

Всероссийская олимпиада школьников по информатике

Вологодская область

II (муниципальный) этап

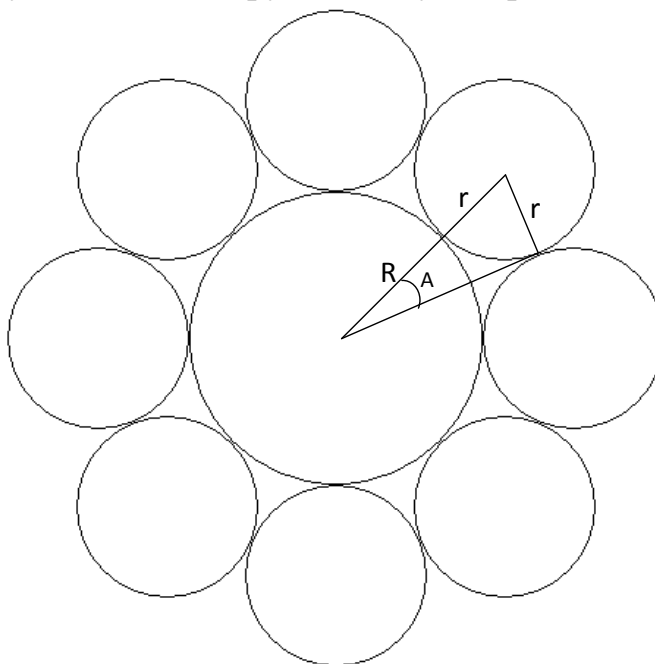
2016-2017 учебный год

9 - 11 классы

**Методические рекомендации по разбору задач**

**Задача 1. Олимпийские медали (100 баллов)**

Рассмотрим изображенный на рисунке прямоугольный треугольник. Здесь  $R$  – известный радиус центрального круга,  $r$  – неизвестный радиус остальных кругов,  $A$  – угол, равный  $\pi/N$ .



В этом треугольнике  $\sin(\pi/N) = r/(r+R)$ . Выразим отсюда  $r$ :  $r = R/(1/\sin(\pi/N) - 1)$ . Суммарная площадь равна  $\pi R^2 + N \cdot \pi r^2$ .

Поясним ещё смысл фразы в условии задачи: *"гарантируется, что во всех тестах задачи точное значение площади отличается от ближайшего к нему целого числа не более чем на 0.49"*. Это примечание позволяет участникам быть уверенными, что в вычисленной ими площади после десятичной точки не идут цифры вида .4999... или .5000... . Из-за погрешности вещественных вычислений такое число могло бы округлиться не в ту сторону по сравнению с ответом жюри.

### Пример решения на Free Pascal

```
var
  R, N: integer;
  r1, S: Double;
begin
  read(R, N);
  r1 := R / (1 / sin(PI / N) - 1);
  S := PI * sqr(R) + N * PI * sqr(r1);
  writeln(s:0:0);
end.
```

Возможны и другие подходы к решению. Например, решение этой задачи через другой треугольник и теорему косинусов можно найти в материалах к задаче (файл *circles\_ia.pas*).

### Задача 2. Школьный концерт (100 баллов)

Пусть  $N$  – количество дисков,  $K$  – количество компьютеров,  $T_i$  – время записи диска на  $i$ -м компьютере.

Для решения задачи воспользуемся бинарным поиском. В качестве левой границы *left* можно взять единицу, в качестве правой границы *right* – время записи на самом быстром компьютере, умноженное на  $N$ .

Положим  $mid = (left + right) \div 2$ . Проверим, можно ли записать все диски за время  $mid$ . Для этого пройдем в цикле по всем машинам и сосчитаем, сколько дисков можно записать на каждой машине за время  $mid$ , полученные результаты сложим. Если найденное число дисков оказалось меньше  $N$ , то нужно увеличить время, сдвинув левую границу к  $mid+1$ . В противном случае запомним текущий ответ и попытаемся ещё уменьшить время, сдвинув правую границу к  $mid-1$ .

Вычислительная сложность алгоритма составляет  $O(N \cdot \log(\min(T_1, T_2, \dots, T_N)))$ .

Стоит ещё обратить внимание на то, что ответ может не поместиться в 32-битный целый тип. Поэтому нужно использовать 64-битный тип (в Паскале это `int64`).

### Пример решения на Free Pascal

```
uses math;

const
  NMAX = 100000;
  KMAX = 100000;

var
```

```

t: array[1..KMAX] of longint;
n, k, i: longint;
left, right, mid, ans: int64;

function canWrite(time: int64): boolean;
var
  i, count: longint;
begin
  count := 0;
  for i := 1 to k do
    count := count + time div t[i];
    canWrite := count >= n;
  end;

begin
  read(n, k);
  left := 1;
  right := maxLongint;
  for i := 1 to k do begin
    read(t[i]);
    right := min(right, t[i]);
  end;

  right := right * n;
  while (left <= right) do begin
    mid := (left + right) div 2;
    if canWrite(mid) then begin
      ans := mid;
      right := mid - 1;
    end else
      left := mid + 1;
    end;
  writeln(ans);
end.

```

Возможны и другие подходы к решению. Пример решения задачи другим способом можно найти в материалах к задаче (файл *con-fm.cpp*).

Подзадачу 1 можно решить проще – например, последовательно увеличивать время на единицу и проверять каждый раз, удаётся ли записать все диски.

### Задача 3. Возраст (100 баллов)

Пусть  $N$  – это количество школьников,  $K$  – суммарный возраст, дату для которого нужно найти.

Установим текущую дату, например, равной 01.01.2011. Она лежит после дат рождения всех школьников, но ещё заведомо не

достигает искомой даты (так как в условии задачи сказано, что  $20 \cdot N \leq K$ ).

Прочитаем входные данные и заполним матрицу  $C$  размера  $12 \times 31$ , где  $C[i][j]$  – это количество школьников, у которых дата рождения приходится на  $j$ -й день  $i$ -го месяца. Также сразу посчитаем в переменной  $sum$  сумму возрастов всех школьников на текущую дату.

Заметим, что если к текущей дате добавить  $d$  лет, то суммарный возраст увеличится на  $d \cdot n$ . Определим такое максимальное (или близкое к максимальному) значение  $d$ , чтобы не перескочить за искомую дату. Значение  $d$  можно взять, например, таким:

$$d = (K - sum - 1) \operatorname{div} N$$

Примечание: можно не вычислять  $d$  формулой, а увеличивать его значение в цикле, пока суммарный возраст не превысит  $K$ . Поскольку по условию  $K \leq 1000 \cdot N$ , то число итераций будет небольшим.

Увеличим год в текущей дате на  $d$ , а переменную  $sum$  – на  $d \cdot N$ . Теперь будем в цикле увеличивать текущую дату на один день, смотреть по матрице  $C$ , сколько дней рождения пришлось на этот день, и увеличивать переменную  $sum$ . Отметим, что если текущая дата оказалась равна 1 марта невисокосного года, то нужно ещё добавлять школьников, родившихся 29 февраля. Как только переменная  $sum$  станет больше или равна  $K$ , ответ найден.

Вычислительная сложность такого решения составляет  $O(N)$ , причём большую часть времени занимает ввод входных данных и заполнение матрицы  $C$ .

Для решения подзадачи 1 можно использовать более простой подход – например, занести все даты в одномерный массив и организовать два вложенных цикла: внешний цикл увеличивает текущую дату на один день, а внутренний пробегается по всем датам и считает, сколько дней рождения пришлось на этот день.

### Пример решения на Free Pascal

```
uses SysUtils;

const
  NMAX = 1000000;
  dm: array[1..12] of longint = (31, 28, 31, 30, 31, 30, 31,
    31, 30, 31, 30, 31);

type
  Date = record
    d, m, y: longint;
  end;
```

```

function isLeap(y: longint): boolean;
begin
    isLeap := (y mod 4 = 0) and (y mod 100 <> 0)
              or (y mod 400 = 0);
end;

function mDays(m, y: longint): longint;
begin
    if m<>2 then mDays := dm[m]
    else
        if isLeap(y) then mDays := 29 else mDays := 28;
    end;
end;

procedure incDate(var dt: Date);
begin
    with dt do begin
        inc(d);
        if d > mDays(m, y) then begin
            inc(m);
            d := 1;
            if m > 12 then begin
                inc(y);
                m := 1;
            end;
        end;
    end;
end;

function Age(d1, d2: Date): longint;
begin
    if (d1.m < d2.m) or (d1.m = d2.m) and (d1.d <= d2.d) then
        Age := d2.y - d1.y
    else
        Age := d2.y - d1.y - 1;
    end;
end;

var
    n, k, sum, i, j, delta: longint;
    dt, cur: Date;
    s: String[10];
    c: array[1..12, 1..31] of Longint;

begin
    readln(n, k);
    sum := 0;
    cur.d := 1; cur.m := 1; cur.y := 2011;
    for i := 1 to 12 do
        for j := 1 to 31 do
            c[i][j] := 0;
        end;
    for i := 1 to n do begin

```

```

readln(s);
dt.d := (ord(s[1])-ord('0')) * 10 + ord(s[2])-ord('0');
dt.m := (ord(s[4])-ord('0')) * 10 + ord(s[5])-ord('0');
dt.y := (((ord(s[7])-ord('0')) * 10
+ ord(s[8])-ord('0')) * 10 + ord(s[9])-ord('0')) * 10
+ ord(s[10])-ord('0'));
sum := sum + Age(dt, cur);
inc(c[dt.m][dt.d]);
end;
delta := (k - sum - 1) div n;
sum := sum + delta * n;
cur.y := cur.y + delta;
while (sum < k) do begin
    incDate(cur);
    sum := sum + c[cur.m][cur.d];
    if (cur.d=1) and (cur.m=3) and (not isLeap(cur.y)) then
        sum := sum + c[2][29];
end;
writeln(Format('%.2d.%.2d.%.2d', [cur.d, cur.m, cur.y]));
end.

```

#### Задача 4. Конкурс капитанов (100 баллов)

Нам дано поле размера  $M \times N$ . Вначале докажем следующее вспомогательное утверждение: для любых  $2 \cdot \max(M, N)$  крестиков на поле обязательно найдётся параллелограмм с вершинами в клетках, помеченных какими-то из этих крестиков.

Доказательство. Для удобства пусть  $M \leq N$ . Будем рассматривать только такие параллелограммы, у которых две стороны горизонтальны (параллельны оси  $X$ ). Чтобы найти такой параллелограмм, нужно найти две строки, в каждой из которых найдётся два крестика с одинаковыми расстояниями между ними.

Выберем на поле любые  $K=2 \cdot N$  крестиков. Можно считать, что на поле нет строк менее чем с двумя крестиками в каждой (если такие строки есть, то просто выбросим их из поля, и тогда количество оставшихся выбранных крестиков по-прежнему будет превышать количество оставшихся строк как минимум вдвое).

Пусть в первой строке стоит  $k_1$  крестиков, во второй -  $k_2$ , и так далее, в  $M$ -й строке -  $k_m$ . При этом  $k_1 + k_2 + \dots + k_m = 2 \cdot N$ .

В первой строке будет не менее  $(k_1-1)$  различных расстояний между крестиками (от первого до второго, от первого до третьего и т.д.). Во второй строке - не менее  $(k_2-1)$  различных расстояний, и так далее. Допустим, что на поле нет параллелограмма с вершинами в каких-то из этих крестиков. Тогда любые два расстояния из всей

совокупности должны быть различны. Общее количество расстояний равно:

$$(k_1-1) + (k_2-1) + \dots + (k_m-1) = (k_1 + k_2 + \dots + k_m) - M = 2 \cdot N - M \geq N$$

У нас получилось не менее  $N$  различных расстояний. Но все расстояния между крестиками в одной строке могут быть только от 1 до  $(N-1)$ , то есть  $N$  различных расстояний получить невозможно. Отсюда следует (принцип Дирихле), что найдутся такие две строки, в которых найдутся две пары крестиков с одинаковым расстоянием между ними, которые и образуют параллелограмм.

Заметим, что если на поле имеется меньше, чем  $2 \cdot \max(M, N)$  крестиков, то параллелограмм на поле, конечно, тоже может присутствовать (причём, возможно, без горизонтальных сторон).

Алгоритм решения выглядит следующим образом: возьмём первые  $2 \cdot \max(M, N)$  крестиков и занесём их координаты в одномерный массив из записей (либо в два отдельных массива для  $x$  и  $y$ ). Если же крестиков на поле меньше, чем  $2 \cdot \max(M, N)$ , то занесём координаты всех крестиков.

Далее с помощью трёх вложенных циклов, проходящих по этому массиву, переберём варианты трёх возможных вершин параллелограмма. Зная координаты трёх вершин, вычислим координаты четвёртой. Если эти координаты не выходят за пределы поля и в клетке стоит крестик, то кандидат в параллелограммы найден. Осталось только проверить, что параллелограмм невырожденный – для этого можно, например, сосчитать его площадь и сравнить с нулём.

Вычислительная сложность алгоритма составляет  $O(\max(M, N)^3)$ . При максимальных входных данных суммарное число итераций самого внутреннего цикла составит около 20 миллионов.

### Пример решения на Free Pascal

```
uses Math;

const
  MAXSIZE = 255;

type
  Coord = record
    x, y: longint;
  end;

var
  a: array[1 .. MAXSIZE] of String;
  c: array[1 .. MAXSIZE * 2] of Coord;
  ySize, xSize, i, j, k, n, x4, y4: longint;
```

```

procedure FillCoord;
var
  y, x: longint;
begin
  n := 0;
  for y := 1 to ySize do
    for x := 1 to xSize do
      if a[y][x] = 'x' then begin
        inc(n);
        c[n].y := y;
        c[n].x := x;
        if n = 2 * max(xSize, ySize) then
          exit;
        end;
      end;
    end;
  end;

begin
  readln(ySize, xSize);
  for i := 1 to ySize do
    readln(a[i]);
  FillCoord;

  for i := 1 to n - 2 do
    for j := i + 1 to n - 1 do
      for k := j + 1 to n do begin
        x4 := c[i].x + c[k].x - c[j].x;
        y4 := c[i].y + c[k].y - c[j].y;
        if (x4 >= 1) and (x4 <= xSize)
          and (y4 >= 1) and (y4 <= ySize) then
          if (a[y4][x4] = 'x') and
            ((c[i].x - c[j].x) * (c[k].y - c[j].y) - (c[k].x - c[j].x) *
              (c[i].y - c[j].y) <> 0) then begin
            writeln(c[i].y, ' ', c[i].x);
            writeln(c[j].y, ' ', c[j].x);
            writeln(c[k].y, ' ', c[k].x);
            writeln(y4, ' ', x4);
            halt(0);
          end;
        end;
      end;
    end;
  writeln(-1);
end.

```

Возможны и другие способы решения. Например, в материалах к задаче имеется файл *cap-fm.cpp* – решение на языке C++, использующее другой алгоритм.

Рассмотрим более простые варианты решения подзадач 1 и 2. Для подзадачи 1 можно просто перебрать все возможные комбинации



четырёх крестиков на поле (не забывая, что для каждой четырёх вершин возможен ещё и разный порядок их обхода). Конечно, каждый внутренний цикл должен запускаться, только если во внешнем цикле мы стоим на крестике.

Для подзадачи 2 можно перебирать все возможные комбинации трёх крестиков прямо на поле (без предварительного занесения координат в отдельный массив) и вычислять позицию, где должен находиться четвёртый крестик.

В материалах олимпиады можно найти также примеры других решений задач на языках C++ и Java.