Разбор задачи «A. Машинки»

Пусть машинка, движущаяся со скоростью V_1, едет против часовой стрелки, а машинка, движущаяся со скоростью V_2 — по часовой. Будем считать, что направление против часовой стрелки — положительное направление движения по окружности.

Тогда через время T первая машинка, двигаясь со скоростью V_1, окажется в точке $P_1 = V_1 T \mod L$ (mod — операция взятия остатка от деления) на окружности, а вторая — в точке $P_2 = L - V_2 T \mod L$. Таким образом, минимальное расстояние между машинками равно $min(|P_1 - P_2|, L - |P_1 - P_2|)$.

Для решения первой подзадачи необходимо производить вычисления в 64-битных целочисленных типах данных.

Так как во второй подзадаче входные данные ограничены 10^{18}, то значения $V_1 T$ и $V_2 T$ могут достигать 10^{36}, что превышает вместимость 64-битных целочисленных типов данных. Для полного решения без собственной реализации длинной арифметики можно воспользоваться языком, который имеет встроенную «длинку» (*Python*, *Java*), либо реализовать умножение двух чисел по модулю через сложение по аналогии с реализацией быстрого возведения в степень по модулю через умножение.

Временная сложность алгоритма — $O(\log T)$ (умножение через сложение), ёмкостная — $O(1)$.

Разбор задачи «B. Игра в числа»

Для решения первых двух подзадач достаточно перебрать все возможные значения A, вычисляя для каждого из них соответствующее B и их НОД, и выбрать наилучшую пару.

Для третьей подзадачи перебрать все варианты не представляется возможным. Рассмотрим более эффективное решение.

Пусть $N = p_1^{\gamma_1} p_2^{\gamma_2} \dots p_m^{\gamma_m}$, где $p_1 < p_2 < \dots < p_m$ — простые числа, а $k = \text{НОД}(A, B)$. Тогда $A = \alpha k, B = \beta k$, а значит:

$$\alpha k + \beta k = N \iff (\alpha + \beta)k = N \iff (\alpha + \beta)k = p_1^{\gamma_1} p_2^{\gamma_2} \dots p_m^{\gamma_m}$$

Так как, по условию задачи, необходимо максимизировать k и затем минимизировать A, то $k = p_1^{\gamma_1 - 1} p_2^{\gamma_2} \dots p_m^{\gamma_m} = N/p_1$, $\alpha = 1$, $\beta = p_1 - 1$.

То есть, ответ на задачу: $A = N/p_1, B = N - N/p_1$, где p_1 — минимальный простой делитель числа N. Причём, если N — составное число, то $p_1 \leqslant \sqrt{N}$, иначе $p_1 = N$.

Таким образом, для решения задачи необходимо найти минимальный простой делитель числа N (перебрав только числа на отрезке $[2; \sqrt{N}]$), а затем вывести ответ, исходя из формул, представленных выше.

Временная сложность алгоритма — $O(\sqrt{N})$, ёмкостная — $O(1)$.

Разбор задачи «C. Игра в клеточки»

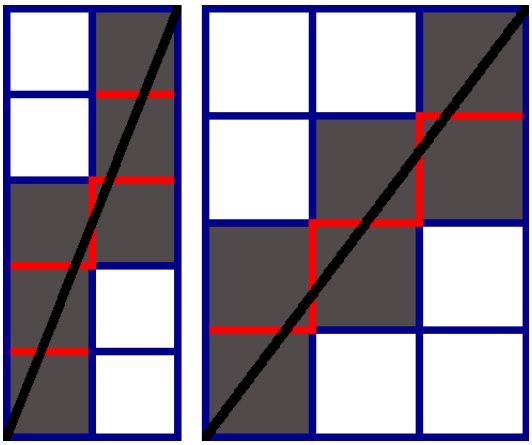
Заметим, что ответ для отрезка $(a, b) - (c, d)$ равен ответу для отрезка $(0, 0) - (X, Y)$, где $X = |a - c|, Y = |b - d|$. Действительно, первый отрезок можно получить из второго путём переноса и отражения относительно осей.

Для решения первой подзадачи достаточно перебрать все клетки внутри прямоугольника, противоположные углы которого находятся в заданных точках, и проверить их на пересечение с отрезком. Временная сложность такого решения — $O(X^2 + Y^2)$.

Для решения второй подзадачи можно перебрать значение первой координаты от 0 до $X - 1$, на каждом шаге поддерживая диапазон значений второй координаты, содержащий пересекающиеся с отрезком клетки, и сдвигая этот диапазон при переходе к следующему шагу. Временная сложность такого решения — $O(X + Y)$.

Решение задачи на полный балл можно свести к следующим этапам:

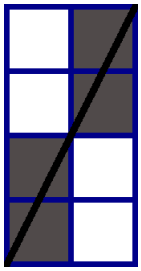
1. Рассмотрим случай, когда отрезок не проходит через точки с целочисленными координатами (помимо концов отрезка). Тогда отрезок пересекает новую клеточку при каждом пересечении координатной сетки (отмечено красным на картинке), плюс первая клеточка. Таким образом, суммарное количество пересечений клеточек равно $X + Y - 1$.



2. Рассмотрим общий случай.

   Пусть $G = \text{НОД}(X, Y)$. Заметим, что на отрезке $(0, 0) - (X, Y)$ содержатся следующие точки с целыми координатами: $(i \cdot dx, i \cdot dy)$, где $dx = X/G, dy = Y/G, i = 0, \ldots, G$.

   Каждый из отрезков $((i-1) \cdot dx, (i-1) \cdot dy) - (i \cdot dx, i \cdot dy)$, где $i = 1, \ldots, G$, можно рассматривать как отдельную подзадачу, причём любая из них сводится к задаче с отрезком $(0, 0) - (dx, dy)$, решение которой представлено выше. Таким образом, ответ в данном случае равен $G \cdot (dx + dy - 1)$.



Временная сложность алгоритма — $O(\log max(X, Y))$, ёмкостная — $O(1)$.

Разбор задачи «D. Урок географии»

Для решения задачи необходимо выделить на карте острова, подсчитать площадь каждого из них и вывести максимальное значение.

Одним из способов сделать это является перебор всех клеток карты, который при обнаружении очередной непосещённой клетки суши запускает обход в ширину, начинающийся с неё и помечающий все клетки обнаруженного острова как посещённые, одновременно подсчитывая их количество.

Временная и ёмкостная сложности алгоритма — $O(N^2)$.

Разбор задачи «E. Менеджер задач»

Заметим, что множество задач и зависимости между ними можно представить в виде ориентированного графа без циклов.

Для решения первой подзадачи достаточно выполнить обход графа из каждой вершины, подсчитывая число посещённых вершин. При эффективном хранении графа (например, с помощью списка рёбер) временная сложность такого алгоритма — $O(N \cdot M)$, а ёмкостная — $O(N + M)$.

Рассмотрим полное решение задачи. Выполним топологическую сортировку графа, то есть упорядочим вершины таким образом, чтобы рёбра шли только из вершин с большим номером к вершинам с меньшим (для этого можно воспользоваться поиском в глубину). Переберём вершины в порядке увеличения номеров. Пусть R_v — множество вершин, достижимых из вершины v, а A_v – множество вершин, смежных с v (то есть таких вершин u, что существует ребро (v, u)). Тогда $R_v = \cup R_u$ для всех $u \in A_v$, причём все R_u уже известны в момент рассмотрения вершины v. Ответом для вершины v является $|R_v|$.

Заметим, что суммарная мощность всех множеств может достигать значений, близких к N^2, а для вычисления всех множеств необходимо выполнить M операций объединения. Ясно, что при наивной реализации хранения и объединения таких множеств программа не уложится в ограничения по времени и памяти.

Чтобы обойти обе указанные проблемы, каждое множество можно хранить в виде массива из $\lceil \frac{N}{64} \rceil$ 64-битных чисел, где наличие или отсутствие вершины v в множестве соответствует значению 1 или 0 бита номер $v \mod 64$ в числе с индексом $v \operatorname{div} 64$ (индексация начинается с нуля). Тогда для объединения множеств нужно выполнить операцию побитового ИЛИ для соответствующих элементов массивов, представляющих эти множества.

Для вычисления мощности множества за менее чем N операций разобьём каждое из 64-битных чисел, представляющих это множество, на, например, 8 групп по 8 битов. Для каждой из 2^8 комбинаций битов заранее предподсчитаем количество единичных битов в ней. Тогда количество единичных битов в 64-битном числе можно относительно быстро вычислить, сложив предподсчитанные значения для каждой из групп его битов. Более эффективный приём можно найти в книге Г. Уоррена мл. *Алгоритмические трюки для программистов* в главе «Подсчёт битов».

Заметим, что оптимизации выше не изменяют асимптотическую сложность решения по сравнению с наивной реализацией (временная — $O(N \cdot M)$, ёмкостная — $O(N^2)$), однако существенное уменьшение констант позволяет уложиться в ограничения задачи.

Разбор задачи «F. Инфляция»

Пусть $A = I \operatorname{div} 100$, $B = I \mod 100$, где *div* и *mod* — операции целочисленного деления и взятия остатка от деления соответственно. Тогда A и B являются целой и дробной частями одного процента от числа I. Таким образом, чтобы найти ответ, необходимо каждое из них умножить на P, а затем просуммировать и округлить согласно условию задачи.

Для решения первых двух подзадач достаточно производить вычисления в 32-битных и 64-битных целочисленных типах данных соответственно.

Для решения третьей подзадачи необходимо реализовать длинную арифметику (реализации сложения длинных чисел либо умножения длинного числа на короткое достаточно) или воспользоваться языком программирования, который имеет встроенную длинную арифметику (*Python*, *Java*).

Временная и ёмкостная сложности алгоритма — $O(\log I)$.

Разбор задачи «G. Статистика»

Для решения первых двух подзадач достаточно выполнить сортировку заданной последовательности и вычислить ответ. Для первой подзадачи можно использовать сортировку за $O(N^2)$, а для второй необходима сортировка за $O(N \log N)$.

Для полного решения нужно заметить, что хотя ограничение по памяти не позволяет хранить всю последовательность, для получения ответа достаточно знать лишь $N \operatorname{div} 2 + 1$, например, наибольших чисел из неё. Можно предложить следующую схему решения:

1. считать $N \operatorname{div} 2 + 1$ чисел в память;
2. считать оставшиеся числа, на каждом шаге сначала добавляя очередное число к имеющемуся множеству, а затем удаляя минимальное число из этого множества;
3. вычислить ответ, используя, в зависимости от чётности N, одно или два минимальных числа хранимого множества.

Структурой данных, позволяющей реализовать операции из пункта 2 с временной сложностью $O(\log N)$ без дополнительного расхода памяти, является куча. Временная сложность всего алгоритма — $O(N \log N)$.