Председатель жюри: Овчаров Иван
Состав жюри: Туров Макар
Мизёв Андрей
Базилевич Григорий

Спонсоры олимпиады: ЛКЛ
PARMA Technologies Group

Отдельная благодарность: Шульгина Галина Михайловна

Задача A. Интересные тройки

Автор и разработчик: Туров Макар

Так как $a \leqslant c; b \leqslant c$, то $a + b \leqslant 2c$. Так как $a + b \vdots c$, то $a + b = c$ или $a + b = 2c$.
Если $a + b = 2c$, то $a = b = c$.

Если $a + b = c$, то $b = c - a$, т.е. $a + c \vdots c - a$, тогда $2a \vdots b$, тогда $a = b 2a = b$.
Тогда, при фиксированном c ;a и b могут быть получены так:

1. $a = b = c$
2. Если $c \vdots 2$, то $a = b = \frac{c}{2}$
3. Если $c \vdots 3$, то $6a = 3b = c$

То есть, количество искомых троек равно сумме количеств $c \leqslant k; c \leqslant k$ и $c \vdots 2; c \leqslant k$ и $c \vdots 3$, то есть ответ равен $k + [\frac{k}{2}] + [\frac{k}{3}]$ ([x] это x, округлённое вниз)

Задача B. Сижу за решёткой в темнице сырой

Автор: Мизев Андрей
Разработчик: Овчаров Иван

Давайте заметим что в графе — квадратной таблице с n вершинами будет сторона из $s = \sqrt{n}$ вершин. Кроме того, у неё будет 4 угловых (степень 2)вершины, $4(n - 2)$ крайних (степень 3) и остальные серединные (степень 4). Из угловой клетки посчитаем расстояния до остальных с помощью bfs. Расстояния до остальных угловых будут $s-1; s-1; 2(s-1)$. Кроме того, если в bfs посчитать массив предков — для каждой вершины сохранить из какой мы её обновили, то можно будет выйти из угловых по этому массиву и найти крайние вершины, соединяющие их и первую угловую. Тогда, по ним можно восстановить какие вершины стоят в решётке на i;j месте — это вершина, которая смежна с вершинами i-1;j и i;j-1, но не является вершиной i-1;j-1.

Выполнив двнный алгоритм и проверяя на каждом шаге не поломалось ли что-то, мы получим полное решение данной задачи, работающее за O(n+m)

Задача C. Модный массив

Автор и разработчик: Туров Макар

Давайте придумаем конструктивное решение и докажем, что оно оптимально. В $k-1$ множество будем брать уникальные элементы, которые остались в массиве a, а в последнее множество все оставшиеся элементы. Посчитаем моду в каждом множестве и получим ответ.

Докажем, что ответ оптимальный. Пусть $cnt_{a_i} \leqslant k-1$, тогда в ответе будут учтены все вхождения числа a_i, если же $cnt_{a_i} \geqslant k$, то мы можем учесть это число в ответ не более k раз, причём $k - 1$ Раз оно точно было учтено. Число a_i может быть модой во всех k множествах, только если a_i является модой массива a. Так как оно должно модой во всех k множествах. Наш алгоритм как раз

добавляет в последнем множестве числа, которые являются модой в изначальном массиве. А значит достигается максимальный ответ. Что и требовалось доказать.

Задача D. Реформы в тридевятом царстве

Автор и разработчик: Овчаров Иван

Подгруппа 1.

Заметим, что в данной подгруппе количество городов довольно небольшое и за $O(2^n)$ можно перебрать все подмножества, тратя ещё $O(n)$ на проверку случая, найдя ответ для всех вершин сразу. Итоговая асимптотика решения $O(n * 2^n)$

Подгруппа 3.

Заметим что в данной подгруппе нет отрицательных значений, значит нам ничего не мешает взять все города в государство. Значит необходимо просто вывести сумму всех чисел. Работать будет за $O(n)$.

Полное решение.

Подвесим дерево за вершину 0. При помощи dfs посчитаем для каждой вершины следующее значение $p_v = \max(0, a_v + \sum p_e)$, где $\sum p_e$ — это сумма значений по всем потомкам вершины. Таким образом p_v — наибольшее значение которое можно получить используя данное поддерево.

Теперь, запустив dfs ещё раз, найдем ответ для каждой вершины. $ans_v = s + a_v + \sum p_e$, где s — максимальная сумма для всего дерева кроме данного поддерева. Это значение будем передавать от предка потомкам следующим образом: $s_e = \max(0, s_v + \sum(p_e) - p_e)$. Получили решение работающее за 2 прохода dfs. T.e. $O(n)$

Задача E. Загадка двери

Автор и разработчик: Туров Макар

Изначально заметим, что такая загадка существует тогда и только тогда, когда $\sum_{i=0}^{n} a_i * (1 - i) = 1$. Так как каждая комната с количеством ведущих дверей j уменьшает количество свободных комнат на $j - 1$, а изначально свободных a_0. Дадим каждой комнате характеристику глубина. Глубина – это количество дверей, которые необходимо пройти для достижения данной комнаты из первой. А также характеристику позиция. Позиция комнаты – это количество комнат на той же глубине, которые находятся левее данной комнаты. Тогда для какого-то разбиения комнат – это максимальная глубина. Заметим, что для комнат u и v, если глубина u больше глубины v и количество дверей в u больше количества дверей v, то мы можем поменять их местами, обменяв комнаты в которые они ведут(комната u получит все комнаты v, и доберёт оставшиеся среди тех, к которым была подсоединена до этого, а комната v получит оставшиеся комнаты u до обмена), при этом не ухудшив ответ. То есть максимальная глубина не увеличится. Также заметим, что мы можем внутри одной глубины менять позиции комнат без изменения ответа. Ещё для комнат на одной глубине мы можем менять местами комнаты, в которые ведут их двери без изменения ответа. Теперь заметим, что мы можем из любого изначального распределения комнат, привести к виду, в котором на каждом слое комнаты идут по неубыванию позиций и глубин, до которых из них можно добраться, а также для соседних глубин, верно, что комната с максимальным количеством дверей у меньшей глубины, имеет их не больше, чем комната с минимальным количеством дверей у большего слоя. И при приводе к этой позиции ответ не ухудшится, а значит эта позиция и имеет оптимальный ответ. Тогда алгоритм нашего решения такой: будем идти по возрастанию i в массиве a_i. Набирать комнаты с минимальной глубиной, если считать не с первой комнаты, а с данной. И обновлять глубину рассматриваемой как максимум из глубин комнат, в которые она ведёт + 1. Для преодоления TLa, необходимо поддерживать 2 пары вида (глубина, количество комнат), так как всегда будет не больше 2 различных глубин.

Задача F. Трудолюбивый молодой человек Вася

Автор и разработчик: Мизёв Андрей

Если поперебирать случаи, можно заметить что цикл в последовательности g не превосходит 6n.

Данный факт является свойством цикла Пизано (погуглите). Тогда, можно написать программу, которая находит цикл, подсчитывает сумму на нём, затем домножит это значение на количество полных повторений данного цикла — $[\frac{m}{l}]$, где l — длина цикла и досчитает остаток цикла. Такая программа работает за $O(n)$