Central European Olympiad in Informatics

28 July – 4 August 2005 Sárospatak, Hungary
http://ceoi.inf.elte.hu

Mobile Service

We provide an algorithm using dynamics programming method.
Notice first that immediate prior to serving the request r_i one of the service staff employee is in the location r_{i-1}. For every i ($0 \leq i \leq n$), and for every locations v_2 and v_3 define the subproblem of serving the first i requests by minimal cost, providing that the other two employees are located in location v_2 and v_3 at the end. Denote this cost by $Opt(i, v_2, v_3)$. The solution of the subtask A is

$$\min_{v_2,v_3} Opt(n, v_2, v_3)$$

Notice that $Opt(i, v_2, v_3)$ is symmetric, that is $Opt(i, v_2, v_3) = Opt(i, v_3, v_2)$. Moreover, if $r_i = r_{i-1}$ then $Opt(i, v_2, v_3) = Opt(i-1, v_2, v_3)$
The cost of serving the request r_i by moving the employee from the location r_{i-1} is

$$cost1 = Opt(i-1, v_2, v_3) + C(r[i-1], r_i),$$

therefore a possible value of $Opt(i, v_2, v_3)$ is $cost1$.
If the employee moves from location v_2 to serve the request r_i then a possible value of $Opt(i, v_3, r_i)$ is

$$cost2 = Opt(i-1, v_2, v_3) + C(v_2, r_i)$$

If the employee moves from location v_2 to serve the request r_i then a possible value of $Opt(i, v_2, r_i)$ is

$$cost3 = Opt(i-1, v_2, v_3) + C(v_3, r_i)$$

We do forward recursion by computing for all v_2 and v_3 the value of $Opt(i, v_2, v_3)$ from the values $Opt(i-1, x_2, x_3)$.
It is essential for the memory usage that there is no need to store the values $Opt(i, v_2, v_3)$ for all i. Clearly, in order to compute $Opt(i, v_2, v_3)$ we need only the values $Opt(i-1, x_2, x_3)$.
In order to construct an optimal solution, (subtask B) we store the location for which the minimum obtained for $Opt(i, v_2, v_3)$ in the array element $Move[i, v_2, v_3]$.

Complexity:
Time: $O(n * m^2)$
Memory: $O(n * m^2)$
Implementation

```
program Service;
Const
  Maxm=200;     {max. # of locations}
  Maxn=1000;    {max. # of requests}
  MaxC=2000;    {max. cost}
  Inf=Maxn*MaxC+1;
Var
  n,m:longint;
  C:array[1..Maxm,1..Maxm] of longint; {cost matrix}
  R:array[1..Maxn] of 1..Maxm;         {the requests}
  Move:array[0..Maxn,1..Maxm, 1..Maxm] of 0..Maxm;
  Cost:Longint; {the optimal total cost}
  v20,v30:longint;

procedure ReadIn;
var i,j:longint;
  inFile:Text;
begin
```

```
  assign(inFile, 'service.in'); reset(inFile);
  readln(inFile, m,n);

  for i:=1 to m do begin
    for j:=1 to m do
      read(inFile, C[i,j]);
    readln(inFile);
  end{for i};
  for i:=1 to n do
      read(inFile, R[i]);

  close(inFile);
end {ReadIn};

procedure WriteOut;
var i,x,vm,v1,v2,v3:longint;
  outFile:Text;
  S:array[1..Maxn] of 0..Maxm;
begin {WriteOut}
  assign(outFile, 'service.out'); rewrite(outFile);

  writeln(outFile, Cost);
  v2:=v20; v3:=v30;
  vm:=Move[n,v2,v3]; S[n]:=vm;
  for i:=n-1 downto 1 do begin
    if vm<>r[i] then begin
       if r[i]=v2 then
         v2:=vm
       else
         v3:=vm
     end;
     if v2>v3 then begin
       x:=v2; v2:=v3; v3:=x;
     end;
     vm:=Move[i,v2,v3];
     S[i]:=vm;
  end;

  v1:=1; v2:=2; v3:=3;
  for i:=1 to n do
    if s[i]=v1 then begin
      write(outFile, 1,' ');
      v1:=r[i];
    end else if s[i]=v2 then begin
      write(outFile, 2,' ');
      v2:=r[i];
    end else begin
      write(outFile, 3,' ');
      v3:=r[i];
    end;
```

```
  writeln(outFile);
  close(outFile);
end {writeOut};

procedure ComputeOpt;
var i,v2,v3,v2i,v3i,newcost:longint;
  Opt:array[boolean,1..Maxm, 1..Maxm] of longint;
  ir,ir1:boolean;
begin {ComputeOpt}
  ir1:=false; ir:=true;
  for v2:=1 to m do
    for v3:=1 to m do Opt[ir,v2,v3]:=Inf;

  Opt[ir,1,2]:=C[3,r[1]]; Move[1,1,2]:=3;
  Opt[ir,2,3]:=C[1,r[1]]; Move[1,2,3]:=1;
  Opt[ir,1,3]:=C[2,r[1]]; Move[1,1,3]:=2;

  Opt[ir,2,1]:=Opt[ir,1,2]; Move[1,2,1]:=3;
  Opt[ir,3,2]:=Opt[ir,2,3]; Move[1,3,2]:=1;
  Opt[ir,3,1]:=Opt[ir,1,3]; Move[1,3,1]:=2;


  for i:=2 to n do begin
    ir1:=ir; ir:=not ir;
    for v2:=1 to m do
      for v3:=v2+1 to m do Opt[ir,v2,v3]:=Inf;

    for v2:=1 to m do begin
      for v3:=v2+1 to m do begin
        newcost:=Opt[ir1,v2,v3]+C[r[i-1],r[i]];
        if (v2<>r[i])and(v3<>r[i])and(newcost<Opt[ir,v2,v3]) then begin
            Opt[ir,v2,v3]:=newcost;
            Move[i,v2,v3]:=r[i-1];
        end;

        newcost:=Opt[ir1,v2,v3]+C[v2,r[i]]; {v2-> r[i]}
        if (r[i-1]<v3)then begin
          v2i:=r[i-1]; v3i:=v3
        end else begin
          v2i:=v3; v3i:=r[i-1];
        end;
        if (v2i<>v3i)and(v2i<>r[i])and(v3i<>r[i])and(newcost<Opt[ir,v2i,v3i]) then begin
          Opt[ir,v2i,v3i]:=newcost;
          Move[i,v2i,v3i]:=v2;
        end;

        newcost:=Opt[ir1,v2,v3]+C[v3,r[i]]; {v3-> r[i]}
        if (r[i-1]<v2)then begin
          v2i:=r[i-1]; v3i:=v2
        end else begin
          v2i:=v2; v3i:=r[i-1];
```


```
        end;
        if (v2i<>v3i)and(v2i<>r[i])and(v3i<>r[i])and(newcost<Opt[ir,v2i,v3i]) then begin
          Opt[ir,v2i,v3i]:=newcost;
          Move[i,v2i,v3i]:=v3;
        end;
      end {v3};
    end {v2};
  end {i};

  Cost:=Inf;
  for v2:=1 to m do
    for v3:=v2+1 to m do
    if (Opt[ir,v2,v3]<Cost) then begin
      Cost:=Opt[ir,v2,v3];
      v20:=v2; v30:=v3;
    end;
end {ComputeOpt};

begin {program}
  ReadIn;
  ComputeOpt;
  WriteOut;
end.
```