должен сохраняться. Рассмотрим две величины $f(x)$ — минимальное число, на которое мы должны суммарно уменьшить элементы на префиксе до i -го, и $g(x)$ — максимальное число, на которые мы можем уменьшить элементы на суффиксе с $i+1$ -го, чтобы сохранить неубывающий порядок элементов в массиве. Так как суммарно за m операций все элементы уменьшатся на mk , то на префиксе суммарное уменьшение будет не меньше $\max(f(x), mk - g(x))$. Легко видеть, что эта оценка достигается, поэтому в случае, когда мы уменьшаем a_i до x , максимальная возможная префиксная сумма равна $a_1 + a_2 + \dots + a_i - \max(f(x), mk - g(x))$. Найти максимум такой функции по x можно с помощью бинарного поиска, так как функция $f(x)$ невозрастает, а функция $mk - g(x)$ неубывает. Без запросов изменения функции $f(x)$ и $g(x)$ можно считать с помощью префиксных сумм на отсортированном массиве, что проходит подзадачи 9 и 10. Для обработки запросов изменения можно использовать дерево отрезков, дерево Фенвика или декартово дерево, что проходит 11 и, в зависимости от эффективности реализации, 12 подзадачу. Время работы решения $O(n \log m \log n)$.