Задача 5. Восстание газонокосилок

Авторы: Азат Сафиуллин, жюри
Разработчик: Николай Будин

В подзадаче 1 ограничение на n маленькое, что позволяет написать перебор направления каждого робота за $O(2^n)$ и проверить за $O(n)$: покроют ли они в таком состоянии весь газон. Получаем решение за $O(2^n \cdot n)$.

В подзадаче 2 все роботы изначально направлены в правую сторону, следовательно, ответом будет 0, если все отрезки между ближайшими роботами будут покрыты. Иначе мы обязаны в каждом

таком непокрытом до конца отрезке поменять направление правого робота и проверить покрытие всего загона. Получаем решение за $O(n)$.

Заметим, что, после того как мы поменяли направления, не должно быть случая, когда два идущих подряд робота направлены в разные стороны. Поскольку отрезок между ними не будет покрыт. Таким образом, направления роботов должны иметь вид: $1, 1, ..., 1, 1, -1, -1, ..., -1, -1$.

В подзадаче 3 можно перебрать самого левого робота i, который в итоге будет направлен в левую сторону, и посчитать (количество $d_j = -1$ для $j < i$) + (количество $d_j = 1$ для $i \leqslant j$). Также надо проверить, что после перенаправления каждый отрезок будет покрыт. Получаем решение за $O(n^2)$.

В подзадаче 4 и 5 для каждого робота заряда хватает для того, чтобы покрыть левый или правый отрезок. Идея такая же, как и в 3 подзадаче, только нужно оптимизировать подсчет поворотов. Это можно посчитать заранее линейными проходам с начала и с конца. Получаем решение за $O(n)$.

Для полного решения достаточно оптимизировать 3 подзадачу. Как говорилось выше, посчитать количество поворотов можно заранее линейными проходами. Это же можно применить для проверки покрытия префикса и суффикса. Получаем решение за $O(n)$.

Задача 6. Интерактивные переходы

Автор:	Валерий Родионов
Разработчики:	Валерий Родионов, Владимир Новиков

Перепишем задачу на язык графов: корпуса — это вершины графа, а переходы — неориентированные ребра. Если подстветка на корпусе выключена, то покрасим вершину в белый цвет, а если включена, то в черный.

Изначально все вершины и ребра покрашены в белый цвет (подстветка на всех корпусах и переходах выключена), а переключение подстветки на корпусе будет соответствовать перекраске вершины и последующей перекраске всех ребер, у которых концы имеют одинаковые цвета.

В подзадачах 1 и 2 ограничения на n и m маленькие, поэтому их можно сдать перебором различной степени эффективности. В подзадаче 2 предполагалось решение, которое строит граф на 2^{n+m} возможных раскрасках и ищет в нем путь из полностью белой раскраски в требуемую обходом в глубину или ширину за $O(2^{n+m}nm)$.

В подзадаче 3 все $c_i = 1$, то есть нужно покрасить все ребра в черный цвет. Заметим, что если у какого-то ребра оба конца покрашены в белый цвет, то и оно само будет покрашено в белый цвет, поэтому если существует такое ребро, то ответ точно «`NO`» (это условие является необходимым, поэтому во всех следующих подзадачах будем считать, что оно выполненоЮ иначе сразу выведем «`NO`»). В противном случае ответ «`YES`» и получить требуемую покраску можно с помощью следующего алгоритма: сначала покрасим все вершины в черный цвет (после этого все ребра станут черными), после чего перекрасим каждую вершину в требуемый цвет.

В подзадаче 4 граф представляет собой звезду. Снова заметим, что если у ребра цвета концов совпадают, а цвет ребра не совпадает с цветом концов, то ответ «`NO`». Покрасим каждое ребро по очереди в нужный цвет, перекрасив оба его конца в этот цвет, после чего перекрасим все вершины в цвета из итоговой покраски.

В подзадаче 5 все ребра удовлетворяют условиям $d_{a_i} = c_i$ и $a_i < b_i$, то есть если ориентировать все ребра в направлении увеличения номера вершины, то цвет каждого ребра будет совпадать с цветом левой вершины. Тогда работает следующий алгоритм: будем перебирать вершины в порядке увеличения номера, пусть сейчас мы перебираем вершину v. Тогда покрасим v и все вершины, в которые из v выходят ребра, в цвет d_v. Легко видеть, что после таких действий все исходящие из v ребра будут иметь правльный цвет и никогда его не изменят.

В подзадачах 6 и 7 граф представляет собой один простой путь. Легко видеть, что работает такой алгоритм: в порядке следования по пути покрасим каждое ребро в требуемый цвет, после чего в обратном порядке покрасим каждую вершину в требуемый цвет.

В подзадачах 8 и 9 граф представляет собой дерево. Посмотрим на любой лист, то есть вершину, с которой соединено ровно одно ребро. Если его требуемый цвет совпадает с требуемым цветом соединенного с ним ребра, то по рассуждению из предыдущей подзадачи мы можем покрасить это ребро в нужный цвет и оно навсегда его сохранит, поэтому можно удалить лист и соединенное

с ним ребро. Если требуемый цвет листа не совпадает с требуемым цветом соединенного с ним ребра, то цвет ребра совпадает с цветом другого конца. Тогда сначала мысленно удалим этот лист и покрасим оставшееся дерево, после чего вернем лист, покрасим его в противоположный требуемому цвет (теперь ребро, соединенное с этим листом, станет правильного цвета), после чего покрасим его в требуемый цвет.

В 10 подзадаче граф представляет собой цикл. Если на цикле есть ребро, цвета концов которого совпадают, то это ребро будет иметь правильный цвет независимо от наших действий, поэтому можно его удалить и свести задачу к пути. Значит цвета вершин на пути чередуются. Для вершину u обозначим через t_u номер последней операции, изменившей цвет вершины u. Пусть есть ребро uv, такое, что $d_u \neq d_v$ и $c_{uv} = d_u$. Тогда $t_u < t_v$, там как в противном случае ребро имело бы другой цвет. Тогда если цвета ребер на цикле чередуются, то мы получим, что для любой вершины u на цикле выполняется $t_u < t_u$, то есть требуемой последовательности операций не существует. В противном случае существует вершина, из которой выходит или входит два ребра и по рассуждению, похожему на использованное в подзадачах с деревом, эту вершину можно удалить и снова свести задачу к решению для пути.

Подзадача 11 сделана для неэффективных реализаций полного решения.

В подзадаче 12 дополнительных ограчений нет. Если есть ребро, у которого цвета концов совпадают, то оно либо покрашено в тот же цвет (в этом случае его можно удалить), либо в противоположный цвет (в этом случае можно сразу вывести «NO»). После этого ориентируем все ребра, как в 10 подзадаче. Мы уже доказали, что если в графе появится цикл, то ответ «NO». В противном случае у вершин графа сущесвует топологическая сортировка. Перенумеруем вершины в соответствии с этой сортировкой и получим, что покраска удовлетворяет всем условиям подзадачи 5, которую мы уже умеет решать.

Задача 7. Гонка дронов

Авторы:	Азат Сафиуллин, Иван Сафонов
Разработчики:	Иван Сафонов, Азат Сафиуллин

В первой подзадаче $n = 2$. Достаточно пройтись двумя указателями по двум массивам, двигая указатель того дрона, который пролетает следующее расстояние быстрее (учитывая индексы дрона). $O(m)$

Во второй подзадаче достаточно запустить гонку для каждого префикса k. Заводим k указателей и двигаем указатель того дрона, который проходит быстрее остальных (учитывая индексы). Получаем решение за $O(n^3 \cdot m)$.

Для решения третьей подзадачи достаточно оптимизировать поиск дрона, который пролетит следующий отрезок быстрее всех. Это можно сделать простым Set. $O(n^2 \cdot m \cdot log(n))$

В четвертой подзадаче расстояния между точками равны. Очевидно, что первым в гонке все расстояния подряд пролетит дрон с минимальной скоростью. Вторым пролетит дрон с минимальной скоростью из оставшихся и так далее. Таким образом, ответ можно вычислить формулой $\frac{(k-1)\cdot k\cdot m}{2}$, где k - текущий префикс.

В пятой подзадаче все скорости равны. Зафиксируем p — индекс левого вхождения максимального расстояния между ближайшими точками. Очевидно, что все отрезки левее p имеют длину меньше. Поэтому в гонке будет момент, когда все дроны соберутся в точке p. Поскольку никакой дрон не может пролететь отрезок максимальной длины быстрее, чем какой-то дрон, который находится левее p. Понятно, что, как только какой-то дрон пролетит максимальное расстояние, он будет лететь до конца непрерывно. А это эквивалентно решению подзадачи 4. То ответ для префикса k будет равен $(p-1)\cdot k\cdot(k-1) + \frac{(k-1)\cdot k\cdot(m-p+1)}{2}$.

Обратим внимание на рекорды слева направо в строго возрастающем порядке. Нетрудно понять, что при преодолении дроном какого-то рекорда, он же продолжит непрерывно лететь до следующего, поскольку до следующего рекорда будут встречаться отрезки не больше текущего рекорда. Таким образом, можно «сжать» количество отрезков, определив за каждым рекордом количество не больших отрезков за ним. Нетрудно доказать, что количество рекордов не больше $\sqrt{s_m}$.

В шестой подзадаче $n \leqslant 100$. Сожмем до рекордов. Запустим решение на подзадачу 3. Получаем

решение за $O(n^2 \cdot \sqrt{s_m} \cdot log(n))$.

В седьмой подзадаче не более 2 различных скоростей. Будем поддерживать ответ на префиксе. Когда добавляется новый дрон, достаточно добавить к ответу количество телепортаций, которое совершит этот дрон и количество вынужденных телепортаций других дронов, которые были созданы новым дроном. Это нетрудно посчитать, разобрав случаи с $t_i = 1$ или $t_i = 2$.

Пусть $d_1 = s_1$ и $d_i = s_i - s_{i-1}$ (для каждого $1 < i$)

Будем поддерживать ответ. Будем его обновлять после добавления нового дрона i.

Есть два случая:

1. когда i вынуждает телепортироваться других
2. когда другие вынуждают телепортироваться i

Переберем рекорд u и будем искать подходящее под условия дроны

1. $t_i \cdot d_m \geqslant t_j \cdot d_u$, т.е. ищем такие j, что i-й еще не финишировал (эквивалентно непрохождению i-го последнего рекорда).
2. $t_i \cdot d_u < t_j \cdot d_m$, т.е. ищем такие j, что еще не финишировали (эквивалентно непрохождению j-го последнего рекорда).

Такие j можно искать корневой декомпозицией с небольшой модификацией. Добавлений не больше $O(n)$, а количество запросов не больше $O(n \cdot \sqrt{s_m})$. Таким образом, на запрос добавления хотим потратить $\sqrt{t_m}$ запросов, а на запрос $O(1)$.

Данная идея с реализацией проходит 11 подзадач.

Для полного решения давайте отсортируем значения $t_{p_1} \leqslant t_{p_2} \leqslant \ldots \leqslant t_{p_n}$. Будем перебирать индексы в порядке возрастания значений t, то есть перебирать p_i. Тогда для всех $1 \leqslant u \leqslant m$ будем поддерживать указатели $ptr_1[u]$ — максимальное значение j, такое что $t_{p_i} \cdot d_m \geqslant t_{p_j} \cdot d_u$ и $ptr_2[u]$ — минимальное значение j, такое что $t_{p_i} \cdot d_u < t_{p_j} \cdot d_m$. Поскольку мы перебираем значения в порядке возрастания, мы можем двигать указатели.

Посмотрим как меняется ответ в момент времени p_i. В неравенствах, написанных выше участвуют все индексы p_j по $j \leqslant ptr_1[u]$ или $j \geqslant ptr_2[u]$ по всем $1 \leqslant u \leqslant m$. Но среди них нужно оставить только $p_j < p_i$. Давайте сделаем массив длины n, где в индексе x будем хранить добавку, которую дает элемент с индексом x. В этой структуре данных нужно $O(n\sqrt{s_m})$ прибавлять к индексу (при движениях указателей) и n раз считать сумму на префиксе. Тогда если применить для этого массива корневую декомпозицию, получится полное решение.

Мы нигде не использовали, что t_i маленькие. Время работы $O(n(\sqrt{s_m} + \sqrt{n}))$, память $O(n)$.

Задача 8. За связь без перебоев

Автор:	Никита Лазарев
Разработчик:	Алексей Михненко

Первые две подгруппы можно сдать, просимулировав описанный в задаче процесс.

Для решение за квадрат переберём антенну, которая будет заменена на запасную и найдём нестойкость покрытия для каждого случая независимо. Если антенны зафиксированы, можно для каждого i за $\mathcal{O}(n)$ найти величину $nxt(i)$, равную максимальному значению r, что существует антенна, покрывающая город i, а также город $r - 1$.

Построим лес (набор деревьев), где предком i-й вершины будет вершина $nxt(i)$. Если $nxt(i) = n$, то вершина i будет корнем какого-то дерева. Пусть $sz(i)$ — размер поддерева вершины i, если каждое дерево подвешено за вершину с максимальным номером. Тогда для фиксированного города $k > 1$ количество пар $s < t$, что при перемещении от города s в город t произойдёт переподключение при перемещении от $k - 1$ в k, равно $(sz(k) - 1) \cdot (n - k)$. Таким образом, ответ на задачу равен:

$$\sum_{k=1}^{n} (sz(k) - 1) \cdot (n - k)$$

Таким образом, задачу можно решить за $\mathcal{O}(n^2)$.

В четвёртой группе можно заметить, что антенны не выгодно заменять на антенну x, поэтому задача решается за $\mathcal{O}(n)$.

В пятой группе все $a_i = 0$, поэтому можно перебрать позицию, в которую мы поставим антенну, после чего нестойкость покрытия вычисляется по простой формуле, аналогичной решению, приведённому ранее.

Для решения на полный балл посмотрим, как именно изменяется структура леса при добавлении антенны мощности x в городе i. Нетрудно видеть, что какой-то отрезок вершин (по номерам) переподвешивается к вершине $i + x + 1$. Обозначим этот отрезок, как $[l_i, r_i]$. Легко убедиться, что $l_i \leqslant l_{i+1}$ и $r_i \leqslant r_{i+1}$, поэтому такие отрезки можно находить двумя указателями.

Будем перебирать i в порядке возрастания и поддерживать текущий отрезок $[l_i, r_i]$, а так же сумму значений $(sz(k) - 1) \cdot (n - k)$ с учётом того, что у вершин с номерами от l_i до r_i ребро в корень удалено. Чтобы поддерживать такую сумму, необходимо при увеличении правой границы на 1 удалять ребро из $r + 1$ в предка, а при увеличении левой границы — добавлять ребро из l_i в исходного предка вершины l_i обратно.

При увеличении правой границы заметим, что все рёбра на пути от r до корня ещё не удалены, поэтому, чтобы посчитать, на сколько изменится нестойкость, необходимо посчитать произведение размера текущего поддерева вершины r и суммы значений $(n - v)$ по всем предкам v вершины r. Такую сумму можно посчитать заранее за $\mathcal{O}(n)$. Чтобы посчитать размер поддерева, переберём всех непосредственных детей вершины r в исходном лесе и, если ребро в ребёнка ещё не удалено, нетрудно видеть, что все рёбра в поддереве ребёнка тоже ещё не удалены, поэтому достаточно просуммировать размеры поддеревьев таких детей.

При увеличении левой границы на 1 достаточно рассмотреть два случая.

- Если предок вершины p в исходном лесе сейчас является корнем, то добавления добавление ребра в этого предка увеличит ответ на $sz(l) \cdot (n - p)$, так как, аналогично увеличению правой границы, легко заметить, что все рёбра в поддереве вершины l точно не удалены
- Если предок вершины l на данный момент не является корнем, то все рёбра от p до корня исходного дерева, в котором содержался p, не удалены, поэтому ответ можно пересчитать, используя сумму, насчитанную для увеличения правой границы.

Кроме всего вышеперечисленного, надо поддерживать суммарный размер всех поддеревьев, у которых ребро в предка удалено. Это можно поддерживать аналогично вышеперечисленному при увеличении границ отрезка. Из-за добавления новой антенны все такие поддеревья подвесятся к вершине $i + x + 1$. Используя суммарный размер поддеревьев, посчитать реальное значение нестойкости можно аналогично увеличению левой границы.

Таким образом, получили решение за $\mathcal{O}(n)$.