Задача A. Граф

Автор задачи: Илья Збань
Автор разбора: Илья Збань

В этой задаче был дан граф, и нужно было узнать, сколько есть пар вершин, не соединенных ребром, таких, что если провести между ними ребро, в графе появится ровно один новый простой цикл.

Решение: Необходимо оставить в графе все мосты, и для любой компоненты связности из k вершин добавить к ответу $\frac{(k-1)(k-2)}{2}$.

Докажем это: если есть путь между вершинами u и v, проходящий через ребро, не являющееся мостом, то его можно удалить, и путь останется. То есть, при добавлении ребра uv в графе появится больше одного простого цикла. Обратно: если в графе есть лишь мосты, то подходит любая пара вершин, не соединенные ребром.

Задача B. Любимая строка

Автор задачи: Евгений Замятин
Автор разбора: Евгений Замятин

В задаче требовалось для строки S найти перестановку строк t_i такую, что последовательно склеив в этой перестановке все строки мы получим S. Длины всех строк t_i равны.

Решение: Посчитаем для каждой строки t_i ее полиномиальный хэш. Разобьем строку S на множество строк s_i. Для каждой строки s_i тоже посчитаем хэш. Теперь осталось упорядочить строки t_i, чтобы хэши соответствовали. Также эта задача легко решается с помощью структуры данных "Бор".

Задача C. Домашнее задание

Автор задачи: Илья Збань
Автор разбора: Илья Збань

В этой задаче было дано дерево, и нужно было посчитать некоторую сумму.

Решение: Во-первых, посчитаем отдельно сумму «$max \cdot len$» и «$-min \cdot len$».

Будем решать методом «разделяй и властвуй на дереве»: найдем в дереве центроид (вершина, в любом поддереве которой меньше половины вершин дерева), посчитаем данную величину для всех путей, которые проходят через этот центроид, и запустимся рекурсивно для его поддеревьев.

Пусть мы нашли центроид — вершину v (сделать это можно обходом в глубину), и подвесили дерево за него. Посчитаем для всех остальных вершин максимум на пути от v до них, и отсортируем полученный массив. Теперь запустим еще один dfs и будем считать данную величину, суммируя по парам вершин из разных поддеревьев. Решение за квадрат: переберем пару вершин, и если максимум на пути до второй меньше, чем до первой, прибавим длину пути, умноженную на максимум до первой, иначе — длину пути на максимум до второй. Используя построенный массив, и перебрав вторую вершину, первую можно отдельно перебрать, взяв сумму длин и максимумов фенвиков до и после позиции второй вершины в этом массиве. Сложность — $O(n \log^2 n)$.

Задача D. Ученье — свет, неученье — тьма

Автор задачи: Дмитрий Филиппов
Автор разбора: Дмитрий Филиппов
Пример кода на C++ (0.8 с, 224 Мб): http://pastebin.com/JH72SR2R
Пример кода на Java (1.3 с, 326 Мб): http://pastebin.com/1NDyKNgE

В этой задаче был дан массив, над которым производились операции двух типов: присвоить элементу массива другое значение и посчитать количество делителей у произведения чисел на отрезке.

Решение: Для начала поймем, что такое количество делителей у произведения чисел на отрезке. Пусть $P = a_l \cdot a_{l+1} \cdot \ldots \cdot a_r$. Тогда если мы представим P в виде $P = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$, где $p_1 < p_2 < \ldots < p_k$ — простые числа, а $\alpha_1, \alpha_2, \ldots \alpha_k$ — степени их вхождения, то нетрудно заметить, что количество делителей P будет равно $(\alpha_1 + 1) \cdot (\alpha_2 + 1) \cdot \ldots \cdot (\alpha_k + 1)$.

Осталось научиться считать это значение. Заметим, что числа в массиве у нас маленькие, а всего простых чисел может быть еще меньше (меньше 1300). Тогда будем просто хранить 1300 деревьев Фенвика, чтобы уметь быстро отвечать на запрос «степень вхождения простого числа p на отрезке $[l, r]$)».

Приравнивать элемент массива другому значению мы теперь умеем за $O(count \cdot \log n)$, где $count$ — количество различных простых чисел в разложении нового числа, которому мы хотим приравнять элемент массива. Понятно, что $count = O(1)$, потому что числа у нас маленькие.

Отвечать на запрос мы умеем за $O(primes \cdot \log n)$, где $primes$ — количество чисел среди $1 \ldots 10000$, являющихся простыми. Это число примерно равно 1300.

Итого, получаем асимптотику $O(q \cdot max(count \cdot \log n, primes \cdot \log n)) = O(q \cdot primes \cdot \log n)$.

Задача E. Новое развлечение

Автор задачи: Дмитрий Филиппов
Автор разбора: Дмитрий Филиппов

В этой задаче было дано два числа, два игрока играли в следующую игру: за один ход можно было либо поделить одно из этих чисел на его делитель, отличный от единицы, либо оба числа поделить на их общий делитель, отличный от единицы. Проигрывал тот, кто не мог сделать ход. Требовалось понять, кто выигрывает при правильной игре.

Решение: Напишем перебор. Состоянием будет два числа a, b — собственно, текущая позиция игры. Переходы будем делать так, как написано в задаче: переберем делитель d числа a, запустимся от чисел $\frac{a}{d}, b$ (здесь и дальше символ / означает деление нацело), если b делится на d, то запустимся от $\frac{a}{d}, \frac{b}{d}$. Затем переберем делитель d числа b и запустимся от чисел $a, \frac{b}{d}$.

Можно заметить, что каждый раз числа в текущем состоянии (a_1, b_1) являются делителями изначальных чисел a, b. А делителей у чисел до 10^9 мало — не более 1344. Тогда можно из состояния (a, b) делать переходы за $O(d(a) + d(b))$, где $d(x)$ — количество делителей числа x. Это довольно просто сделать, предподсчитав делители всех делителей изначальных чисел a, b.

Тогда переходов в сумме будет довольно мало, такое решение вполне себе укладывается в Time limit.

Задача F. Рутинная работа

Автор задачи: Илья Збань
Автор разбора: Илья Збань

В этой задаче нужно было отсортировать перестановку n стеками.

Решение: Пусть у нас есть 2^n чисел и n стеков. Возьмем 2^{n-1} меньших чисел, положим их из первой очереди во вторую, а 2^{n-1} больших чисел — в первый стек. Сначала отсортируем 2^{n-1} первых чисел, используя $n-1$ стек (это так же можно сделать, по индукции), а потом так же и 2^{n-1} больших чисел.

Задача G. Ломать — не строить

Автор задачи: Дмитрий Филиппов
Автор разбора: Дмитрий Филиппов

В задаче было дано n точек на плоскости, каждая из которых была соединена ровно с двумя другими. Требовалось у каждой точки удалить ровно один отрезок, выходящий из нее, чтобы в итоге никакие два отрезка не пересекались.

Решение: Эта задача сводится к задаче 2-SAT. Будем считать, что для i-й точки один из ее отрезков — это x_i, а второй — $!x_i$ (здесь и дальше под унарной операцией «!» имеется в виду операция «not»). Тогда несложно заметить, что удаление отрезка можно описать переменными, соответствующими двум его концам, а пересечение отрезков можно описать переменными, соответствующими двум концам разных отрезков (при пересечении один из отрезков точно надо удалить). Объединив все эти описания удалений отрезков, мы получим выражение, записанное в форме Крома (2-CNF). А эта задача решается за $O(D)$, где D — количество дизъюнктов в 2-CNF.

Ограничения были очень маленькие, поэтому построить выражение можно было просто перебрав все пары отрезков. Итоговая сложность решения — $O(n^2)$.

Задача H. Маленькая шалость

Автор задачи: Евгений Замятин
Автор разбора: Евгений Замятин

В задаче требовалось удалить из взвешенного графа одно ребро, чтобы кратчайшее расстояние из первой вершины до всех других изменилось для максимального числа вершин. Веса всех ребер положительны.

Решение: По исходному графу построим ориентированный граф кратчайших расстояний из вершины 1. Вершины в этом графе будут соответствовать вершинам исходного графа, а ребро из вершины a в вершину b будет тогда, когда существует кратчайший путь до вершины b, проходящий через вершину a. Построить такой граф можно с помощью алгоритма Дейкстры.

Заметим, что полученный граф является ациклическим, т.к. веса всех ребер положительны. Также заметим, что если в нашем новом графе вершине инцидентны хотя бы два ребра, то удаление

любого из этих ребер не изменит кратчайшее расстояние ни для одной вершины. Значит, нам осталось перебрать $O(n)$ ребер. А именно, нам нужно перебрать вершину. Посмотреть, сколько ребер ей инцидентно. И если ребро одно, то попытаться его удалить.

Пускай мы хотим удалить ребро, инцидентное вершине v. Тогда посчитать ответ для ребра можно так: Запустим обход в глубину из первой вершины, причем в вершину v ходить не будем. Ответом для ребра будет $n - cnt$, где cnt — количество помеченных вершин. Для всех вершин, до которых мы не дошли, расстояние в исходном графе увеличится. Следовательно это то, что мы ищем.

Возьмем максимум по всем ребрам. Это и будет ответ.

Итоговая сложность: $O(n^3)$.

Задача I. Сообщения

Автор задачи: Войт Захар
Автор разбора: Войт Захар
Пример кода: http://pastebin.com/6HhCnsJq

В этой задаче был дан невзвешенный граф, из первой вершины отправлялись сообщения в K других вершин. Сообщения могут перейти по ребру или остаться на месте за одну секунду. Два сообщения не могут в одну секунду идти по одному ребру. Нужно было вывести время прибытия последнего сообщения, если выбрать их маршруты оптимальным образом.

Решение: Будем искать ответ двоичным поиском. Пускай мы фиксировали время X и нужно проверить, смогут ли все сообщения дойти за такое время. Проверить это можно при помощи алгоритма поиска максимального потока.

Перестроим граф. Пускай каждая вершина в новом графе будет представлять собой пару (v, t) — вершина в исходном графе и время, в которое мы в этой вершине находимся. Время лежит в отрезке $[0, X]$. Проведем ребра $(v, t) \to (v, t + 1)$ и $(v, t) \to (u, t + 1)$, если вершины u и v связны. Также сделаем фиктивную вершину-сток и проведем в нее ребра из всех вершин (v, X), где v — вершина-получатель в исходном графе. Вместимости ребер $(v, t) \to (v, t + 1)$ и ребер, ведущих в вершину-сток, сделаем равными бесконечности, а вместимости ребер $(v, t) \to (u, t + 1)$ — равными единице. Тогда если максимальный поток из вершины $(1, 0)$ в вершину-сток равен K, то все сообщения можно доставить за время X, иначе - нельзя.

Задача J. Диаграмма

Автор задачи: Евгений Замятин
Автор разбора: Евгений Замятин

В этой задаче требовалось по массиву a_i и числу k найти массив b_i, такой, что в нем не более k соседних элементов различаются, и при этом сумма $|a_i - b_i|$ была минимальна.

Решение: Пусть для каждого отрезка массива с l по r мы знаем оптимальное число. Т.е. такое число t, что сумма $|a_i - t|$ минимальна для всех $i : l \le i \le r$. Тогда будем решать задачу методом динамического программирования.

Состоянием будет пара чисел (n, k) — число элементов массива, которые мы уже обработали (т.е. выставили в массиве b_i первые n чисел) и количество "перепадов которые мы имеем на данный момент.

Переход будет такой: выберем число c и поставим на отрезке с n по $n + c - 1$ оптимальное число

x, которое мы как-то умеем находить. Тогда мы перейдем в состояние $(n + c, k + 1)$. Для каждого такого c обновим состояние.

Итого, получили динамику за $O(n^2k)$. Теперь подробнее о том, как находить оптимальное число для отрезка.

Предпосчитаем для каждой пары (l, r) оптмальное значение. Делать будем так: для каждого l будем двигаться по массиву вправо, при этом пересчитывая для каждого r ответ. Нам необходимо в каждый момент времени знать упорядоченное множество элементов, которые располагаются между l и r. Тогда ответом для текущего отрезка будет медиана этого множества, т.е. число, расположенное посередине (на позиции $\lceil \frac{l+r}{2} \rceil$). Пусть ответ — не медиана. Тогда, взяв следующий элемент в сторону середины массива, мы улучшим ответ.

Поддерживать упорядоченное множество можно, например, с помощью структуры данных "set".