Разбор задачи «Найти Николая П.»

Решение 1. Будем поддерживать квадрат, в котором находится искомая точка. Изначально это вся наша область. Теперь, пока квадрат не сошелся в точку, повторяем следующие действия:

1. Сделаем запросы в углы текущего квадрата.

2. Разделим его на 4 маленьких квадратика.

3. В зависимости от того, к какому углу ближе искомая точка, обновим текущий квадрат соответствующим маленьким.

Таким образом, сторона квадрата будет постоянно сокращаться в 2 раза, следовательно потребуется $log_2(N) < 30$ итераций. Делая по 4 запроса на каждой итерации, получим

Решение 2. Сделаем запросы в два соседних угла исходного квадрата. Пусть ответы на запросы — это a и b. Тогда имеем треугольник с вершинами в точках из запросов и искомой точке, длины сторон которого N, a, b. Исходя из этого, можем найти площадь треугольника, его высоту и т. д. Также надо аккуратно обработать вещественный тип данных. Для этого решения потребуется 2 запроса.

Решение 3. Используем тернарный поиск. Сначала зафиксируем одну из координат и тернарным поиском найдем вторую координату, при которой расстояние до Николая минимально. Теперь, зная вторую координату, еще одним тернарным поиском ищем первую, при которой расстояние минимально. Это и будет ответом. Асимптотика тернарного поиска $log_{\frac{3}{2}}(N)$, что на максимальном тесте означает необходимость выполнения менее 50 итераций. На каждой итерации используется 2 запроса. Тогда для двух тернарных поисков итоговое количество запросов равно $4 \cdot log_{\frac{3}{2}}(N) < 200$

Разбор задачи «Парад планет»

Для каждой планеты определяется минимальное расстояние в градусах, которое ей нужно пройти. Затем находим минимум из двух значений. Первое — модуль разности угла планеты и заданного угла, второе — 360 минус модуль разности угла планеты и заданного угла.

После этого расстояние в углах делится на 360 и умножается на длину окружности — $2 \cdot \pi \cdot R$, полученное число прибавляется к ответу.

Разбор задачи «Смешарики в космосе»

Для решения задачи необходимо в первую очередь выбирать растения, вырабатывающие наибольшее количество кислорода. Отсортируем растения по убыванию количества вырабатываемого кислорода. Будем вычитать из T количество вырабатываемого растением кислорода с учетом того, сколько таких растений осталось. Как только T стало меньше либо равно нулю, выводим количество сделанных вычитаний. Если мы вычли все растения, но T осталось больше нуля, выводим -1

Разбор задачи «Сильнейший Герой»

Запишем результаты формул в отдельные переменные. Запустим двойной цикл по всем противникам. При победе обновим значения в переменных, если они больше, чем характеристики противника. Пометим, что этого противника мы победили, и занесем его позицию в ответ.

Если такое решение получает вердикт «Превышено ограничение времени» на интерпретируемых языках, то необходима оптимизация: если за внутренний цикл не удалось победить ни одного противника, необходимо вызвать оператор «break» или аналогичный ему.

Разбор задачи «Домашние питомцы»

Задача решается методом динамического программирования.

Необходимо перебрать все возможные случаи расстановки цифр в позициях номера. Так как заранее не известно, сколько вложенных циклов потребуется, чтобы каждым из циклов перебирать значения в конкретной позиции номера, можно использовать рекурсию. Рекурсия и будет возвращать ответ на задачу.

В рекурсию будут передаваться два параметра pos и $prev$, где pos — текущая позиция в номере (позиции будем нумеровать слева направо, начиная с 1), $prev$ — цифра, установленная на предыдущую позицию $pos - 1$.

В рекурсии необходимо перебрать все возможные варианты цифр, которые можно поставить в текущую позицию pos так, чтобы выполнялось условие $|prev - newprev| \leqslant K$, где $newprev$ — цифра, которую можно поставить в текущую позицию. Для первой позиции перебираются все цифры от 1 до 9. Если номер имеет длину один, то необходимо перебрать все цифры от 0 до 9.

Для очередной подходящей цифры new_prev нужно вызвать рекурсивную функцию, увеличив позицию на единицу и в качестве параметра $prev$ передать new_prev. Когда на некотором вызове рекурсии значение параметра pos станет больше заданной длины номера, то можно считать, что найден один из вариантов расстановки цифр в позициях, при котором номер считает подходящим. Тогда текущий вызов рекурсии возвращает результат, равный 1.

На каждой итерации рекурсии накапливается количество вариантов расставить цифры в номере, начиная с текущей позиции. Это количество представляет собой сумму всех результатов вызовов рекурсии для следующей позиции на текущей итерации. Результат должен быть взят по модулю $1000000007 (10^9 + 7)$.

Чтобы рекурсия не совершала несколько проходов для одинаковых значений pos и $prev$, промежуточные результаты суммирования можно сохранить в двумерном массиве dp, где первым измерением будет являться позиция цифры, вторым — цифра на предыдущей позиции. При новом вызове рекурсии нужно проверить, был ли посчитан результат для данной пары чисел pos и $prev$, и, если значение было посчитано, вернуть этот результат как $dp[pos][prev]$.