РАЗБОР НАИБОЛЕЕ СЛОЖНЫХ ЗАДАЧ ОБУЧАЮЩЕЙ ИНТЕРНЕТ-ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ

ТЕОРЕТИЧЕСКИЙ ТУР

Задача 4. «Плохие лотерейные билеты»

Решение задачи основано на методе динамического программирования. Пусть $D[i][j]$ – количество билетов, состоящих из i цифр и сумма цифр которых при делении на 13 даёт остаток j. База: $D[1][i]$=1, i=0..9, $D[1][i]$=0, i>9. Переход в динамике осуществляется следующим образом. Рассмотрим билеты длины i, сумма цифр которых при делении на 13 даёт остаток j. Как она могла быть получена? Если мы к билетам длины $i-1$ приписали 1, то нам подходят билеты длины $i-1$, дающие остаток $j-1$ при делении на 13. Рассуждая аналогичным образом, приходим к тому, что нам подходят все билеты длины $i-1$, сумма цифр которых при делении на 13 даёт остаток, равный $j-k$, где k=0..9. Таким образом, мы получаем следующую формулу: $D[i][j]=\sum_{k=0}^{9} D[i-1][j-k]$, i=2..n, j=0..12. Но, при $j<k$ мы получаем отрицательные индексы, чтобы избежать этого, заметим, что $j-k$ и $j-k+13$ дают одинаковые остатки при делении на 13. Таким образом, финальная формула принимает вид: $D[i][j]=\sum_{k=0}^{9} D[i-1][(j-k+13)\%13]$, i=2..n, j=0..12. Взятие по модулю $j-k+13$ необходимо для случая j>k, так как в этом случае число $j-k+13$ будет больше 12. Ответом на задачу будет число $D[n][0]$.

Задача 5. «Двапалиндром»

Из условия можно заметить, что двапалиндром – строка, являющаяся палиндромом, и обе половины которой также являются палиндромами. Таким, образом, по сути, следует построить такую строку, что обе её половины равны и являются палиндромами, изменив при этот минимальное количество символов. Будем считать, что символы в нашей строке S нумеруются с 0 до $S.size()-1$, $S.size()$ – размер введенной строки S. Зафиксируем четыре указателя: p[0]=0, p[1]=$S.size()/2-1$, p[2]=$S.size()/2$, p[3]=$S.size()-1$. Будем двигать p[0], p[2] вправо, p[1], p[3] - влево. Исходя из определения двапалиндрома, символы, стоящие на этих местах, должны быть равны. Заменим их в строке S на наиболее часто встречающийся символ среди символов S[p[i]], i=0..3. Если таких символов несколько, можно выбрать любой.

Задача 6. «Сортировка»

Приведем решение, работающее в худшем случае за *2n* обменов отрезков местами. Пусть числа *1..k-1* стоят на своих местах. Передвинем число *k* максимум за 2 хода. Оно либо в первой половине ещё не отсортированной части перестановки, и тогда нам понадобится лишь один ход, либо во второй половине, и тогда нам понадобится 1 ход, чтобы передвинуть его в первую половину.

ПРАКТИЧЕСКИЙ ТУР

Задача 1. «Счастливые билетики по-ярославски»

Рассмотрим операцию подсчёта суммы цифр по-ярославски половины билета. Нетрудно заметить, что это операция подсчёта цифрового корня числа. Цифровой корень числа – это остаток деления числа на 9, если число не делится на 9 либо равно 0, и 9 в противном случае. Для удобства будем считать, что пока билета 00…0 (*2n* нулей) в рулоне нет. Теперь возникает вопрос: сколько всего существует *n*-значных чисел, цифровой корень которых равен *k*? Их 11…1 (*n* единичек). Таким образом, *2n*-значных билетов, у которых цифровой корень первой половины билета равен цифровому корню второй половины билета, всего может быть (11…1 (*n* единичек))^2 Всего цифровой корень принимает 9 значений. Умножим полученное ранее количество билетов на 9 и прибавим 1 (так как мы не учли билет 00…0 (*2n* нулей)). Не забываем проводить все наши вычисления по модулю 10^9+7.

Задача 2. «Замощение улицы»

Задача решалась методом динамического программирования. Рекуррентная формула следующая:

База динамического программирования: a[0]=0; a[1]=1; a[2]=1; a[3]=5;

Итоговая формула: a[n]=a[n-1]+5*a[n-2]+a[n-3]-a[n-4].

Так как в формуле присутствует вычитание, а вычисления должны проводиться по модулю, формула трансформируется в следующую:

a[n]=(a[n-1]+5*a[n-2]+a[n-3]-a[n-4]+MOD)%MOD, где $MOD=10^9+7$.

Задача 3. «Упаковка печенья»

Будем поддерживать 2 очереди с приоритетом, приоритетом будет являться диаметр печенек. Назовём одну *minPQ*, а другую *maxPQ*.

Если поступает печенька диаметра *d*, сравним её диаметр с диаметром наименьшей печеньки из *minPQ*. Если она строго больше, добавим *d* в *minPQ*. Если размер *minPQ* стал больше чем (размер *maxPQ*)+1, перенесем минимальный элемент из *minPQ* в *maxPQ*. Если же d меньше либо равна печеньки с наименьшим диаметром, добавим d в *maxPQ*. Если размер *maxPQ* стал строго больше размера *minPQ*, перенесем максимальный элемент из *maxPQ* в *minPQ*.

Если поступил запрос на упаковку, поступаем следующим образом:

1. Забрать минимальный элемент *m* из *minPQ*;
2. Если размеры *minPQ* и *maxPQ* различаются, перенесем максимальный элемент из *maxPQ* в *minPQ*.