Гонка дронов

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

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

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

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 $\frac{(k-1)km}{2}$, 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 $(p-1)k(k-1) + \frac{(k-1)k(m-p+1)}{2}$.

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 в рекорды. Запустим решение подзадачи 3. Получаем решение за $O(n^2\sqrt{s_m}\log(n))$.

В седьмой подзадаче $t_i \leqslant 2$. 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 $t_i = 1$ или $t_i = 2$.

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

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

Есть два случая:

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

Переберем рекорд u и будем искать подходящие 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-м 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-м последнего рекорда).

Такие j можно искать корневой декомпозицией. Добавлений не больше $O(n)$, а количество запросов не больше $O(n \cdot \sqrt{s_m})$. Таким образом, добавление хотим выполнить за $O(\sqrt{t_m})$, а запрос $O(1)$. Получаем решение за $O(n(\sqrt{s_m} + \sqrt{maxv}))$.

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 $O(n)$.