Разбор задач муниципального этапа всероссийской олимпиады школьников для 7-9 классов 2017-18 учебного года

Задача 1. Древняя Курская земля

Решение задачи сводится к поиску :

1) минимального и максимального элемента среди элементов X1, X2, X1+W1, X2+W2. Разность между максимумом и минимумом есть одна сторона огораживаемого участка;
2) минимального и максимального элемента среди элементов Y1, Y2, Y1+H1, Y2+H2. Разность между этими максимумом и минимумом есть другая сторона огораживаемого участка;
3) после этого вычислить длину забора не составляет труда.

Возможный вариант решения задачи:

```
var X1,Y1,W1,H1,X2,Y2,W2,H2,x3,x4,y3,y4,d,v,
    minx,maxx,miny,maxy: integer;
begin
  assign(input,'Kursk.in');  reset(input);
  assign(output, 'Kursk.out');  rewrite(output);
  readln(X1,Y1,W1,H1);
  readln(X2,Y2,W2,H2);
  x3:=X1+W1; X4:=X2+W2;
  Y3:=Y1+H1; Y4:=Y2+H2;
  IF x1>x2 then begin minX:=x2; MaxX:=X1 end
           else begin minX:=x1; MaxX:=X2 end;
  IF x3<minx then minX:=x3
           else if x3>maxX then MaxX:=x3;
  IF x4<minx then minX:=x4
           else if x4>maxX then MaxX:=x4;
  d:=maxx-minx;
  IF y1>y2 then begin miny:=y2; Maxy:=y1 end
           else begin miny:=y1; Maxy:=y2 end;
  IF y3<miny then miny:=y3
           else if y3>maxy then Maxy:=y3;
  IF y4<miny then miny:=y4
           else if y4>maxy then Maxy:=y4;
  v:=maxy-miny;

  writeln(2*d+2*v);
close(input);
close(output);
end.
```

Задача 2. Ванина задача

Сначала определим, сколько целых страниц будет заполнено, если напечатать первые N строк – вычислим частное от целочисленного деления N на K. Это частное будет равно искомому количеству страниц, на которых напечатано по K строк.
Если остаток от целочисленного деления равен нулю, то строка № N будет последней на странице с номером N div K. Если остаток не равен нулю, то эта строка будет уже на следующей странице, номер которой (N div K) + 1. Её порядковый номер будет равен остатку от целочисленного деления N на K – N mod K. Один из вариантов программы:

```
readln(K,N);
if N mod K = 0 then write(N div K, ' ', K)
else write (N div K + 1, ' ', N mod K);
```

Оператор условия можно не использовать:

```
readln(K,N);
write((n-1) div k) + 1, (n-1) mod k + 1);
```

Задача 2. Магические заклинания

Запишем буквы, обозначающие флажки, в массив a. Сначала определим функцию, которая будет проверять симметричность ли последовательности a[1], a[2], …, a[k]. Будем сравнивать числа a[1] и a[k], a[2] и a[k-1], и т. д. до a[k div 2] и a[k+1-(k div 2)].

a[1]	a[2]	…	a[k div 2]	a[k+1-(k div 2)]	…	a[k-1]	a[k]

Если в некоторой паре числа неравны, то последовательность симметричной не является:

```
function simm (x:integer):boolean;
var i:integer;
begin
  simm:=true;
  for i:=1 to x div 2 do
  if a[i]<>a[x+1-i] then
  begin
    simm:= false;
```

```
    exit;
  end;
end;
```

Алгоритм решения задачи следующий.

В худшем случае нужно будет дописать N-1 чисел. Например, если последовательность 1 2 3 … N-1 N, то кратчайшая симметричная последовательность будет: 1 2 3 … N-1 N N-1 … 3 2 1.

В лучшем случае последовательность уже симметрична и ничего делать не нужно. Учитывая это все, ответ в задаче – это число от 0 до N-1.

Один из алгоритмов решения задачи. Сначала с помощью функции Simm проверяем, является ли заданная последовательность симметричной. Если нет, проверяем, достаточно ли будет добавить только одно число. При этом необходимо в конец дописать число a[1]: a[N+1]:=a[1];

Теперь опять функцией simm проверяем, получилась ли симметричная последовательность. Если нет, то пытаемся дописать в конец исходной последовательности два числа:

a[N+2]:=a[1]; a[N+1]:=a[2]

И т. д., пока функция simm на очередном шаге не выдаст true.

Возможная реализация алгоритма:

```
 [1] for i:=0 to N-1 do
     begin
 [2]   for j:=1 to i do
 [3]     a[N+j]:=a[i+1-j];   //Добавляем в конец i чисел
 [4]   if simm(N+i) then
       begin
 [5]     writeln(i); //вывод количества добавленных чисел
 [6]     for j:=1 to i do
 [7]     write(a[N+j],' ');  //вывод добавленных чисел
 [8]     exit;
 [9]   end;
[10] end;
```

При такой реализации алгоритма выполняются лишние операции: мы дописывая в конец те же числа, что стоят в начале последовательности и потом сравниваем эти числа в функции simm. Если проверять только числа, которые входят в исходную последовательность, то можно оптимизировать алгоритм.

Для этого в цикле внутри функции simm начинаем проверять числа не с a[1], а с числа, которое сравнивается с числом a[N], то есть с числа a[k+1-N]:

```
for i:=k+1-N to k div 2 do ...
```

Тогда в основной программе не нужно дополнять массив новыми числами: строки [2] и [3] можно удалить, а строку [7] заменить такой: write(a[i+1-j],' ');

Заметим, что согласно условию задачи, максимально возможная длина симметричной строки - 199. Это определяет размер массива a.

Задача 4. Наибольшее произведение

В задаче нам необходимо найти такие три числа, произведение которых максимально. То есть, какие бы другие три числа мы не выбирали, их произведение будет всегда меньше или равно максимальному. Теперь проанализируем, какие именно числа в последовательности могут давать максимальное произведение. Самое очевидное – три максимальных числа последовательности. Для последовательностей с неотрицательными элементами это действительно так.

Рассмотрим пример:

3	1	5	0	***9***	4	***6***	2	***6***	6

Когда все числа в последовательности неотрицательны, то максимальное произведение дадут именно три максимальных элемента. В нашем примере это `9*6*6=324`.

Остается понять, как искать три максимальных элемента – «тройной максимум». В нашем примере первый максимум = 9, второй (следующий по значению) = 6, третий = 6. Обратите внимание, что максимумы могут совпадать по значению.

Можно применить такой алгоритм. Пусть в переменных max1, max2, max3 хранятся первый, второй и третий максимум соответственно. Присвоим им начальные значения, равные 0. В процессе работы программы они будут либо улучшены.

Воспользуемся идеей «проталкивания сверху вниз» очередного элемента в текущие три максимума. Пусть в нашем примере мы уже просмотрели четыре элемента последовательности и правильно заполнили переменные max1, max2 и max3.

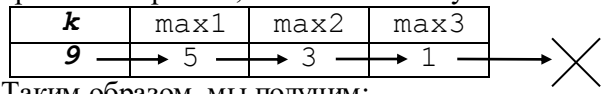
max1	Max2	max3

5	3	1

Читаем очередное число последовательности ***k*=9,** сраваем его с переменной max1.

```
if k > max1 then
 begin
      max3 := max2;
      max2 := max1;
      max1 := k;
 end
```

Поскольку 9>5, то произведем «проталкивание» нового максимума – запишем ***k*** в max1, а в max2 – старое значение max1, которое было больше или равно max2. В max3 запишем старое значение max2, прежнее значение max3 при этом теряется, оно больше не нужно.



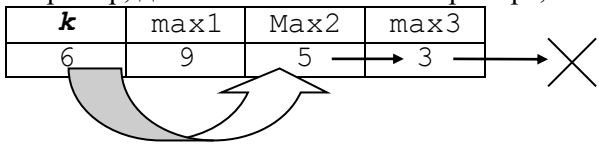
Таким образом, мы получим:

max1	max2	max3
9	5	3

Если окажется, что ***k***≤max1, тогда следует его сравнить с max2.

```
if k > max2 then
 begin
      max3 := max2;
      max2 := k;
 end
```

Например, для 7 элемента нашего примера, ***k***=6:



Мы не изменяем max1, вместо max2 записывается ***k***, а на место max3 становится прежний max2.

В противном случае сравнивается ***k*** и max3.

```
if k > max3 then max3 := k;
```

Можно реализовывать не «проталкивание сверху вниз», а «проталкивание снизу вверх». Программная логика в этом случае:

```
if k>max3 then max3 := k;
if k>max2 then
```

```
        begin
            max3 := max2;
            max2 := k;
        end;
      if k>max1 then
        begin
            max2 := max1;
            max1 := k;
        end;
```

Приведем полный текст решения данной задачи на языке PascalABC. Обратите внимание на то, что мы не создаем массив и не храним последовательность элементов в памяти. Поскольку задача решается в один проход по массиву, то для нашего алгоритма этого и не требуется – в каждый момент времени нам требуется знать только один текущий элемент последовательности. Мы обрабатываем его, а затем «забываем».

```
Var
  n:integer; //число элементов в последовательности
  max1, max2, max3:integer; {первый, второй и третий максимумы в
последовательности}
   k : integer; {текущий элемент последовательности}
   i : longint;
begin
     assign(input, 'max.in');
     reset(input);
     assign(output, 'max.out');
     rewrite(output);
     max1 := 0; max2 := max1; max3 := max1;
     readln(n);
     {читаем по очереди все элементы последовательности}
     for i:=1 to n do
      begin
         read(k);
         if k > max1 then
           begin
             max3 := max2; max2 := max1; max1 := k;
           end
         else
         if k > max2 then
           begin
             max3 := max2; max2 := k;
           end
         else
```

```
         if k > max3 then
              max3 := k;
      end;

      writeln(max1*max2*max3);
      close(input);
      close(output);
end.
```