Задача A. Треугольники

Автор задачи: Илья Збань
Автор разбора: Илья Збань

В задаче нужно было посчитать число треугольников, построенных на данных точках, содержащих строго внутри себя заданную точку.

Решение: Давайте считать число треугольников, не содержащих внутри данную точку, и вычтем это число из $n \cdot (n-1) \cdot (n-2)/6$.
Для любого такого треугольника можно провести прямую из данной точки так, что одна из точек треугольника лежит на этой прямой, а остальные — в левой полуплоскости от нее. Зная это, можно перебрать точку, лежащую на этой прямой, и, найдя число точек на этой прямой и в левой полуплоскости от нее, из простых комбинаторных соображений найти ответ на задачу. Чтобы для каждой точки посчитать число точек в левой полуплоскости от прямой, можно вращать луч, исходящий из заданной точки, используя метод заметающей прямой.

Задача B. Геном

Автор задачи: Григорий Шовкопляс
Автор разбора: Григорий Шовкопляс

В задаче был дан переход, который представлял из себя следующее преобразование: $x_{i+1} = y_i + z_i$, $y_{i+1} = x_i + z_i$, $z_{i+1} = x_i + y_i$. Нужно было посчитать разность $x_k - y_k$ по начальным значениям x_0, y_0, z_0 и числу k.

Решение: Рассмотрим данный переход повнимательней. Первый шаг нам дан в условии, в нем $x_1 - y_1 = y_0 - x_0$. Посмотрим второй: $x_2 = y_1 + z_1 = 2x_0 + y_0 + z_0$, $y_2 = x_1 + z_1 = x_0 + 2y_0 + z_0$, таким образом $x_2 - y_2 = x_0 - y_0$. Так как от значений x, y, z результат не зависел, для любых x, y, z, через два перехода разность сохраняется, а через один меняет знак. Чтобы найти ответ, нужно посчитать $x_0 - y_0$ и умножить его на `-1`, если k — нечетное, иначе оставить без изменений.

Задача C. Стеки

Автор задачи: Илья Збань
Автор разбора: Дмитрий Филиппов

В задаче было дано n изначально пустых стеков, к которым делалить три вида запросов:

- A l r x — положить на вершину каждого из стеков с номерами с l по r число x
- G x — вернуть число, лежащее на вершине x-го стека
- R i — отменить i-й по порядку запрос добавления числа на стеки: удалить соответствующее число из каждого стека

Данная задача предполагала два возможных решения:
Решение 1: Заведем дерево отрезков. В его вершине v, которая покрывает отрезок $[a, b]$, будем хранить сет номеров запросов добавления, которые содержали в себе отрезок $[a, b]$.

Тогда запросы на добавление числа в стеки будут делаться так же, как и в обычном дереве отрезков с групповой модификацией: разбиваем отрезок в запросе на $\log n$ отрезков, а затем в сет каждой вершины, отвечающей за один из этих отрезков, добавляем номер запроса.

Запрос на удаление числа из стека делается аналогично — находим нужную вершину в дереве отрезков, удаляем из ее сета номер запроса.

Число на вершине стека ищется тоже очень просто — возьмем в дереве отрезков вершину, соответствующую этому одному стеку. В сете этой вершины самое большое число — последний неотмененный запрос на добавление к стеку из запроса. Собственно, это и будет ответом.

Решение отвечает на один запрос на $O(\log^2 n)$.

Решение 2: Заведем дерево отрезков, в каждой вершине которого теперь будет храниться не сет, а просто массив, содержащий те же номера запросов. Также будем хранить в вершине булевский массив, в котором будет храниться — удален этот запрос из вершины или нет.

Запросы на добавление числа делаются так же, как и в предыдущем решении, только теперь мы не добавляем в сет, а добавляем в конец массива.

Запрос на удаление числа из стека — просто пометить в булевском массиве, что число удалено.

Найти последнее число в стеке — пока последнее число в массиве помечено как удаленное, удалить его и перейти к предыдущему. Первое неудаленное число — ответ на запрос.

Понятно, что с каждым числом будет произведено за время всех запросов не больше трех операций: добавление в массив, установление метки, что оно удалено, и, собственно, удаление. Получаем, что суммарное время работы — $O(m \log n)$, где m — количество запросов, а n — количество стеков.

Задача D. Последовательности

Автор задачи: Илья Збань
Автор разбора: Дмитрий Филиппов

В задаче нужно было среди первых $n + 1$ натурального числа выбрать n, чтобы из них можно было составить такую последовательность чисел длиной $2 \cdot n$, что каждое число встречается в ней ровно два раза, и для каждого числа x между вхождениями его в последовательность стоит ровно $x - 1$ других чисел.

Решение: Несмотря на достаточно большие ограничения ($n \leq 100$), в этой задаче заходит перебор. Будем строить нашу последовательность с начала, но числа будем перебирать от большего к меньшему — это очень сильно ускоряет перебор. Собственно, больше ничего для этой задачи делать не нужно было, можно было только ускорять перебор — например, хранить доступные числа и незанятые позиции в двусвязных списках.

Для всех $n \leq 100$ это решение не работает, нужно было запустить его у себя локально, а потом отослать в систему сгенерированные ответы.

Комментарий

Ответа `-1` никогда не бывает.

Задача E. Экзамен

Автор задачи: Дмитрий Филиппов
Автор разбора: Дмитрий Филиппов

В этой задаче нужно было найти количество целых чисел x на отрезке с l по r, таких, что $10 \cdot x$ — точный квадрат, а $6 \cdot x$ — точный куб.

Решение: Здесь и дальше будем называть число x «хорошим», если оно удовлетворяет условиям задачи.

Либо использовав простейшие знания теории чисел, либо написав перебор, можно было заметить, что все хорошие числа x имеют вид $x = 36000 \cdot y^6$, где y — целое неотрицательное число. Теперь можно легко находить количество таких чисел на префиксе (для данного x_0 найти количество хороших чисел x, что $x \le x_0$). Для этого надо просто поделить число x_0 нацело на `36000`, а затем взять из получившегося числа корень шестой степени, тоже округлив вниз.

После этого осталось только заметить, что количество хороших чисел на отрезке с l по r — количество хороших чисел на отрезке с 0 по r минус количество хороших чисел на отрезке с 0 по $l - 1$.

Комментарий

Если писать решение в дробных числах, то нужно было быть аккуратным с взятием корня шестой степени: перед округлением вниз к нему надо было прибавить eps.

Задача F. Пароль

Автор задачи: Дмитрий Филиппов
Автор разбора: Дмитрий Филиппов

В этой задаче нужно было угадать строку из нулей и единиц длиной не более `1000`, если можно спрашивать наличие в пароле какой-либо подстроки. Нужно было сделать не больше `1024` операций. Причем была известна длина строки-пароля.

Решение: Сначала найдем самый длинный подотрезок из нулей в нашей строке. Это можно сделать бинарным поиском по длине этого отрезка. На это нам понадобится не более `10` операций.

Теперь начнем добавлять в конец этой строки из нулей другие символы. Каждый раз будем спрашивать, есть ли в пароле подстрока s+«1», где s — текущая строка. Если такая строка есть, то добавляем к s в конец единицу, иначе — ноль.

Единственное, что может нам помешать — это конец строки. Действительно, если наша текущая строка s — уже суффикс строки-пароля, то в ней может не быть как s+«1», так и s+«0». Тогда будем поддерживать, сколько нулей подряд мы добавили в конец строки. Если это количество больше, чем максимальное число нулей в строке-пароле (а мы это уже нашли бинарным поиском), то прекращаем добавлять символы — где-то мы пропустили конец строки.

Теперь давайте еще одним бинарным поиском найдем, где же мы, собственно, пропустили конец строки — сделаем бинарный поиск по количеству нулей, которые надо оставить в конце строки. На это нам потребуется еще `10` запросов (на самом деле, `9`, но это не имеет большого значения).

Теперь, когда мы нашли суффикс нашего пароля, нам осталось только добавить в начало какие-то символы. Будем действовать так же, как мы действовали с суффиксом — добавлять единицу в начало строки, а если получившейся подстроки нет, то ноль. Только теперь мы уже не пропустим начало строки, потому что мы знаем длину пароля, следовательно, операцию добавления символа в начало надо просто повторить фиксированное количество раз.

Этот алгоритм сделает не более `10` запросов на первый бинарный поиск, не более `10` запросов на второй и не более $|s| + 1$ запросов на все остальное — в конец мы добавим не более $l + 1$ лишних нулей (l — максимальное число нулей подряд в пароле), а на добавление остальных символов у нас уйдет $|s| - l$ запросов. Итого мы сделаем не более $10 + 10 + |s| + 1$ запросов, что спокойно

укладывается в ограничение 1024.

Задача G. Росомаха и стеллаж

Автор задачи: Дмитрий Филиппов
Автор разбора: Дмитрий Филиппов

В задаче было дано двоичное дерево, в каждой вершине которого написано число. За одну операцию можно было либо прибавить единицу к любой вершине, либо вычесть. Нужно за минимальное количество операций получить дерево, где во всех листьях стоит `0` или `1`, и число в каждой вершине равно сумме чисел в детях.

Решение: Будем считать динамику $dp_{vertex,value}$ — какое минимальное количество операций нужно сделать, чтобы поддерево вершины $vertex$ удовлетворяло всем свойствам из условия, если в $vertex$ поставить значение $value$. Чтобы посчитать динамику, запустим `dfs` из корня. Находясь в вершине, будем перебирать значения $value_{left}$ и $value_{right}$ — новые значения детей. Перебрав их, надо надо обновить значение динамики $dp_{vertex,value_{left}+value_{right}}$.

С первого взгляда кажется, что эта динамика работает за $O(n^3)$, но если перебирать значения $value_{left}$ и $value_{right}$ до размеров соответствующих поддеревьев, такая динамика будет работать за $O(n^2)$. Это можно легко доказать по индукции: для дерева из одной вершины это безусловно верно. Теперь сделаем переход: пусть в левом поддереве корневой вершины s_1 вершин, а в правом — s_2. Тогда по предположению индукции в левом и правом поддеревьях ответ найдется за $O(s_1^2)$ и $O(s_2^2)$ соответственно. В пересчете динамики мы сделаем $O(s_1 \cdot s_2)$ операций. Итого получаем $O(s_1^2 + s_2^2 + s_1 \cdot s_2) = O((s_1 + s_2)^2) = O(n^2)$. Переход доказан, значит этот алгоритм работает за $O(n^2)$.

Задача H. Забег

Автор задачи: Евгений Замятин
Автор разбора: Дмитрий Филиппов

В задаче надо было найти длину минимального не обязательно простого пути в неориентированном взвешенном графе.

Решение: Это была задача-шутка — надо просто найти минимальное по весу ребро и ходить по нему туда-обратно. Понятно, что это будет оптимальный маршрут. Соответственно, ответ — $(k-1) \cdot w$, где w — вес минимального ребра.

Задача I. Нечетное или четное?

Автор задачи: Григорий Шовкопляс
Автор разбора: Дмитрий Филиппов

В задаче было дано выражение, состоящее из арифметических операций `+`, `-`, `*`, из чисел произвольной длины, а также из переменных x, y. Также гарантировалось, что выражение имееет хороший вид — никакие переменные и арифметические знаки не стоят подряд, нет унарных операций, а также число не стоит рядом с переменной — между ними всегда есть арифметическая

операция.

Про переменные x, y была известна их четность, нужно было найти четность всего выражения.

Решение: Нужно было просто написать то, что просят в условии. Выражение надо было разделить на слагаемые, каждое из которых состоит из нескольких чисел и переменных, разделенных знаком умножения. Посчитать четность каждого слагаемого — просто перемножить четности всех множителей. Затем нужно было сложить (сложение и вычитание — одинаковые операции, если мы считаем только четность) четности всех слагаемых и вывести ответ.

Задача J. Игра в домино

Автор задачи: Евгений Замятин
Автор разбора: Дмитрий Филиппов

В задаче было дано n доминошек s_i, причем для всех доминошек их правый конец (количество точек на правом конце) было меньше, чем левый конец. Надо было найти наибольшее по размеру множество индексов M, что $s_{M_1}, s_{M_2}, ..., s_{M_k}$ — цепочка, где k — размер множества M, а цепочка — такая последовательность доминошек, что у любых двух соседних конец левой из них совпадает с началом правой.

Решение: Здесь и дальше будем называть доминошки отрезками, где начало отрезка — количество точек на левой половине доминошки, а конец — количество точек на правой.

Отсортируем отрезки по возрастанию их правого конца. Теперь будем считать динамику f_i — максимальная длина цепочки, заканчивающейся в координате i. Рассмотрим i-й отрезок. Если мы возьмем его в ответ, то отрезок перед ним должен будет иметь конец равный началу i-го отрезка. Значит надо попробовать улучшить ответ для координаты r_i ответом для координаты l_i: $f_{r_i} = max(f_{r_i}, f_{l_i} + 1)$. Так как отрезки отсортированы по правому концу, динамика будет считаться правильно.

Комментарий

Координаты отрезков были довольно большими, поэтому нужно было либо изначально сжать их, либо использовать структуру данных `map` для подсчета динамики.