

# Начальное значение суммы *s* считаем равным 0. В цикле просматриваем значения всех чисел, обозначатощих возраст участников группы. Если возраст меньше 11, то значение суммы s не изменяется, и мы переходим к просмотру следующего числа. Если возраст меньше 19, значение суммы s увеличиваем на 50 (py6.), в противном случае увеличиваем *s* на 100 (py6.).

for i := 1 to п do begiп read(d);

if *d I I* then continue

if *d i 9* then Inc(s, 50) else Inc(s, 60);

# end;

write(s),’

(Автор задачи и решений — *Еиндер М.И. )*

ПEPBOE РЕШЕНИЕ. Воспользуемся жадным алгоритмом. Из данного числа т вычтем нпп9олошпfі возможный факториал ii! и к полученной разности применим те же рассуждения. Другими словами, из полученной разности снова вычтем наибольший факториал, и так далее.

Количество вычитаемых факториалов и будет искомым ответом в задаче. Реализация этого алгоритма несложная. Корректность жадного алгоритма следует ив неравенства

1- 1! + 2 2! + .. . + (п — 1) (п — 1)! < п! .

Действительно, если для заданного числа т выполняется неравенство п! й х < (п + 1)!, то представление х в виде суммы наименьшего количества факториалов обязательно содержит слагаемое п!. В противном случае, как следует из приведённого неравенства, сумма «остальных» факториалов будет меньше *х.* (Если факториал k! входит с коэффициентом с, > k, то слагаемое с, *h! —— (h + I)! +* (с, — *h — I)h! можно* заменить суммой меньшего числа слагаемых.)

# BTOPOE РЕШЕНИЕ основано на применении *факторианьной* спсіпежЬі cчпcлечпл. В этой системе счисления основанием служит последовательность натуральных чисел: 1!, 2!, ... , #!. В факториальной системе натуральное число т представляется единственным образом в виде

*х ——* с, 1! + C2 ’ 2! + . . . + с, u!, где 0 с, L.

# Из этого представления следует, что *наименьшее* количество факториалов, при сложении которых получается заданное т, равно сумме «цифр» с, в записи числа х в факториальной системе счисления, то есть равно

S - С; + C2 + :. + С .

Осталось научиться переводить число в факториальнуіо систему. Оказывается, для этого не обязательно вьшислять факториалы натуральных чисел. Действительно, каждое из чисел *h!* делится на все натуральные числа от 1 до *h,* поэтому слагаемые вида с, k! делятся на 2 при ліобом k й 2. Значит, число с, равно остатку от деления х на 2. Разделив нацело х на 2, то есть отбрасывая слагаемое с, 1!, применим те же рассуждения к числу х div 2. Теперь число cc равно остатку от деления х div 2 на 3. Этот процесс продолжаем до тех пор, пока частное от деления на очередное число *h* не станет уже равным нуліо. Ниже приведен фрагмент кода на языке Delphi, реализующего этот алгоритм.

s := 0; L := 2;

while (п > 0) do begiп

s := s + т mod L; т : = х div й; me(й);

end;

write(s),

(Автор задачи и решений — *Kuitдep М.И. )*

В десятичной записи ліобого палиндрома равноудаленные от концов цифры совпадаіот. Из h цифр заданного набора можно будет составить палиндром только в том случае, когда каждая цифра от 0 до 9 входит в этот набор *чётиое* количество раз (возможно, ни разу), или же *ровно* обно цифра входит нечётное число раз, а все остальные — чётное число. Например, последнему условиіо удовлетворяіот числа 1112002 и 2200550, из которых можно составить палиндромы 2011102 и 2050502. В записи этих палиндромов цифра, входящая *нечётное* число раз, находится ровно посередине. Исклк›чение составляют числа, которые содержат чётное число нулей и ровно одну ненулевуіо цифру. К ним относится, в частности, число 10000, из которого составить палиндром невозможно.

Теперь составим *наименьший* палиндром. Подсчитаем количество вхождений *d[m]* каждой цифры m от 0 до 9. Наименьшуіо ненулевуіо цифру поставим на первое место палиндрома. Сформируем первуіо «половину» палиндрома. Сначала выводит на печать каждуіо из цифр m в точности *d[In]* div 2 раз. «Середина» наименьшего палиндрома заполняется цифрой m, которая входит в запись исходного числа *нечётиое* число раз. «Вторая» половина палиндрома получается симметрично «первой». Реализация этого алгоритма несложная, но требует умения работать с отдельными цифрами-символами исходного набора. Сложность алгоритма — О(п) операций.

(Поити фо'іыt/іор, решения задаии— *Киндер М.И. )*

Обозначим через *sum(t)* сумму расстояний от точки t до всех точек многоугольника, а через *F(X, У) —* функцию двух точек А и У, которая с помощыо тернарного поиска нп *отрезке ХУ* находит наибольшее значение суммы расстояний snin(f).

РЕШЕНИЕ ПОДЗАДАЧИ 1. Пусть А— искомая точка в треугольнике *ABC.* Рассмотрим одну из сторон треугольника, например, AB. Начиная от точки *А,* будем двигаться по направлениіо к точке*В в роль* отрезка *AB. Для* каждой промежуточной точки Х отрезка *AB* будем вьшислять значение функции *F(X,С) ——* max sum(t) на отрезке CC. Промежуточные точки Х тоже будем выбирать с помощыо тернарного поиска по отревку AB. Среди найденных вначений функции *F(X,С),* конечно, будем напоминать только наибольшее. Таким образом, фактически будет найдено наибольшее значение sum(I) в треугольнике ABC и на его границе. Сложность алгоритма — О(£ *i/2),* где *h* — количество запросов (число районов), *і/* — количество итераций для вычисления наибольшего значения функции *F(X, У)* вдоль выбранного отрезка.

РЕШЕНИЕ ПОДЗАДАЧИ 2. Идеіо решения подзадачи 1 можно применить в более общей ситуации, когда количество вершин многоугольника не превышает 20. Выберем вершину *А* многоугольника и для каждого треугольник а А А;А; + (2 й i й u — 1) найдем наибольшее значение *ф* ункцпп *sum(t) (cc.* подзадачу 1). Сложность алгоритма — O(k n2 *i f2),гдe* п количество вершин многоугольника.

РЕШЕНИЕ ПОДЗАДАЧИ 3. Попробуем искать наибольшее значение функции sum(f) только вдоль границы многоугольнігка. Другими словами, на каждой стороне многоугольнггка *А,А, , (I i п, А + —— А )* найдём максимум функции suin(I), а затем среди них выберем самое наибольшее значение. Сложность алгоритма — O(L п2 *it).*

# РЕШЕНИЕ ПОДЗАДАЧИ 4. Наибольшее значение функции *sum(t)* достигается в одной из вершин многоугольника. Доказательство этого факта можно провести с помощьто несложных геометрических рассуждений, и мы его здесь опускаем. Таким образом, для полного решения задачи достаточно перебрать все вершины многоугольника и для каждой из них вычислить значение функции *sum(t),* а потом выбрать среди них наибольшее. Сложность алгоритма — O(L 2)

(Легенда составлена студентами ІїФУ, автор з:ідачи и решений— *би чдер М И.)*

# РЕШЕНИЕ ПОДЗАДАЧИ 1. Для «небольших» значений k и m (2 й k й 3, 1 й in й 100) все требуемые наборы можно найти несложным перебором. Такое решение проходит тесты на 30 баллов.

РЕШЕНИЕ ПОДЗАДАЧИ 2. Поскольку требуется подсчитать только количество требуемых наборов чисел, можно применить комбинаторные рассуждения. Пусть пз — наименьшее общее кратное всех чисел нашего набора, и пусть m — р 1 р2 ' .. р ' — разложение m на простые множители.

Тогда условито задачи удовлетворятот все наборы чисел п, — р,""'p2 ""' ... р,"‘"', 1 й *i* й fi, у которых неотрицательные показатели o[i,s] степеней простых чисел *р,* не превосходят ct, и для которых

выполняіотся условия

max(n[1,s], o[2,s], ... , n[3,s]) = п, ,

для всех 1 й s й *г.* Подсчитаем число наборов по каждому показателіо о отдельно. Количество наборов, которые удовлетворяіот условиям 0 й ct, й п для всех i от 1 до k, равно



В самом деле, число о, может принимать п + 1 значений это все неотрицательные числа от 0 до о, и по правилу произведения количество таких наборов равно (п + 1)'. Осталось искліочить из этого множества наборы, для которых max(n1. 2. «) < п. Для таких наборов 0 й п, й о — 1 для всех i от 1 до k, поэтому их количество равно п\*. Значит, количество требуемых наборов равно (п + 1)' — п'. Теперь по правилу произведения число наборов по асеж показателям

# равно:

*L(т, k) —— [(а, + )k gk]*

Например, для m = 10 = 21 51 и L = 2 получаем

*L(1* О, 2) *= [(1+* 1)2 }2) . 1 + 1)2 }2] ш

упорядоченных наборов из двух чисел, у которых HOW равен 10. Алгоритмическая сложность такого решения — *О(Fact(т) +* k).Taкoe решение проходит тесты ещё на 35 баллов.

РЕШЕНИЕ ПОДЗАДАЧИ 3. Для чисел k из диапазона [1; 10' ] для быстрого вьшисления (о + 1)a и о\* используем бинарный алгоритм возведения в степень. Алгоритмическая сложность такого решения — *О(Fact(т) +* logk).

*Председатель жюри М.И. Киндер*