Олимпиада школьников по программированию «ТехноКубок» 2021

Первый отборочный тур

Разбор задач

A. В поисках Саске

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Заметим, что подходит следующий массив: $-a_2, a_1, -a_4, a_3, ..., -a_n, a_{n-1}$.

B. Новая техника

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Для решения этой задачи достаточно для каждой строки определить на каком месте в исходной таблице она находится. Если мы рассмотрим первый символ в строке и найдём в каком столбце и на каком месте он находится, то это место будет как раз будет задавать положение нашей строки в исходной таблице. Так как все символы исходной таблицы различны, то положение каждой строки однозначно восстановится.

C. Простота исполнения

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Рассмотрим все возможные значения ладов, которые могут использоваться, для этого выпишем массив пар $(b_j - a_i, j)$, для всех возможных i, j, и посортируем его. Тогда в данном массиве надо найти подотрезок, с минимальной разницей первых аргументов, и при этом содержащий все возможные варианты второго аргумента (так как это будет означать, что для каждой ноты мы можем выбрать некоторую струну, чтобы лад попал в нужный отрезок).

Для этого посчитаем массив $right[l]$, в котором для всех возможных левых концов будет содержаться минимальный правый край, такой что на подотрезке $[l, right(l)]$ встречаются все возможные значения второго аргумента. Нетрудно заметить, что $right[l] \le right[l+1]$, так как если на отрезке $[l+1, right[l+1]]$ содержатся все возможные варианты второго аргумента, то и на отрезке $[l, right[l+1]]$ это тоже будет верно. Тогда чтобы искать все значения $right[l]$, можно двигать два указателя, и в каждый момент времени поддерживать множество нот, которые встречаются на текущем отрезке.

Когда мы нашли массив $right$, нам надо просто взять минимум разности первых аргументов, по всем возможным подотрезкам $[l, right[l]]$. Итоговая асимптотика $O(nm \log(nm))$.

D. Сюрикены

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Заметим, что если в данный момент покупается сюрикен стоимости x, то единственная информация, которую мы должны запомнить про оставшиеся сюрикены, что их стоимости $\geq x$. Также понятно, что если мы рассматриваем в какой-либо момент времени сюрикены, оставшиеся на витрине, то чем раньше сюрикен был положен на витрину, тем более сильное ограничение на него действует. Теперь рассмотрим какие варианты событий могут происходить при покупке сюрикена стоимости x. Если про все сюрикены на витрине мы знаем, что их стоимость $> x$, тогда ответ на задачу отрицательный. Иначе у всех сюрикенов, у которых ограничение меньше чем x, мы должны поднять ограничение до x, и удалить любой из них, так как никакие другие сюрикены мы удалить не можем, а все эти сюрикены неразличимы между собой. Но так как мы знаем, что на самый последний сюрикен положенный на витрину действует наиболее слабое ограничение, то можно ничего не поддерживать и просто каждый раз удалять его, а после проверить, что такая последовательность действий нам подходит. Проверку для конкретной последовательности можно делать используя любую кучу на минимум. Итоговая асимптотика будет $O(n \cdot \log n)$.

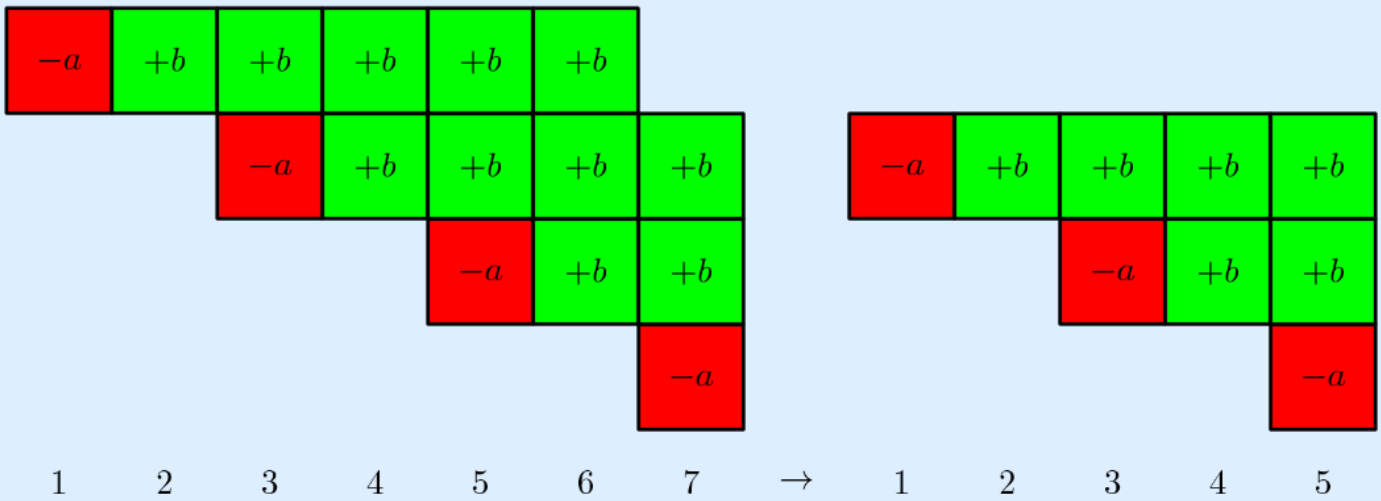
E. Oracle соло мид

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Будем использовать схематичные изображения всего происходящего в таком виде, что каждое изменение здоровья будет отображено в виде клетки, каждая строка будет обозначать одно применение заклинания, и каждый столбец будет обозначать секунду, в которую что-то происходит.

Во-первых, если $a > b \cdot c$, то ответ -1. В самом деле, после t секунд суммарный нанесённый урон равен $(a - bc)$ для всех заклинаний, действие которых уже прошло, а также ещё какой-то урон от заклинаний, которые ещё действуют. Первое слагаемое может быть сколь угодно большим, а второе ограничено снизу, например, числом $-bc^2$, поскольку максимум c заклинаний ещё не истекли, и каждое из них вылечило врага максимум на b единиц здоровья в секунду на протяжении максимум c секунд. Таким образом, нанесённый урон может быть сколь угодно велик.

С другой стороны, если $a \leq bc$, то ответ всегда существует. В самом деле, имеет смысл смотреть только на моменты времени, делящиеся на d, т. е. в точности на моменты, когда мы наносили урон. Очевидно, что для любого другого t в момент времени t у врага было не меньше здоровья, чем за секунду до этого. Также если $t \geq c$, то у врага не меньше здоровья, чем в момент времени $t - d$. В самом деле, разница в этих уровнях здоровья лишь в одном заклинании, которое целиком истекло, а это неотрицательное число:



Теперь, когда мы знаем, что нужно рассматривать лишь $t < c$, мы точно знаем, что ответ существует. Поймём теперь, когда в общем случае следует вычитать d из t, чтобы уменьшить здоровье врага. Из таких же соображений следует, что если $t < c$, то при уменьшении t на d здоровье врага увеличивается на $a - tb$, а поскольку $t = dk$ для некоторого целого k, мы увеличиваем здоровье на $a - bdk$. Очевидно, это следует делать, пока $a - bdk < 0$:

Иными словами, задача звучит теперь следующим образом: найти наибольшее такое k, что $a \geq bdk$, и применить заклинание $(k + 1)$ раз. У врага будет наименьший уровень здоровья как раз как только мы применим заклинание в $(k + 1)$ -й раз. Ответ, таким образом, равен $a(k + 1) - \frac{k(k+1)}{2}bd$. Время работы составляет $O(1)$ на тест.

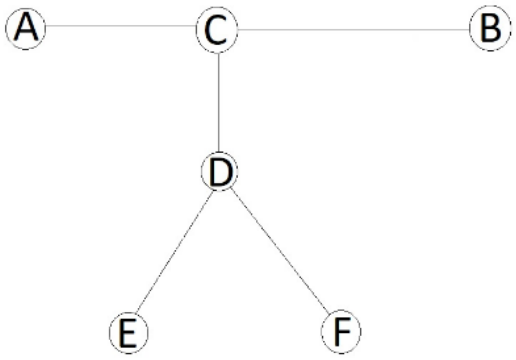
Также можно было заметить, что здоровье врага — выпуклая функция от времени, и найти ответ тернарным поиском. Это занимает $O(\log maxanswer)$ времени на тест, что всё ещё достаточно быстро.

F. Дороги и рамен

ограничение по времени на тест: 5 секунд
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Один из оптимальных путей всегда начинается в одном из концов диаметра дерева.

Доказательство:



Пусть на рисунке выше, AB — это диаметр дерева, а оптимальный ответ — это EF. Тогда чётность количества каменных дорог на DE и DF одинаковая, также отсюда чётность количества каменных дорог на CE и CF тоже одинаковая. Так как диаметр — это самый длинный путь в дереве, то чётность количества каменных дорог на AC и BC различна (иначе диаметр был бы оптимальным ответом). Значит чётность количества каменных дорог на CE совпадает с чётностью на AC или на BC. Пусть, не теряя общности, это будет путь AC. Тогда на пути AE чётное количество каменных дорог. Заметим, что так как AB — это диаметр, то AC не короче, чем CF, отсюда AD не короче, чем DF, и наконец отсюда AE не короче, чем EF. А следовательно существует оптимальный путь, начинающийся в одном из концов диаметра.

Осталось научиться решать задачу, когда один из концов пути зафиксирован. Если подвесить дерево за эту вершину, и для каждой вершины выписать чётность количества дорог до корня, то тогда изменение веса одного ребра меняет все чётности в некотором поддереве. Если занумеровать все вершины дерева в порядке обхода dfs, то тогда любое поддерево является непрерывным отрезком в такой нумерации. А значит наша задача свелась к следующей: «Изначально есть массив из нулей и единиц, приходят запросы заменить все значения на подотрезке на противоположные, и после каждого запроса надо найти позицию, в которой записан ноль, и при этом у неё самый большой параметр глубины». Эту задачу можно решать с помощью дерева отрезков, поддерживая в вершине ДО самую глубокую вершину, в которой написан ноль, и самую глубокую вершину, в которой написана единица. Итоговая асимптотика $O(n \log n)$.