

Разбор задач муниципального этапа олимпиады по информатике

1. Лифт (все классы)

Тема: вывод формулы, разбор случаев

Сложность: простая

Нужно рассмотреть, движется ли лифт в том направлении, куда нужно Петру, будет ли он или пассажир лифта сходить на промежуточном этаже, и вывести формулы для этих вариантов. Существует 4 варианта: 1) пассажир едет на этаж, где находится Петр; 2) Петр едет в том же направлении, что и лифт, и лифт может посадить его по пути, Петр выходит раньше или одновременно с пассажиром лифта; 3) тоже что и 2, но пассажир выходит раньше Петра; 4) лифт сначала отвозит пассажира, затем отвозит Петра.

Пример реализации:

```
uses math;
var a,b,c,d:integer;
begin
  read(a,b,c,d);
  if d=a then
    writeln(abs(c-d)+1+abs(a-b)+1)
  else if not((a<b) xor (c<d)) and (a>=min(c,d)) and (a<=max(c,d)) then
    begin
      if (b>=min(c,d)) and (b<=max(c,d)) then
        writeln(abs(c-a)+1+abs(a-b)+1)
      else
        writeln(abs(c-a)+1+abs(a-d)+1+abs(d-b)+1);
    end
  else
    writeln(abs(c-d)+1+abs(d-a)+1+abs(a-b)+1);
end.
```

2. Алгоритм (7-9 класс)

Тема: реализация программы по схеме алгоритма

Сложность: простая

Пример реализации:

```
var s,p,i,j:integer;
begin
  read(s,p);
  for i:=0 to 1000 do
    for j:=i to 1000 do
      if (i+j=s) and (i*j=p) then
        begin
          writeln(i, ' ', j);
          exit;
        end;
    end;
end.
```

2. Классификация снежинок (10-11 класс)

Тема: быстрая сортировка

Сложность: средняя

1 и 2 подзадачи решаются с помощью алгоритма быстрой сортировки, реализация которого может найдена в пояснениях к нескольким задачам на сайте ipc.susu.ru. Для решения 3 и 4 подзадачи необходимо сначала выполнить перевод всех снежинок в стандартную форму: например, из 12 форм полученных в результате вращения и переворачивания можно взять форму, являющуюся лексикографически наибольшей. Затем необходимо отсортировать такие формы также лексикографически и найти количество соседних несовпадающих форм в этом упорядоченном массиве.

Пример реализации:

```
type info=record a:array[1..6] of integer; end;
{ лексикографическое сравнение }
function less(var x,y:info):boolean;
var i:integer;
```

```

begin
  for i:=1 to 6 do
    if x.a[i]<y.a[i] then
      begin
        less:=true;
        exit;
      end
    else if x.a[i]>y.a[i] then
      begin
        less:=false;
        exit;
      end;
    less:=false;
  end;
var p:array[1..100000] of info;
{ быстрая сортировка }
procedure qsort(l,r:integer);
var x,t:info;
    i,j:integer;
begin
  if l>=r then exit;
  x:=p[l+random(r-l+1)];
  i:=l;
  j:=r;
  while i<=j do
    begin
      while less(p[i],x) do inc (i);
      while less(x,p[j]) do dec (j);
      if i<=j then
        begin
          t:=p[i];p[i]:=p[j];p[j]:=t;
          inc(i); dec(j);
        end;
    end;
  qsort(l,j);
  qsort(i,r);
end;
var n,k,i,j,a:integer;
    t,m:info;
begin
  read(n);
  for i:=1 to n do
    begin
      for j:=1 to 6 do
        read(t.a[j]);
      m:=t;
      { поворачиваем 6 раз }
      for j:=1 to 6 do
        begin
          a:=t.a[1];
          for k:=1 to 5 do
            t.a[k]:=t.a[k+1];
          t.a[6]:=a;
          if less(m,t) then
            m:=t;
        end;
      { переворачиваем }
      for k:=1 to 3 do
        begin
          a:=t.a[k];
          t.a[k]:=t.a[7-k];
          t.a[7-k]:=a;
        end;
      { поворачиваем 6 раз }
      for j:=1 to 6 do
        begin

```

```

a:=t.a[1];
for k:=1 to 5 do
    t.a[k]:=t.a[k+1];
t.a[6]:=a;
if less(m,t) then
    m:=t;
end;
p[i]:=m;
end;
{ сортировка }
qsort(1,n);
{ сравнение соседних }
k:=1;
for i:=2 to n do
    if less(p[i-1],p[i]) then
        inc(k);
    writeln(k);
end.

```

3. Подпоследовательность (все классы)

Тема: модулярная арифметика

Частичные решения: полный перебор, частичные суммы

Сложность: ниже среднего

Рассмотрим частичные суммы элементов последовательности от ее начала до i -го элемента включительно. Так как нас интересуют остатки от деления суммы на 1000000 и из свойства остатков известно, что $(X+Y) \bmod M = ((X \bmod M) + (Y \bmod M)) \bmod M$, можно рассмотреть не сами суммы, а их остатки от деления. Так как нас интересует остаток равный 0, мы должны найти в последовательности остатков частичных сумм ближайшие одинаковые значения (если остатки одинаковы, значит остаток для суммы элементов подпоследовательности между ними равен 0). Для этого для каждого остатка нужно запоминать позицию, когда этот остаток встречался и находить минимальный вариант для разницы позиций.

Пример реализации:

```

var s:array[0..999999] of integer;
    n,i,l,p,v,o:integer;
begin
    read(n);
    o:=0;
    for i:=1 to 999999 do
        s[i]:=-1;
    s[0]:=0;
    l:=1000000;
    for i:=1 to n do
        begin
            read(v);
            o:=(o+v) mod 1000000;
            if s[o]>=0 then
                begin
                    if i-s[o]<l then
                        begin
                            l:=i-s[o];
                            p:=s[o]+1;
                        end;
                    end;
                s[o]:=i;
            end;
        if l=1000000 then
            writeln(-1)
        else
            writeln(l, ' ',p);
        end.

```

4. Слово и робот (7-8 класс)

Тема: исполнители
Сложность: простая

В первой подзадаче последовательно перемещаем буквы в нужные позиции, во второй подзадаче необходимо задействовать ветвление 1-й и 2-й буквы слова и 3-й и 4-й буквы. В третьей подзадаче необходимо создать универсальный вариант программы - последовательно с помощью ветвления находим буквы слова и выводим их в правильные позиции.

Примеры программ:

```
>>%<%<%>>%<%>%  
(W:%)>>(R:%)  
(W:>(W:>(W:%)<%)<%)>(O:>(O:%)<%)>(R:%)
```

4. Сборка компьютера (9-11 класс)

Тема: динамическое программирование
Частичные решения: полный перебор для 1 и 2 подзадачи

Сложность: выше среднего

```
type info=record c,r,t:integer; end;  
const inf=100000000;  
var shop:array[1..1000] of info;  
    n,b,best,c,r,i,j,k,t:integer;  
    dt:array[0..5,0..3000] of integer; { таблица ДП }  
    hw:array[0..5,0..3000] of integer; { как получили }  
    cmp:array[1..5] of integer;  
begin  
    read(t);  
    read(n);  
    for i:=1 to n do  
        read(shop[i].c,shop[i].r,shop[i].t);  
    read(b);  
    for i:=0 to t do  
        for j:=0 to b do  
            dt[i][j]:=-inf;  
        { прямой проход }  
        dt[0][0]:=0;  
        hw[0][0]:=0;  
        for i:=1 to t do  
            for k:=1 to n do  
                if shop[k].t=i then  
                    begin  
                        c:=shop[k].c;  
                        r:=shop[k].r;  
                        for j:=0 to b-c do  
                            if dt[i-1][j+c]<dt[i-1][j]+r then  
                                begin  
                                    dt[i][j+c]:=dt[i-1][j]+r;  
                                    hw[i][j+c]:=k;  
                                end;  
                            end;  
                        end;  
                    { поиск наилучшего решения }  
                    r:=-inf;best:=-1;  
                    for j:=0 to b do  
                        if dt[t][j]>r then  
                            begin  
                                r:=dt[t][j];  
                                best:=j;  
                            end;  
                        end;  
                    if best>0 then  
                        begin  
                            { обратный проход по таблице }  
                            writeln(r);  
                            for i:=t downto 1 do  
                                begin
```

```
        cmp[i]:=hw[i][best];
        best:=best-shop[cmp[i]].c;
    end;
    for i:=1 to t do
        write(' ',cmp[i]);
        writeln;
    end
else
    writeln(-1);
end.
```