Разбор задачи «Распродажа!»

Научимся решать для фиксированного q. Для этого отсортируем книги. Переберем сколько раз мы делаем заказ — пусть это число x. Поймем, что выгодной стратегией является покупка $q + w$ самых дорогих еще не купленных книг $x - 1$ раз. Оставшиеся книги мы купим за 1 раз.

Теперь переберем Q. Переберем количество заказов, которые мы совершим пусть это x. Заметим, что x от 1 до $\frac{n}{q+w} + 1$.

Заметим, что книги мы всегда покупаем на отрезке в отсортированном порядке. Действительно, пусть мы купили k, по нашей жадности мы хотим купить следующим шагом еще $q+w$ самый дорогих не купленных книг. Все эти книги следуют сразу после k-ой книги и лежат на отрезке длины $q + w$.

Теперь посчитаем префиксные суммы и за $O(\frac{n}{q+w})$ решим задачу для фиксированного q. Известно, что сумма $O(\frac{n}{1} + \frac{n}{2} + ... + \frac{n}{n}) = O(n \cdot log(n))$.

Разбор задачи «Доктор Стрэндж и выставка»

Посчитаем динамическое программирование «по битовым маскам».

$dp[i][m]$ — каким наименьшим количеством чисел от 1 до i можно получить битовую маску m, переходы следующие:

$dp[i+1][m \texttt{ AND } a_{i+1}] = \min(dp[i+1][m \texttt{ AND } a_{i+1}], dp[i][m]+1)$ — случай, когда мы берем число a_{i+1}

$dp[i+1][m] = \min(dp[i+1][m], dp[i][m])$ — случай, когда мы не берем число a_{i+1}.

База динамического программирования:

$dp[0][2^{12} - 1] = 0$, остальные значения равны ∞ (некому числу, заведому большему ответа, например, $n + 1$).

После чего, нужно заметить, что количество памяти, требуемое для массива dp требует $n \cdot 2^{12}$ ячеек памяти, что занимает около 320мб памяти, если использовать тип int.

Для того, чтобы избежать этой проблемы, нужно либо проводить все вычисления в 2-байтовом типе данных ($short\ int$ в C++, либо $char$ в Java).

Либо убрать из массива a повторяющиеся элементы, тогда n не будет превосходить 2^{12}.

Либо использовать стандартный подход: заметить, что после подсчета i-го слоя массива dp, к предыдущим слоям мы уже обращаться не будем и на подсчет dp они никак не повлиют, поэтому можно хранить только текущий слой, то есть массив $dp[0 \dots 2^{12} - 1]$, и значения динамического программирования для следующего слоя сохранять в нем же.

После подсчета динамики нужно проверить, что $dp[0] \leqslant k$.

Сложность этого решения составляет $n \cdot 2^{12}$.

Разбор задачи «Граненые стаканы»

Нужно найти суммарную площадь многоугольников, и поделить объем на нее.

Площадь многоугольника можно найти с помощью векторных произведений.

Разбор задачи «Преследование»

Первое решение:

Будем считать динамику dp_{ijt} — количество разбиений отрезка числа $[0, j]$, если последнее выбранное число в нем — $[i, j]$, и уже выбрано t чисел.

Переход: находясь в текущем состоянии (i, j, t), переберем новое число в разбиении — переберем p от $j+1$ до $|x|$ и попробуем взять число $x[j+1..p]$, если абсолютная разность с предыдущим попадает в отрезок $[l, r]$.

Итоговая асимптотика решения: $O(|x|^4)$, но длина числа x меньше 20, поэтому это легко укладывается по времени даже со 100 мультитестами.

Второе решение:

Давайте заметим, что всего возможных разбиений C_{18}^9, что меньше 50000. Поэтому достаточно просто перебрать все возможные разбиения и проверить их на корректность.

Разбор задачи «Библиотека»

Первоначально преобразуем данные о каждой книги в отрезки дней, в каждый из которых книга уже может быть сдана, то есть отрезки $books$ вида $[s_i + c_i, f_i]$ — от момента прочтения книги до дня, по прошествии которого книга должна быть сдана. Затем отсортируем все отрезки по левому концу.

Задача решается жадностью: в каждый свободный день выбираем из всех книг, которые к этому дню уже прочитаны, книгу, которая должна быть сдана раньше остальных. Затем переходим к следующему дню и проделываем то же самое.

Реализовывать будем следующим образом. Каждый правый конец отрезка по мере возможности сдачи соответствующей ему книги в текущий день будем добавлять в множество. Текущий день храним в переменной j, еще не рассмотренный отрезок - в переменной i. Первоначально j соответствует минимальному левому концу среди левых концов всех отрезков. На каждой итерации цикла сначала добавляем все доступные для сдачи в текущий день j книги (увеличивая при этом переменную i), а затем вынимаем из множества минимальный правый конец среди всех правых концов доступных книг. И если вынутый правый конец оказался меньше j, то есть текущего свободного дня, выводим `NO` и завершаем программу.

Иначе идем дальше. Если после извлечения элемента множество оказалось пусто, присваиваем переменной j значение левого конца еще не рассмотренного отрезка, то есть $j = books[i].left$. В ином случае, если множество не оказалось пустым, увеличиваем j на один. Затем переходим к следующей итерации.

После цикла выводим `YES`.

Разбор задачи «Магические сферы»

Будем по 1 вершине добавлять в граф в порядке от 1 до n. Для вершины i рассмотрим все ребра, которые идут в вершины с номерами от 1 до $i-1$. Жадно покрасим вершину i, в тот цвет, который дает наименьшую сумму равных пар. Для каждой вершины мы получим как минимум $\frac{S[i]}{2}$, где $S[i]$ сумма по всем ребрам, которые выходят из вершины i.

Разбор задачи «Знания — сила»

Для начала заметим, что носители размножаются независимо друг от друга. А значит, достаточно почитать ответ ans, если изначально был только один носитель, тогда итоговое количество — $ans \times n$. Все дальнейшее рассуждение предполагает, что изначально есть только один носитель.

Пусть A_i — количество носителей, появившихся за i-ый дней. То есть

$A_1 = 2$, $A_2 = 5$, $A_3 = 13$ и так далее.

Тогда верно равенство

$A_n = A_{n-1} + \sum_{i=1}^{n-2} A_i$, где A_{n-1} — количество носителей, появившихся за $n-1$ день, $\sum_{i=1}^{n-2} A_i$ — количество новых носителей.

Заметим, что $F_{2k} = A_k$. Докажем данное утверждение по индукции.

Предположим, что для 1 lei lek равенство верно. Тогда докажем равенство для $k+1$. То есть мы хотим доказать, что

$F_{2(k+1)} = A_{k+1}$, тогда

$A_{k+1} = A_k + \sum_{i=1}^{k-1} A_i = F_{2k} + F_{2k-1}$

Заметим, что $A_k = F_{2k}$ по индукционному предположению. Тогда должно соблюдаться, что

$\sum_{i=1}^{k-1} A_i = F_{2k-1}$

Вновь распишем левую и правую части равенства, получим

$\sum_{i=1}^{k-1} A_i = A_{k-1} + \sum_{i=1}^{k-2} A_i = F_{2k-2} + F_{2k-3}$

Вновь заметим, что по индукционному предположению

$A_{k-1} = F_{2k-2}$. Тогда повторим процесс заново для оставшейся суммы, каждый раз индексы будут уменьшаться, и одно из слагаемых и в правой и в левой части будут сокращаться. В итоге придем к равенству

$A_1 = F_2$ — очевидно верно.

Разбор задачи «Путешествие сквозь миры»

Различных цифр, из которых могут состоять номера, интересные для Стрэнджа, 9 штук. Переберем и зафиксируем цифру d, а затем посмотрим на все подходящие числа, состоящие только из

этой цифры. Для этого будем последовательно рассматривать числа $\overline{d}$, $\overline{dd}$, ... (здесь под $\overline{x_1x_2x_3\dots x_n}$ подразумевается десятичная запись числа $x_1x_2\dots x_n$). Как только первое из них станет не меньше l, мы попали в отрезок $[l,r]$ и можно начинать обновлять ответ. Как только число стало больше r, дальше можно не продолжать.

Также нужно было быть аккуратным с переполнениями — при неаккуратной реализации могло не хватить даже 64-битного типа данных.

Разбор задачи «Карточный трюк»

Рассмотрим простую эмуляцию того, что написано в программе. Такая программа может не уложиться в ограничение по времени.

Заметил, что некоторое количество раз подряд прямоугольник, образованный пересечением карт, будет одинаковый. Пусть $a \leqslant b$ и $c \leqslant d$, тогда до тех пор, пока $b \geqslant c$, $b \geqslant a$, $d \geqslant a$ и $d \geqslant c$, будут сохранятся равенства, что $a \leqslant b$ и $c \leqslant d$, поэтому ориентация прямоугольников меняться не будет, а пересечением прямоугольников будет являться прямоугольник размера $c \times a$. Можно вычислить количество раз, которое можно отрезать такой прямоугольник, чтобы выполнялись все 4 неравенства, это и нужно сделать. После этого изменить ориентацию прямоугольников. И повторять это, пока они не исчезнут.

Заметим, что этого также недостаточно. Например, если одна из карт имеет почти квадратную форму, а другая — очень узкая и длинная, то после отрезания каждого прямоугольника, первая карта будет переворачиваться. Поэтому нужно сделать проверку, что наступил такой случай. Заметим, что в таком случае первая карта действительно будет переворачиваться после каждого отрезания. Поэтому, уменьшение длины первой стороны и второй стороны этой карты будут чередоваться. Этот случай можно обработать бинпоиском, в котором найти максимальное количество отрезаний прямоугольника, что длинная сторона второй карты все еще длиннее короткой. Чтобы найти длину, на которую уменьшится длинная сторона второй карты, нужно найти сумму двух арифметических прогрессий. Первая — сумма длин одной стороны первой карты, вторая — второй.

Разбор задачи «Две карты»

Будем поддерживать ответ. При добавлении отрезка нужно добавить к ответу количество отрезков, которые дают длину s в объединении с новым, при удалении — отнять.

Как посчитать, сколько отрезков $[l_i;r_i]$ в объединении с $[L;R]$ дают длину s: рассмотрим несколько случаев.

1. Если $R-L>s$, то ответ 0

2. Эти два отрезка не имеют общих точек, $l_i > R$ или $r_i < L$. Тогда длина второго отрезка равна $len_i = s-(R-L)$, а l_i либо больше R, либо меньше $L-len_i$. Чтобы быстро находить количество таких отрезков, будем для каждой длины поддерживать структуру, в которой будем хранить левые концы отрезков, чтобы быстро искать их количество. Это может быть декартово дерево, дерево отрезков, дерево Фенвика (последние два — разреженные).

3. Эти два отрезка пересекаются, но не являются вложенными. Тогда либо $l_i < L$, $L \leqslant r_i < R$, либо $r_i > R$, $L < l_i \leqslant R$. В первом случае объединение отрезков - это $[l_i;R]$, поэтому $R-l_i=s$. Это означает, что l_i фиксирован, а r_i находится в некотором интервале. Для этого для каждого левого конца отрезка будем поддерживать структуру, в которой будем хранить правые концы отрезков, чтобы быстро искать их количество. Во втором случае аналогично — нужно будет завести такую структуру для каждого правого конца.

4. Отрезки вложенные, длина $[l_i;r_i]$ больше либо равна длине $[L;R]$. Это означает, что $r_i-l_i=s$, $R-s \leqslant l_i \leqslant L$. Найдём это количество при помощи структуры из второго пункта.

5. Отрезки вложенные, длина $[l_i;r_i]$ меньше длины $[L;R]$. Это может быть только в том случае, когда $R-L=s$. Требуется посчитать, сколько есть отрезков таких, что $L<l_i<r_i<R$. Для каждого L будем поддерживать количество отрезков строго внутри $[L;L+s]$. Отрезок $[l_i;r_i]$ лежит строго внутри отрезка $[L;L+s]$, если $r_i-s+1 \leqslant L \leqslant l_i-1$. Поэтоэму при добавлении

отрезка $[l_i; r_i]$ прибавим единицу на отрезке $[r_i - s + 1; l_i - 1]$, при удалении — вычтем. Это можно делать при помощи дерева отрезков.

Время работы решения: $O(n \cdot logn)$.