Olimpiada Națională de Informatică
2004
etapa județeană

clasele XI-XII

{ Mugurel Ionut Andreica - Bucuresti, ROMANIA }

{ Time Complexity : O(log(K) * (N*K)^3 ? }

{ Cautare binara + Bellman Ford (cu coada) }

```
Program _lanterna_;

const filein = 'lanterna.in';
      fileout = 'lanterna.out';
      MAXN = 50;
      MAXK = 1000;
      infinit = 30000;
      maxdim1 = 63;

type linie = array [0..MAXK] of integer;
     plinie = ^linie;
     pcoada = ^coada;
     coada = record
               i : byte;
               k : integer;
               urm : pcoada;
             end;

var T, W : array [1..MAXN, 1..MAXN] of integer;
    Tmin : array [1..MAXN] of plinie;
    charge : array [1..MAXN] of boolean;
    inq : array[1..MAXN, 0..MAXK] of boolean;
    i, j, p, M, N, K, li, ls : integer;
    TOK, bestK : integer;
    cst, cfin, aux : pcoada;

procedure readdata;
begin
assign(input, filein);
reset(input);
read(N, K);

for i := 1 to N do
  begin
    read(p);
    charge[i] := (p = 1);
  end;
```

```
read(M);

for i := 1 to N do
  for j := 1 to N do
    begin
      T[i,j] := infinit;
      W[i,j] := MAXK + 1;
    end;

for p := 1 to m do
  begin
    read(i, j);
    read(T[i,j], W[i,j]);
    T[j,i] := T[i,j];
    W[j,i] := W[i,j];
  end;

close(input);

for i := 1 to N do
  begin
    new(Tmin[i]);
  end;
end;

procedure writedata;
begin
assign(output, fileout);
rewrite(output);
writeln(TOK, ' ', bestK);
close(output);
end;

function MinT(kmax : integer) : integer;
var i, j, k, p, q, tN : integer;

begin
for i := 1 to N do
  for j := 0 to kmax do
    begin
      Tmin[i]^[j] := infinit;
    end;

Tmin[1]^[0] := 0;
inq[1, 0] := true;

new(cst);
cst^.i := 1;
cst^.k := 0;
cst^.urm := nil;
```

```
cfin := cst;

tN := infinit;

while (cst <> nil) do
  begin
    i := cst^.i;
    k := cst^.k;

    if (Tmin[i]^[k] < tN) then
      for j := 1 to N do
        if (T[i,j] < infinit) and (k + W[i,j] <= kmax) then
          begin
            p := Tmin[i]^[k] + T[i,j];
            q := k + W[i,j];

            if (charge[j]) then
              q := 0;

            if (p < Tmin[j]^[q]) then
              begin
                Tmin[j]^[q] := p;

                if (j = N) and (Tmin[j]^[q] < tN) then
                  tN := Tmin[j]^[q];

                if (not inq[j, q]) and (Tmin[j]^[q] < tN) then
                  begin
                    inq[j, q] := true;

                    new(aux);
                    aux^.i := j;
                    aux^.k := q;
                    aux^.urm := nil;
                    cfin^.urm := aux;
                    cfin := aux;
                  end;
              end;
          end;

    inq[i, k] := false;
    aux := cst;
    cst := cst^.urm;
    dispose(aux);
  end;

MinT := tN;
end;

begin
readdata;
```

```
TOK := MinT(K);
bestK := K;

li := 1; ls := K-1;
while (li <= ls) do
  begin
    p := (li + ls) shr 1;

    if (MinT(p) = TOK) then
      begin
        bestK := p;
        ls := p-1;
      end
    else
      li := p+1;
  end;

writedata;
end.
```

{Solutia problemei MOSIA LUI PACALA - Rodica}
{Se ordoneaza punctele folosind alg. Hill pentru infasuratoare convexa
se calculeaza pentru fiecare par i aria d[i]*dist(par[i-1],par[i+1])/2
si se determina cu un alg. elementar de prog. dinamica secventa de pari
nealaturati care conduce la suma maxima a ariilor}

```
const maxi=251;fi='mosia.in';fo='mosia.out';
type punct=record x,y,d:longint end;
     puncte=array[0..maxi+1]of punct;
     sir=array[1..maxi]of integer;
var f:text; p:puncte;
    s,o:sir;
    n:integer;

procedure citire;
var i:integer;
begin
    assign(f,fi);reset(f);
    readln(f,n);
    for i:=1 to n do readln(f,p[i].x,p[i].y,p[i].d);
    for i:=1 to n do o[i]:=i;
    close(f)
end;

procedure qsort(l, r: Integer);
var
  i,j,mx,my,ax:integer;aux:punct;
begin
  i:=l; j:=r;
  my:=p[(l+r)div 2].y;
  mx:=p[(l+r)div 2].x;
  repeat
    while (p[i].y<my)or((p[i].y=my)and(p[i].x<mx)) do inc(i);
    while (p[j].y>my)or((p[j].y=my)and(p[j].x>mx)) do dec(j);
    if i<=j then begin
      aux:=p[i];p[i]:=p[j];p[j]:=aux;
      ax:=o[i];o[i]:=o[j];o[j]:=ax;
      inc(i);dec(j)
    end;
  until i>j;
  if l<j then qsort(l, j);
  if i<r then qsort(i, r);
end;

function sign(a,b,c:punct):shortint;
begin
     if c.x*(a.y-b.y)+c.y*(b.x-a.x)+a.x*b.y-b.x*a.y>=0 then
        sign:=1
     else sign:=-1
end;
```

```
procedure convex;
var free:array[1..maxi]of boolean;
    i,vf:integer;
begin
   for i:=1 to n do free[i]:=true;
   s[1]:=1;s[2]:=2;free[2]:=false;
   vf:=2;
   for i:=3 to n do begin
      while sign(p[s[vf-1]],p[s[vf]],p[i])=-1 do
        begin free[s[vf]]:=true;dec(vf) end;
      inc(vf);s[vf]:=i;free[i]:=false
   end;
   for i:=n-1 downto 1 do
      if free[i] then begin
        while sign(p[s[vf-1]],p[s[vf]],p[i])=-1 do dec(vf);
        inc(vf);s[vf]:=i
      end;
   if vf-1<>n then begin n:=vf-1;writeln('Date eronate') end
end;

procedure ssort;
var i:integer;
    p1:puncte;o1:sir;
begin
    for i:=1 to n do begin
       p1[i]:=p[s[i]];o1[i]:=o[s[i]]
    end;
    p:=p1;o:=o1
end;

function dist(a,b:punct):real;
begin
     dist:=sqrt(sqr(a.x-b.x)+sqr(a.y-b.y))
end;

procedure dinamic;
var i,j,k:integer;sum:real;
    ds:array[1..maxi]of real;
    d:array[1..2,1..maxi] of record s:real;i:byte end;
begin
   for i:=1 to n do ds[i]:=p[i].d/2*dist(p[i-1],p[i+1]);
   d[1,n].s:=ds[n];d[1,n].i:=1;
   d[1,n+1].s:=0;d[1,1].i:=0;
   for i:=n-1 downto 2 do
      if d[1,i+2].s+ds[i]>d[1,i+1].s then begin
        d[1,i].s:=d[1,i+2].s+ds[i];d[1,i].i:=1
      end
      else begin
        d[1,i].s:=d[1,i+1].s;d[1,i].i:=0
      end;
```

```
  d[1,1].s:=d[1,2].s;
  d[2,n].s:=0;d[2,n].i:=0;
  d[2,n+1].s:=0;
  for i:=n-1 downto 1 do
     if d[2,i+2].s+ds[i]>d[2,i+1].s then begin
        d[2,i].s:=d[2,i+2].s+ds[i];d[2,i].i:=1
     end
     else begin
        d[2,i].s:=d[2,i+1].s;d[2,i].i:=0
     end;
  if d[1,1].s>d[2,1].s then j:=1 else j:=2;
  i:=1;k:=0;
  while i<=n do
     if d[j,i].i=0 then inc(i)
     else begin inc(k);s[k]:=i;inc(i,2) end;
  for i:=1 to k do write(o[s[i]],' ');
  writeln;writeln(d[j,1].s:20:6);
  assign(f,fo);rewrite(f);write(f,d[j,1].s:20:6);close(f)
end;

begin
    citire;
    qsort(1,n);
    convex;
    ssort;
    p[0]:=p[n];p[n+1]:=p[1];
    dinamic;
end.
```