

Разбор задачи «Маленькая шалость»

Автор идеи и разработка задачи: Демид Кучеренко

Решение первое. Отсортируем исходный массив, после этого поменяем первую пару двух соседних неравных элементов местами. Поскольку в исходном массиве все элементы были различны и количество элементов было больше двух (условие существования ответа), то это даст правильный ответ.

Решение второе. Будем случайно перемешивать входные данные, пока не получится неупорядоченная последовательность.

Разбор задачи «Глиняные таблички»

Автор идеи: Станислав Наумов

Разработка задачи: Роман Коробков

В задаче требовалось разбить строку A на подстроки, каждая из которых является подпоследовательностью надписи на некоторой табличке. При этом необходимо было минимизировать суммарную длину табличек, подпоследовательности из которых мы возьмем.

Пусть dp_i — минимальная суммарная длина табличек, чтобы набрать первые i символов из A . Тогда $dp_0 = 0$, $dp_i = \min(dp_j + |s|)$, если подстрока $A[j, i)$ входит в s , как подпоследовательность. Ответом на задачу будет $dp_{|A|}$.

Так как у надписей размер не превосходит 9, то $i - 9 \leq j < i$. Значит, необходимо быстро определять минимальную по длине надпись, которая содержит заданную подстроку.

Для этого сгенерируем все подпоследовательности каждой надписи и положим их в хеш-таблицу (`unordered_map` в C++) или бор, которые по строке будут возвращать минимальную длину надписи, содержащей эту строку.

Пусть время работы структуры $= t$. Итоговое время работы: $O(n \cdot 2^9 \cdot t + |A| \cdot 9 \cdot t)$.

Разбор задачи «Игра в монетку»

Разработка задачи: Демид Кучеренко

Заметим, что игроки в свой ход могут обеспечить величины ответ $|a_{n-1} - a_n|$, оставив на столе одну монету. Таким образом первый игрок решает, заберет ли он всё (тогда ответом будет $|y - a_n|$), или заберет всё кроме одной (ответом будет $|a_{n-1} - a_n|$). Исключением является случай, когда монета одна, тогда ответом является число $|y - a_1|$.

Разбор задачи «Челябинск-сити»

Автор идеи и разработка задачи: Роман Коробков

Заметим, что любое число длиной более 10 цифр может образовывать прогрессию, только если состоит из одинаковых цифр. Например, $22 \dots 22$ или $55 \dots 5$ будут арифметическими прогрессиями. Прогрессий же из разных цифр не очень много (примерно 250).

Можно сгенерировать все возможные прогрессии из разных цифр, а затем для каждой сгенерированной прогрессии проверить, удовлетворяет ли она условию. Также нужно не забыть учесть прогрессии из одинаковых цифр. Если явно генерировать все возможные прогрессии из одинаковых цифр, то это займет очень много времени. Можно было просто прибавить все прогрессии, которые точно подходят, $9 \cdot \max(|B| - |A| - 1)$, а затем генерировать прогрессии длины $|A|$ или длины $|B|$.

Разбор задачи «Идея»

Автор идеи и разработка задачи: Демид Кучеренко

По принципу Дирихле, ответ не существует, если $TM < N$ (мы не можем заполнить даже одну строку). Во всех остальных случаях ответ есть, покажем как его построить.

Будем заполнять матрицу “диагоналями”. Таким образом, у нас в каждой строке и каждом столбце будет получаться циклический сдвиг предыдущей строки или столбца соответственно. Покажем способ на примере, при $N = 6, T = 2, M = 3$.

```
1 2 3 1 2 3
2 3 1 2 3 1
3 1 2 3 1 2
1 2 3 1 2 3
2 3 1 2 3 1
3 1 2 3 1 2
```

Существовали также и другие тактики построения примера.

Разбор задачи «НОД цифр»

Автор идеи и разработка задачи: Николай Ведерников

Будем искать ответ в виде $X..XY00..0$, где X — цифра из исходного числа, Y — цифра, которая больше цифры числа Пети. Заметим, что чтобы ответ был как можно меньше, нам надо, чтобы часть состоящая из X , была как можно длиннее.

Для того чтобы найти ответ, посчитаем НОД для каждого префикса исходного числа, назовем этот массив gcd . Переберем i от $n - 1$ до 0. Теперь для конкретного i надо понять, можем ли мы получить ответ. Для этого переберём цифру на позиции $i + 1$ от $a[i + 1] + 1$ до 9, где $a[i + 1]$ — цифра на $i + 1$ позиции в исходном числе. Если не существует такой цифры на позиции $i + 1$, что НОД её и $gcd[i]$ больше единицы, то такое i не подходит и переходим дальше, иначе выводим ответ.

Заметим, что в этой задаче есть крайний случай для числа вида 99..99, ответом будет число 200..00. Чтобы отдельно не рассматривать этот случай, можно добавить ведущий 0 в исходное число.

Разбор задачи «Андрей и квест»

Автор идеи и разработка задачи: Алексей Саплин

Для начала нужно узнать на каком уровне находится каждая вершина, сделаем это с помощью алгоритма поиска в ширину.

Далее насчитаем две динамики: dp_front и dp_back .

- dp_front_v - минимальная стоимость пути от 1 до v .
- dp_back_v - минимальная стоимость пути от n до v по обратным ребрам.

Считаем их в порядке прямого и обратного обхода в ширину соответственно. Для каждого уровня найдем минимальное значение каждой динамики, пусть это будут dp_front_best и dp_back_best .

Если мы не строим переход, то ответ содержится в dp_front_n .

Если мы строим переход, то можем перебрать какие уровни он будет соединять, пусть это $level_begin$ и $level_end$, тогда ответ - $dp_front_best_{level_begin} + p + dp_back_best_{level_end}$. Итоговая сложность $O(n^2)$.

Заметим, что такое решение получает TL, тогда будем перебирать уровень, в который будет вести переход. Заметим, что ответ в таком случае будет таким - $\min\{dp_front_best_{max\{0, level-k\}}, \dots, dp_front_best_{level-1}\} + p + dp_back_best_{level}$. Считать минимум мы можем, с помощью структур данных таких, как set или дерево отрезков. Сложность в таком случае будет составлять $O(n \log n)$, что уже позволяет сдать задачу, однако, если использовать очередь на минимум, то итоговая сложность будет $O(n)$.

Итоговый ответ — минимум из двух найденных ответов выше.

Разбор задачи «Пароль»

Автор идеи: Демид Кучеренко

Разработка задачи: Николай Ведерников

Переберем количество букв 'а' в пароле. Пусть их k (чётное), тогда каждый из оставшихся $n - k$ символов будет либо буква 'b', либо буква 'c', что дает в итоге 2^{n-k} вариантов расставить буквы 'b' и 'c' на оставшиеся позиции. Кроме того, буквы 'а' мы могли расставить C_n^k способами, что в итоге для фиксированного k даёт $C_n^k \cdot 2^{n-k}$ вариантов паролей. Переберем все возможные k и просуммируем. Итого получаем $\sum_{k=0, k \text{ even}}^n C_n^k \cdot 2^{n-k}$. Но так как n большое мы не можем по быстро вычислить эту формулу.

Рассмотрим функцию: $\frac{1^k + (-1)^k}{2}$. Она равна 0, если k нечётное, а иначе 1.

Тогда $\sum_{k=0, k \text{ even}}^n C_n^k \cdot 2^{n-k} = \frac{1}{2} \cdot \left(\sum_{k=0}^n C_n^k \cdot 1^k \cdot 2^{n-k} + \sum_{k=0}^n C_n^k \cdot (-1)^k \cdot 2^{n-k} \right)$. По биному Ньютона получаем $\frac{1}{2} \cdot ((2+1)^n + (2-1)^n) = \frac{3^n+1}{2}$. Но не забываем, что пароль может быть введен как маленькими буквами, так и большими, поэтому ответ надо умножить на 2. Итого ответом на эту задачу будет $3^n + 1$.

Так как n довольно большое, то 3 возводить в степень следует быстрым возведением в степень.

Отметим, что решить эту задачу можно, если посчитать количество паролей при малых n и заметить закономерность.

Разбор задачи «Ваша любимая задача возвращается»

Разработка задачи: Демид Кучеренко

Предполагалось построить ответ руками. В данной задаче при автоматическом построении ответа, хорошо работает жадный алгоритм, его можно было применить на бумаге. Поставим сначала две одинаковые фигуры, затем будем жадно ставить фигуры, которые еще не встречались ранее, и в конце немного переберем.

Возможный вариант разделения:

```
1 1 1 2 2 2 3 3 3
1 1 1 2 2 2 4 3 3
5 5 5 5 6 4 4 4 3
7 7 5 5 6 6 4 4 8
7 9 9 9 6 6 8 8 8
7 9 9 10 6 11 11 12 8
7 7 9 10 11 11 12 12 8
10 10 10 10 11 11 12 12 12
```

Разбор задачи «Красивые числа»

Автор идеи и разработка задачи: Николай Ведерников

Будем решать задачу: сколько красивых чисел на отрезке от 1 до x . Тогда ответом на исходную задачу будет $count(r) - count(l - 1)$.

Заметим, что максимальная сумма цифр у чисел $\leq 10^{18}$ не превосходит 200. Кроме того, простых чисел меньше 200 не так много. Тогда для каждого запроса будем перебирать какая простая сумма цифр должна быть в красивом числе. Теперь для фиксированной суммы p и числа x нам надо уметь говорить сколько чисел меньше x имеют сумму p .

Для этого воспользуемся динамическим программированием. Посчитаем $dp[i][j]$ — количество чисел длины i с суммой цифр j . Тогда чтобы ответить на запрос, нам надо пройти по числу, перебрать цифру на позиции i и добавить к ответу значение $dp[len - i - 1][p - sum - c]$, где len — длина числа x , c — цифра на позиции i , sum — сумма цифр на префиксе от 0 до $i - 1$. Таким образом, на один запрос мы умеем отвечать за количество простых чисел меньше 200 умножить на 10 умножить на длину числа, что при данных ограничениях получает превышение по времени.

Ускорим наше решение. Мы не будем перебирать цифру на позиции i , а предподсчитаем с помощью $dp2[i][j][k]$ — количества чисел длины i с суммой цифр j , где на позиции $i + 1$ с конца стоит

цифра k . Таким образом, на один запрос мы умеем отвечать за количество простых чисел меньше 200 умножить на длину числа, что при данных ограничениях так же получает превышение по времени.

Для того чтобы ускорить наше решение, уберем перебор по простым числам. Для этого посчитаем $dp3[i][sum][k] = \sum_{p: \text{primes}} dp2[i][p - sum][k]$ — количество чисел, у которых осталось ещё i позиций, на позиции $i + 1$ с конца стоит цифра k и префикс имеет сумму sum и эту сумму можно дополнить до простых.

С помощью $dp3$ мы умеем отвечать на запрос за длину числа, что укладывается в ограничения по времени.

Разбор задачи «Ремонт кольцевой»

Автор идеи: Демид Кучеренко

Разработка задачи: Демид Кучеренко, Денис Шпаковский

В задаче были даны отрезки на окружности, необходимо было для каждого отрезка вычислить, сколько еще отрезков нужно задействовать вместе с ним, чтобы покрыть всю окружность.

Применим общий прием для задач на окружности: каждый отрезок продублируем новым с координатами $[x + m; y + m]$. С помощью метода сканирующей прямой для каждого отрезка x найдем такой отрезок y , что они пересекаются, и при этом y имеет самый дальний правый конец. Таким образом мы получим для каждого отрезка максимальную по координате точку, до которой можно добраться, начав от отрезка x и последовательно закрывая все точки окружности.

Несложно понять, что теперь нам нужно для каждого отрезка $[x, y]$ найти такое число t , что совершив t последовательных переходов на отрезок с максимальной правой границей, мы достигнем точки с координатой $\geq x + m$. Для каждой точки насчитаем с помощью двоичных подъемов величины $up[x][t]$ — номер отрезка, до которого мы можем дойти, начиная с вершины x , совершив 2^t переходов. Тогда $up[x][t] = up[up[x][t - 1]][t - 1]$.

Теперь можно с помощью массива up восстановить ответ для отрезка x . Попробуем прыгнуть на расстояние $p = 2^t$, если таким образом покрыто больше чем полная окружность (можно делать с помощью проверки левой границы отрезка, в который прыгаем), то уменьшим степень t , иначе добавим к ответу 2^t , установим $x = up[x][t]$, и тоже уменьшим степень t (поскольку на 2^{t+1} прыгать мы уже пытались).

Решение работает за $O(n \log n)$ на сортировку и $O(n \log n)$ на двоичные подъемы. Также есть решение за $O(n \log n)$ на сортировку и $O(n)$ на вычисление ответа с помощью обхода в глубину.

Разбор задачи «Орел и решка»

Автор идеи: Демид Кучеренко

Разработка задачи: Денис Шпаковский

Будем решать задачу методом динамического программирования. Пусть $dp[i]$ ($1 \leq i \leq n$) — ответ на задачу для $n_{\text{new}} = i, k_{\text{new}} = k_{\text{last}}$. Допустим, мы уже посчитали $dp[i]$ для $1 \leq i \leq j - 1$ и теперь хотим найти значение $dp[j]$.

Решение за $O(nk)$

Будем вычислять значение $dp[i]$ для каждого $1 \leq i \leq n$ за $O(K)$. Тогда получим решение за $O(nk)$. Итак, хотим найти значение $dp[j]$. Переберем длину блока из одинаковых букв, которая закончится в позиции j . Длина такого блока не превосходит k . Тогда сразу за этим блоком должна стоять буква, не совпадающая с буквой, из которой состоит блок. Таким образом, получаем следующую формулу пересчета:

$$dp[j] = \sum_{l=1}^k dp[j - k].$$

Где $dp[i] = 0, i < 0$, а значение $dp[0]$ удобно считать равным 2.

Решение за $O(n)$

Заметим, что переход можно делать за $O(1)$, используя префиксные суммы. Получим формулу пересчета:

$$dp[j] = 2 * dp[j - 1] - dp[j - k - 1].$$

Где ответом является значение $dp[n] - dp[n - 1]$.

Разбор задачи «Цветной металл»

Автор идеи и разработка задачи: Савелий Григорьев

В этой задаче, в первую очередь, хотелось бы научиться много раз сортировать подотрезки массива за приемлемое время. Заметим, что сортировать произвольный массив достаточно сложно, но если бы в нём было мало различных значений, то мы могли бы отсортировать его за количество различных. Эта мысль подталкивает к решению: давайте сделаем бинпоиск по ответу (правда ли, что число на финальной позиции не больше x). Пусть на текущей итерации бинпоиска мы проверяем число x . Тогда заменим числа на 0 (если число не больше x) или на 1 (если число больше x). Чтобы отсортировать подотрезок, необходимо посчитать число нулей и единиц на нём (пусть их будет a и b соответственно), а затем заменить его на два отрезка из a подряд идущих нулей и b подряд идущих единиц (порядок этих отрезков зависит от того, сортируем мы по возрастанию или по убыванию). Рассмотрим две оставшиеся операции: третья операция, если значение элемента равняется 0, не меняет подотрезок, иначе заменяет весь подотрезок на 1. Четвёртая операция, если значение элемента равняется 0, заменяет весь подотрезок на 0, иначе не изменяет его. Реализовать эти операции можно либо с помощью дерева отрезков, либо, поддерживая в сети отрезки из подряд идущих нулей и единиц. Итоговая асимптотика: одна итерация бинпоиска - $O(n \log n)$, всего итераций $O(\log n)$, итого $O(n \log^2 n)$.