Московская олимпиада школьников по информатике 2019

Разбор задач

Задача «Морской бой»

Автор и разработчик задачи: Григорий Резников

Решение

- Предположим, что w1 != w2
- Классифицируем отмеченные клеточки на две категории
- В первой категории будут клеточки, соседние по углу с фигурой
- Таких клеток 5
- Во второй категории будут клеточки, соседние по стороне с фигурой
- Число таких клеток равно периметру фигуры минус 1
- Таким образом, ответ равен 2 * (max(w1, w2) + h1 + h2) + 4
- Заметим, что при w1 = w2 эта формула тоже корректна
- Асимптотика *O(1)*

Задача «Автобусы»

Автор задачи: жюри олимпиады
Разработчик: Филипп Грибов

Формальная постановка

- Три автобуса одновременно выезжают и ходят с интервалами a, b и c минут соответственно ($a, b, c \le 10^6$).
- В сутках t минут ($t \le 10^9$).
- Сутки начинают свой отсчёт с момента отправления всех трёх автобусов
- Требуется узнать, сколько раз все три автобуса были на остановке в сутках под номером d ($d \le 10^9$).

20 баллов (a, b, c, t, d ≤ 100)

- Заметим, что если в какой-то момент времени x все 3 автобуса были на остановке, то x делится на a, на b, на c.
- Так как t, d ≤ 100, то все интересующие нас моменты времени не превышают 10000. Тогда переберём все эти моменты времени.
- Для каждого момента времени проверим, что он лежит в интересующих нас сутках (т.е не меньше (d - 1) * t и строго меньше d * t).
- Среди всех таких моментов посчитаем число таких, которые делятся на a, b, c. Это и будет ответом на задачу.
- Асимптотика $O(d * t)$

50 баллов ($t \le 10^6$)

- Вместо того, чтобы перебирать все моменты времени, переберём только те, которые лежат в интересующих нас сутках, то есть между $(d - 1) * t$ и $d * t - 1$ (включительно).
- Среди всех этих моментов посчитаем число тех, которые нацело делятся на a, b, c. Это и будет ответом на задачу.
- Асимптотика $O(t)$

Важное замечание

- Заметим, что минимальный положительный момент времени, когда все автобусы были на остановке это *наименьшее общее кратное чисел a, b, c* (далее НОК(a, b, c)).
- Любое другое число, которое делится на эти 3 числа обязательно должно делиться на НОК(a, b, c), и любое число, которое делится на НОК(a, b, c) делится на a, b, c.
- Тогда нам для решения задачи достаточно найти НОК(a, b, c) и посчитать число моментов времени в сутках, которые на него делятся.

Нахождение НОК(a, b, c)

- Для нахождения НОК(a, b, c) достаточно найти НОК(a, b), а потом найти НОК(НОК(a, b), c). Научимся находить НОК двух чисел.
- Для этого достаточно воспользоваться известным фактом, что НОК двух чисел это их произведение, делённое на их *наибольший общий делитель* (Далее НОД), и воспользоваться стандартным алгоритмом поиска НОД.
- Однако можно заметить, что НОК не превышает произведения чисел, и так как каждый раз, когды мы ищем НОК чисел x и y, $y \le 10^6$, то можно перебрать все числа от x до x*y, делящиеся на x, и найти минимальное делящееся на y. Это и будет НОК(x, y).
- Асимптотика такого алгоритма $O(a + b + c)$. Асимптотика стандартного алгоритма поиска НОД $O(\log a + \log b + \log c)$

20 баллов (d=1)

- Мы уже свели задачу к поиску числа моментов времени, которые делятся на НОК(a, b, c). Теперь надо найти число моментов времени в первых сутках, которые делятся на НОК(a, b, c).
- В первых сутках моменты времени это 0, 1, 2, …, t - 1.
- Для любого n число моментов времени от 0 до n, которые делятся на НОК(a, b, c) это n / НОК(a, b, c) + 1.
- Тогда ответ: (t - 1) / НОК(a, b, c) + 1.
- Асимптотика $O(a + b + c)$ или $O(\log a + \log b + \log c)$ (в зависимости от реализации поиска НОК).

Полное решение

- Нам надо найти число моментов времени в d-х сутках, которые делятся на НОК(a, b, c).
- В d-х сутках моменты времени от (d - 1) * t до d * t - 1 (включительно).
- Тогда достаточно найти число таких моментов времени от 0 до d * t - 1 и вычесть из этого число таких моментов от 0 до (d - 1) * t - 1. Так мы получим ответ.
- Асимптотика $O(a + b + c)$ или $O(\log a + \log b + \log c)$ (в зависимости от реализации поиска НОК).

Задача «День рождения»

Автор задачи: жюри олимпиады
Разработчик: Дмитрий Саютин

Постановка задачи

- Дано n чисел a_1, a_2, …, a_n

- Нужно переставить их таким образом, чтобы наибольшая разность соседних чисел (где первое и последнее число также являются соседними) была бы минимальной возможной.

Частичные решения

- $n \le 5$: переберём все $n!$ перестановок чисел a_i и выберем лучшую.

- Жадные идеи работающие на какой-то части тестов.

- Тесты $a = [1, 2, \ldots, n]$: любая конструкция, приводящая к ответу 2.

Полное решение

- Упорядочим $a_1, a_2, \ldots, a_n$ так, что $a_i \le a_{i+1}$
- Тогда в ответ выпишем сначала все элементы с нечётными индексами, в порядке возрастания, а потом все элементы с чётными индексами, в порядке убывания.

- Например для $n=5$: $a_1\ a_3\ a_5 \mid a_4\ a_2$
- Например для $n=6$: $a_1\ a_3\ a_5 \mid a_6\ a_4\ a_2$

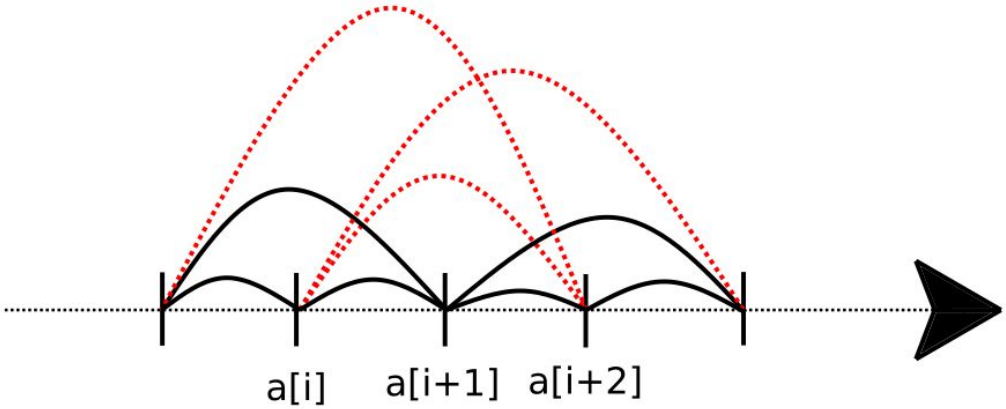
Checked on many tests

Осталось доказать. Заметим, что наша конструкция приводит к ответу не превосходящему $\max a_{i+2} - a_i$.

Несложно доказать, что для любой разницы $a_{i+2} - a_i$ ответ не может быть меньше. Для этого изобразим возможные переходы как граф. Вершинами будут числа a_i, решению задачи соответствует гамильтонов (то есть проходящий по всем вершинам) цикл.

works

Попробуем справиться не используя $a_{i+2} - a_i$. Красным отмечены запрещённые переходы. Несложно видеть, что a_{i+1} является точкой сочленения и гамильтонова цикла не существует.



Задача «Произведение строк»

Автор и разработчик задачи: Вениамин Феафанов

Формальная постановка

- Введём операцию “произведения” двух строк: $s \bullet t = t + s[0] + t + s[1] + \ldots + s[k - 1] + t$, где k - длина s.
- Назовём строку “хорошей”, если все символы в ней одинаковые.
- Дано n строк: s_1, s_2, … s_n.
- Требуется найти длину наибольшей хорошей подстроки в строке $((\ldots ((s_1 \bullet s_2) \bullet s_3) \bullet \ldots) \bullet s_n$
- Обозначим за S сумму длин всех данных строк, а за L - длину конечного произведения.

20 баллов, L ≤ 100

- Посчитаем конечную строку честно. Так как длина s • t хотя бы вдвое больше длины s, генерация работает за $O(L + L / 2 + L / 4 + ...) = O(L)$
- На получившейся строке посчитаем ответ, например, перебрав все подстроки и для каждой проверив, хорошая ли она.
- Это решение за $O(L + L^3) = O(L^3)$

40 баллов, L ≤ 100000

- Как и в предыдущем решении, сгенерируем конечную строку честно.
- Посчитаем ответ более эффективным линейным алгоритмом за один проход по строке: будем хранить длину текущего блока из одинаковых символов и сбрасывать, обновляя ответ, когда символ меняется.
- Это решение за $O(L + L) = O(L)$

20 и более баллов, n = 2

- Пусть $run_s[c]$ - длина наибольшей хорошей подстроки из символа c в строке s.
- Такой массив для строки можно насчитать за один проход по ней: каждый конец блока из одинаковых символов будем обновлять ответ для соответствующего символа.
- Пусть первая строка - s, вторая - t, а длина t - m.
- Насчитаем run_s и run_t. Через них будем считать $run_{s \bullet t}$
- Тогда ответ - максимальное значение в $run_{s \bullet t}$

- $run_{s \bullet t}[c] \geq run_t[c]$. Для некоторых символов неравенство может быть строгим в таких случаях:
  - Если символа c нет в t, но есть в s, то ответ для c равен 1.
  - Если символ c - крайний символ в строке t, и он есть в s, ответ можно обновить значением “длина максимального блока из символа c на краю t“ + 1.
  - Если символ c - крайний и справа, и слева, и есть в s, правый блок одной копии строки t через символ из s склеивается с левым из следующей копии.
  - Если t полностью состоит из символа c, много копий t склеиваются в один блок через $run_s[c]$ символов.

100 баллов, полное решение

- Будем хранить для строки 6 значений:
  - Ответ (длина наибольшей хорошей подстроки).
  - Левый символ.
  - Длина наибольшей хорошей подстроки, начинающейся с первого символа.
  - Аналогично справа.
  - Хорошая ли сама строка.

- Зная эти значения для строк s и t, можно посчитать значения для их соединения (s + t) за *O(1)*:
  - Левый символ - левый символ строки s.
  - Правый - правый в строке t.
  - Длина хорошей строки слева равна таковой в s. Если s - хорошая, и левый символ t совпадает с левым символом s, к этому значению надо прибавить длину хорошей подстроки слева в t.
  - Аналогично для длины хорошей подстроки справа.
  - Строка хорошая, если и s, и t хорошие, и у них совпадают левые символы.

Ключевая идея

- Заметим, что умножение строк ассоциативно, то есть $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- Значит, мы можем идти с конца и поддерживать наши 6 значений для произведения последних нескольких строк: сначала s_n, затем $s_{n-1} \cdot s_n$, затем $s_{n-2} \cdot (s_{n-1} \cdot s_n)$ и так далее
- Конечный результат будет таким же, как при произведении с начала.

- Для последней строки посчитаем значения честно.
- Теперь научимся считать значения для s • t, в предположении, что для t они уже известны.
- s • t = t + s[0] + t + s[1] + … + t
  Мы умеем поддерживать значения для соединения строк, и можем тривиально посчитать их для каждого символа s по отдельности как для строки, поэтому просто сделаем все соединения.
- Всего таких неявных умножений мы сделаем n - 1, и для каждого соединений на одно больше, чем длина текущей строки, поэтому решение работает за *O(S)*

Задача «Выбор гурмана»

Автор задачи: Елена Андреева
Разработчик: Евгений Шульгин

Формальная постановка

- Загадано два массива из n и m чисел.
- Отгадать их надо в соответствии с таблицей размера nxm, где для каждой пары чисел из разных массивов есть информация, какое из них больше, меньше или что числа равны.
- Если ответ существует, найти такой, где максимальное число в массивах как можно меньше.

20 баллов ($1 \le n, m \le 3$)

- В любом правильном ответе все числа не превосходят $n + m$.
- Делаем полный перебор всех возможных вариантов ответа, проверяем их на соответствие таблице, обновляем оптимальный ответ.
- Перебор работает за $O((n + m)^{(n + m)})$

+10-30 баллов (Нет символов ‘=’ в таблице)

- Жадное решение - на каждом шаге постоянно берём те блюда, у которых все символы в соответствующей строке/столбце, равные ‘>’ (если они есть), указывают на блюда, взятые на предыдущих шагах.
- Присваиваем таким блюдам минимально возможное натуральное число с учетом символов ‘>’.
- Если нельзя выбрать такое блюдо, то ответ “No”
- В зависимости от решения (наивный поиск каждый раз или поддержка структурами данных) сложность $O(n^2m^2)$ или $O(nm\ lognm)$ и баллы от 10 до 30.

Решение на 100 баллов (ключевая идея)

- Используем СНМ (систему непересекающихся множеств) на n + m элементов и будем объединять множества, если какие-то два элемента из них имеют отношение ‘=’
- Далее можно либо завести таблицу сравнений размером kxk без ‘=’ (где k - количество множеств) и использовать решение задачи без символов ‘=’
- Также представим каждое множество как вершину, с направленными ребрами от “бóльшей” вершины к “меньшей”
- Если в полученном графе есть цикл - ответ “No”
- Иначе значение в каждой вершине равно самому длинному пути из нее

Задача «Ася и котята»

Автор и разработчик задачи: Никита Сендерович

Формальная постановка

- Исходно есть n одноэлементных множеств.
- На каждом шаге объединяются некоторые два множества. Через n - 1 шаг процесс заканчивается.
- Требуется поставить исходные элементы в ряд так, чтобы при каждой процедуре объединения сливались множества, соседствующие в этом ряду.

20 баллов (n ≤ 10)

- Перебор всех перестановок элементов.
- Моделирование процесса объединения с проверкой, что на каждом шаге сливаются соседние множества.
- Порядка n! операций.

40 баллов (n ≤ 100)

- Любые решения, медленнее чем за $O(n^2)$.

70 баллов (n ≤ 2000)

- Храним для каждого множества список его элементов.
- Когда нужно слить множества с элементами X и Y пробегаем по всем спискам, определяем, в каких списках находятся элементы, конкатенируем.
- Сложность $O(n^2)$.

100 баллов

- В дополнение ко спискам храним ещё для каждого элемента номер списка, которому он принадлежит.

- Когда нужно слить множества с элементами X и Y, за *O(1)* определяем, в каких списках находятся элементы.

- Делаем меньшее из множеств частью большего, добавляя меньший список к большему и переопределяя номер списка для всех элементов меньшего множества.

100 баллов

- Ни для какого элемента номер списка не изменится более log n раз, поскольку каждое изменение означает по крайней мере удвоение множества, которому этот элемент принадлежит.

- Это простейший вариант *системы непересекающихся множеств* (СНМ), *O(n log n)*.

- Более сложные варианты СНМ с дополнительной эвристикой сокращения пути дают *O(n * A(n))*.

Задача «Самая опасная акула»

Автор и разработчик задачи: Владимир Романов

Формальная постановка

- Даны массивы h, c размера m, где h_i, c_i - высота и стоимость i-ой доминошки.
- За c_i монет мы можем толкнуть доминошку влево или вправо. Ее падение может опрокинуть доминошку с индексом j, если $|i - j|$ меньше h_i и $j < i$, если толкали влево, иначе $i < j$. В свою очередь, доминошка j может уронить другие доминошки и т. д..
- Доминошку нельзя уронить более 1 раза.
- Требуется за минимальное число монет уронить все доминошки.

11 баллов ($1 \le m \le 10$)

- Доминошки можно ронять в порядке возрастания индексов.
- Рассмотрим с помощью полного перебора все способы уронить доминошки.
- У нас 3 варианта: уронить текущую доминошку влево, вправо или пропустить ее.
- Данное решение работает за $O(3^m m)$.

20-42 балла ($1 \le n \le 500\text{-}5000$)

- Заметим, что при падении одной доминошки падает непрерывный отрезок доминошек.
- Пусть L_i - минимальная по индексу и R_i - максимальная доминошки, которые может уронить i-ая доминошка.

20-42 балла ($1 \le m \le 500\text{-}5000$)

- Для нахождения ответа воспользуемся методом динамического программирования. Пусть dp[i] - минимальное число монет, если уронили первые i доминошек.
- dp[i + 1] = min(dp[j] + c[i] (где $L_i \le j \le i$), dp[j] + c[j] (где $R_j = i$))
- Данное решение работает за $O(m^2)$.

54-67 баллов ($1 \le m \le 10^5$-10^6)

- Предподсчитаем L_i, R_i за *O(m)* при помощи стека.
- dp[j] + c[i] (где $L_i \le j \le i$)
  - Вынесем c[i], тогда нам нужно найти минимум dp на отрезке $[L_i; i]$.
- dp[j] + c[j] (где $R_j = i$)
  - Заметим что нам достаточно для j один раз прорелаксировать ответ в $dp[R_j + 1]$ значением dp[j] + c[j]
- Будем хранить dp в дереве отрезков.
- Данное решение работает за *O(m log m)*.

+11 баллов ($c_i = 1$)

- Будем рассматривать возрастающий стек значений динамики.
- Тогда чтобы узнать минимум на отрезке нам достаточно посмотреть последние значения стека.
- При добавлении нового элемента мы удаляем все, кроме одного, значения динамики, которые мы рассмотрели.
- Данное решение работает за $O(m)$.

Полное решение

- Заметим, что не вложенные в друг друга отрезки $[L_i; i]$ не пересекаются.
- Тогда мы можем удалять из стека все значения динамики, кроме последнего, которые были рассмотрены на i-ом шаге.
- Данное решение работает за $O(m)$.