Takmičarski zadaci na školskom takmičenju iz informatike 2004. godine za učenike srednjih škola

mr. Vesna Veličković, asistent
vvesna@bankerinter.net
Aleksandar Ilić, student
andrejko@bankerinter.net

Prirodno Matematički Fakultet, Niš
Seminar nastave informatike
20.03.2004.

Takmičenja učenika srednjih škola iz informatike se održavaju svake godine u organizaciji Ministarstva prosvete Republike Srbije i Društva Matematičara Srbije. Postoji više rangova takmičenja. U zemlji su to školsko (ako je potrebno), okružno, republičko, savezno i Mala Olimpijada. Na Maloj Olimpijadi se bira četvoročlana ekipa, koja predstavlja našu zemlju na međunarodnim takmičnjima. Međunarodna takmičenja su: IOI (International Olympiad in Informatics) - što predstavlja svetsko takmičenje, BOI (Balkan Olympiad in Informatics) - što je takmičenje balkanskih zemalja, CEOI (Central European Olympiad in Informatics) - takmičenje centralno-evropskih zemalja.

Propozicije takmičenja su sledeće:
Svaki takmičar radi samostalno na posebnom kompjuteru. Uobičajeno je izrada zadataka 4 sata na nižim rangovima takmičenja, a 5 sati počev od saveznog takmičenja. Rade se tri zadatka. Zadaci se automatski testiraju po test primerima. Uobičajeno svaki test primer donosi 10 poena, što znači da je ukupan broj poena 300. Uobičajeno vremensko ograničenje po test primeru je 1 sekunda. Na nižim rangovima takmičenja, postoje dve kategorije: A za učenike koji rade po specijalnom programu Matematičke Gimanzije i B za ostale učenike.

Zadatak 1. Igra slagalica

Problem:
Mali Perica je gledao kviz "Muzičku slagalicu" i mnogo mu se dopala igra sa slovima. Naime, od datih N slova engleske abecede treba sastaviti što dužu reč, ali svako slovo se sme upotrebiti najviše jednom. Zato je Perica nabavio i ogromni rečnik koji ima M reči. Ali avaj, pretraživanje po rečniku je vrlo zamoran posao.
Vaš zadatak je da pomognete Perici i napišete program koji će naći najdužu reč. Ukoliko ima više reči iste dužine, štampati prvu u leksikografskom redosledu.

Ulaz:
U prvom redu ulazne datoteke igra.in nalazi se broj N ($1 \leq N \leq 100$). U sledećih N redova nalazi se po jedno malo slovo 'a'..'z' abecede.
Zatim sledi linija koja sadrži broj M ($1 \leq M \leq 10000$) i u sledećih M redova u svakom redu po jedna reč dužine manje ili jednake od 100.

Izlaz:
U izlazni fajl igra.out u prvom redu upisati dužinu najduže reči, a u drugom redu samu reč. Ukoliko tražena reč ne postoji, upisati -1.

Primeri:

```
IGRA.IN                    IGRA.OUT
4                          3
a                          aca
c
a
b
4
aca
maca
bcca
caa
```

```
IGRA.IN                    IGRA.OUT
3                          2
f                          ac
a
c
5
faca
cca
ac
a
cc
```

Rešenje zadatka 1.

Standardan zadatak iz informatike za stringove. Za svaku od M reči, treba videti da li je sastavljena od datih slova. Ovo se najlakše proverava, tako što se izbroji svako slovo engleske abecede koliko se puta javlja u trenutnoj reči i uporedi sa datim slovima. U nizu d se čuvaju podaci o tome koliko se puta nalazi i-to slovo. Za svaku reč generišemo niz a, koji sadrži informacije koliko se puta dato slovo pojavilo u trenutnoj reči. Prostim upoređivanjem nalazimo najduže rešenje.

Teža varijanta ovog problema je iz TV kviza "Muzička slagalica", kada svako slovo ima svoju celobrojnu pozitivnu vrednost. Recimo samoglasnici imaju vrednost 1, slova Š, Ž, Č, Đ, Dž imaju vrednost 3, a ostala slova vrednost 2.

Složenost je O (M · N).

```
procedure Igra;
var
  i, j, k, Len, n, m: LongInt;
  s, Sol: string;
  c: array [1..MaxN] of Char;
  a, d: array [1..MaxN] of LongInt;
  p: Boolean;
begin
{ ucitavanje i brojanje slova}
  ReadLn (f, n);
  for i := 1 to n do
    ReadLn (f, c [i]);
  FillChar (d, SizeOf (d), 0);
  for i := 1 to n do
    Inc (d [Ord (c [i]) - Ord ('a') + 1]);
  Len := 0; Sol := '';
  ReadLn (f, m);
{ ucitavanje reci i provera da li je sastavljena od istih slova}
  for i := 1 to m do begin
    ReadLn (f, s);
    k := Length (s);
    for j := 1 to 26 do
      a [j] := d [j];
    for j := 1 to k do
      Dec (a [Ord (s [j]) - Ord ('a') + 1]);
    p := True;
    for j := 1 to 26 do
      if (a [j] < 0) then begin
        p := False;
        Break;
      end;
{ nalazenje najduze reci}
    if p then begin
          if (Len < k) or ((Len = k) and (s < Sol)) then begin
        Len := k;
        Sol := s;
      end;
    end;
  end;
  if (Len <> 0) then begin
    WriteLn (g, Len);
    WriteLn (g, Sol);
  end
  else begin
    WriteLn (g, '-1');
  end;
end;
```

Zadatak 2. Mešanje karata

Problem:
Mali Perica je dobio za rođendan špil od 2N karata obeleženih redom od 1 do 2N i meša ih na sledeći način:

- preseče špil tačno na pola i dobije dve gomile A i B od N karata, gde je A gornja polovina i B donja polovina
- kombinuje karte tako što redom uzima kartu sa gomile A, pa kartu sa gomile B i stavlja na zajedničku gomilu, dok ne izmeša svih 2N karata.

Kako je Perica završio sve domaće zadatke, zanima se tako što meša karte ponovo i ponovo, dok ne dobije prvobitan raspored karata {1, 2, 3,..., 2N}. Zanima ga koliko puta mora da promeša špil da bi se karte vratile u orginalan raspored.

Ulaz:
U prvom redu ulazne datoteke karte.in se nalazi broj N ($1 \leq N \leq 5000$).

Izlaz:
U izlaznu fajl karte.out treba upisati broj mešanja potrebnih da se 2N karata vrate u originalnu poziciju.

Primeri:

KARTE.IN	KARTE.OUT
3	4
KARTE.IN	KARTE.OUT
5	6

Rešenje zadatka 2.

Zadatak iz nizova, sa lepom idejom. Najlakše rešenje i najočiglednije je ono kada redom mešamo karte, sve dok ne dobijemo početnu kombinaciju. Može se pokazati da je potrebno najviše 2N mešanja, pa je složenost ovog algoritma $O(N^2)$. Za ovo rešenje se dobijalo 65 poena u Pascal-u i 85 poena u C++.

Najbrže rešenje je inspirisano teorijom permutacija. Naime, svaka permutacija se sastoji od disjunktnih ciklusa. Red permutacije je najmanji broj n, tako da je f^n identička permutacija. Dakle, podelimo datu permutaciju u cikluse i prosto nadjemo najmanji zajednički sadržalac. Složenost problema je O(n).

Najmanji zajednički sadržalac je jednak količniku $a \cdot b$ i NZD (a, b). Najveći zajednički delilac nalazimo primenom Euklidovog algoritma.

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
(10, 5, 9, 4, 8, 3, 7, 2, 6, 1)
(1, 8, 6, 4, 2, 9, 7, 5, 3, 10)
(10, 2, 3, 4, 5, 6, 7, 8, 9, 1)
(1, 5, 9, 4, 8, 3, 7, 2, 6, 10)
(10, 8, 6, 4, 2, 9, 7, 5, 3, 1)
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Dakle, imamo cikluse (1, 10) (2, 8, 5) (3, 6, 9) (4) (7).

```
{funkcija za najveci zajednicki delilac Euklidovim algoritmom}
function Nzd (a, b: LongInt): LongInt;
begin
  if b = 0 then
    Nzd := a
  else
    Nzd := Nzd (b, a mod b);
end;

procedure Solve;
var
  i, k, x: LongInt;
begin
  Sol := 1;
  FillChar (c, SizeOf (c), False);
  for i := 1 to 2 * n do
    if (not c [i]) then begin
{ nalazenje ciklusa za nemarkiranu kartu i}
     x := i;
     k := 0;
     repeat
        c [x] := True;
        if (x <= n) then
         x := 2 * (n - x + 1)
       else
          x := 4 * n - 2 * x + 1;
        Inc (k);
      until (x = i);
          Sol := (Sol * k) div Nzd (Sol, k)
    end;
end;
```

Zadatak 3. Put do škole

Problem:
Prošao je raspust, zima je u jeku, i mali Perica mora ponovo u školu. Data je pravougaona mapa dimenzija N × M i mali Perica se nalazi u donjem levom uglu P, a škola u gornjem desnom S. On se kreće samo desno ili nagore, nikako dijagonalno, dole ili levo. Na putu do škole mali Perica mora da se sretne sa svim drugarima koji se nalaze na mapi označeni sa D. Kako je sneg mnogo napadao, mali Perica ne može da ide po poljima koja su zavejana snegom, na mapi označena sa X.
Odrediti broj različitih puteva kojim Perica može da dođe u školu, tako da zaobiđe sve prepreke i pokupi sve drugare. Dva puta su različita ako se razlikuju bar u nekom delu puta.

Ulaz:
U prvom redu datoteke skola.in su četiri prirodna broja razdvojenih razmakom: dimenzije table N i M ($1 \le N, M \le 100$), broj drugara D ($1 \le D \le 10000$) i broj snežnih prepreka X ($1 \le X \le 10000$).
U svakom od sledećih D redova stoji par brojeva (A, B) koordinate kuća Pericinih drugara. Koordinata levog donjeg ugla, gde se nalazi Perica je (1, 1). Dalje, slede X redova koji svaki sadrži koordinate snežnog nanosa.
U ulaznom fajlu neće biti poklapanja parova koordinata drugara i prepreka. Ni prepreka ni kuća se ne nalaze u gornjem desnom ili donjem levom uglu.

Izlaz:
U jedinom redu fajla skola.out treba upisati broj različitih puteva od kuće do škole. Ukoliko nije moguće da Perica pokupi sve drugare na traženi način upisati -1. Garantuje se da će rezultat biti manji do 2^{31}.

Primer:

```
SKOLA.IN
5 8 4 4
1 2
2 5
3 5
4 7
1 5
2 2
2 7
3 6
```

```
SKOLA.OUT
4
```

```
SKOLA.IN
5 5 2 3
1 4
3 3
3 2
2 4
2 5
```

```
SKOLA.OUT
-1
```

0	0	0	0	0	0	2	4
0	0	0	0	2	2	2	2
0	0	0	0	2	X	0	0
0	X	1	2	2	0	X	0
1	1	1	1	X	0	0	0

Rešenje zadatka 3.

Lep problem iz dinamičkog programiranja i pretraživanja matrica. Sličan zadatak je onaj kada se kralj nalazi u donjem levom ćošku i želi da dođe u gornji desni, ali tako da pokupi što je moguće veći zbir, a sme da se kreće samo gore, desno i dijagonalno gore-desno.

U zadatku je izbegnuto korišćenje velikih brojeva zbog jednostavnosti.
Najprostije rešenje je rekurzijom (backtrack). Ovo rešenje donosi 30 poena.

Dinamičko programiranje je rešavanje problema svođenjem na prethodne manje probleme. To je jedna metoda problema optimizacije. Dakle, napravimo novu matricu d [n, m], za koju važi d [i, j] je broj različitih puteva od polja (1,1) do polja (i, j), ali takvih da smo pokupili sve drugare na putu i zaobišli sve prepreke. Matricu ćemo da popunjavamo odozdo nagore i sleva udesno. Ispravnost algoritma se dokazuje indukcijom po dimenziji matrice.

Naravno, drugare mora da obilazi sortirane po x osi (analogno po y osi). Promenljive p i q služe da zapamte poslednjeg obiđenog drugara. Sada je formula generisana na sledeći način:
d [i, j] = 0
d [i, j] = d [i - 1, j] + d [i, j], ako je i ≥ p + 1 i a [i - 1, j] nije prepreka
d [i, j] = d [i, j - 1] + d [i, j], ako je j ≥ q + 1 i a [i, j - 1] nije prepreka

Složenost algoritma je O (nm), jer smo samo jednom obišli matricu.

```
procedure Skola;
var
  i, j, p, q: LongInt;
begin { ucitavanje matrice i inicijalizacija}
  Read (f, n, m, d, x);
  FillChar (c, SizeOf (c), False);
  for i := 1 to d do begin
    Read (f, p, q);
    c [p, q] := True;
  end;
  c [n, m] := True;
  FillChar (a, SizeOf (a), 0);
  for i := 1 to x do begin
    Read (f, p, q);
    a [p, q] := -1;
  end;
  p := 1; q := 1;
  a [1, 1] := 1; {obilazak matrice i racunanje a [i, j]}
  for i := 1 to n do
    for j := 1 to m do
      if (p <= i) and (q <= j) and (a [i, j] <> -1) then begin
        if (i > 1) and (a [i - 1, j] <> -1) and (i - 1 >= p) then
          a [i, j] := a [i, j] + a [i - 1, j];
        if (j > 1) and (a [i, j - 1] <> -1) and (j - 1 >= q) then
          a [i, j] := a [i, j] + a [i, j - 1];
        if c [i, j] then begin
          p := i;
          q := j;
        end;
        c [i, j] := False;
      end;
  Sol := -1; {provera da li je svaki drugar obidjen}
  for i := 1 to n do
    for j := 1 to m do
      if c [i, j] then
        Exit;
  Sol := a [n, m];
end;
```

Zadatak 4. Drugar Jovica

Problem:
Data je mreža sa N gradova. Mali Perica se nalazi u tački A, a Jovica se nalazi u tački B negde na mapi. Mali Perica hoće što pre da stigne do Jovice, jer je čuo novi vic i jedva čeka da ga ispriča. On može ići peške ili koristiti železnicu, što je naravno brže nego da hoda. Ali ima novca samo za jednu kartu, tj. neograničenu vožnju dok ne izađe iz metroa. Mali Perica može da uđe ili izađe iz voza, samo u nekom od gradova. Pretpostavite da Perica ne gubi vreme pri menjanju vozova i da ne čeka na stanicama.
Vaš zadatak je da na osnovu njihovih koordinata i postojećih železnica odredite najkraće vreme za koje mali Perica može stići iz A u B. Sve železnice su prave linije, tako da se vreme računa po poznatoj formuli: t = s / v. Ukoliko ima više puteva iste dužine štampati bilo koji.

Ulaz:
U prvom redu jovica.in su dva realna broja v_1 i v_2 ($0 \le v_1 \le v_2 \le 1000$). Prvi broj je brzina hodanja, a drugi je brzina podzemne železnice. U sledećem redu su koordinate tačaka A i B, tj. Perice i Jovice.
U sledećem redu je je broj gradova N ($0 \le N \le 200$). Zatim slede N redova sa realnim koordinatama gradova (x, y). U sledećih M ($0 \le M \le 5000$) redova sa parovima brojeva (P, K), što kaže da su gradovi P i K povezani pravom železnicom. Ulaz se prekida parom brojeva 0 0.

Izlaz:
U prvom redu izlazne datoteke jovica.out treba ispisati minimalno vreme na 5 decimala. U sledećem redu broj stanica koje mali Perica koristi za prevoz (moguće i nula), a u novom redu i lista stanica u redosledu kretanja.

Primer:

```
JOVICA.IN                    JOVICA.OUT
1 100                        2.63463
10 10                        4
10 0                         4 2 1 3
4
0 0
1 0
9 0
9 9
1 2
1 3
2 4
0 0
```

```
JOVICA.IN                    JOVICA.OUT
1 20                         1.00000
0 0                          0
1 0
3
2 2
3 3
2 1
1 2
1 3
0 0
```

Rešenje zadatka 4.

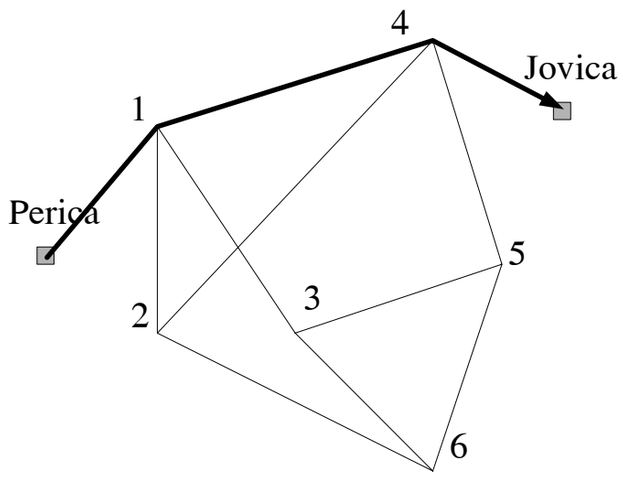
Ovo je najteži zadatak na školskom takmičenju i ranga je republičkog takmičenja. Prevedimo sve na teoriju grafova. Dakle, dato je n čvorova grafa i skup njegovih ivica koje imaju svoju težinu W (euklidsko rastojanje između gradova podeljeno sa brzinom v_2). Perica treba da uđe u neki čvor, kreće se po grafu, i stigne do Jovice.

Prvo rešenje je korišćenje Floyd-Warshal - ovog algoritma, koje donosi 70 poena. Ovaj algoritam nalazi najbliže puteve od svakog do svakog čvora i to čuva u matrici d. Sada se jednostavno proba svaki grad kao ulazni i svaki grad kao izlazni, i dobije se minimalno rešenje. Složenost ovog algoritma je $O(n^3+n^2)=O(n^3)$.

Optimalno rešenje je Dijkstrin algoritam. Definišimo još dva čvora u grafu 0 - koordinate Perice i n+1 - koordinate Jovice i ubacimo još 2n+1 ivicu: od čvora 0 do svakog od ostalih, i od čvora n+1 do svakog od ostalih čvorova. Sada se problem svodi na nalaženje najkraćeg puta u grafu od čvora 0 do čvora n+1. Dakle, startni čvor za Dijkstrin algoritam je čvor 0, a završni je n+1.

Ideja je razmatrati čvorove reodm prema dužinama najkraćih puteva od 0 do njih. Na početku se sve dužine inicijalizuju na beskonačno (ili -1 zbog implementacije). Najpre proveravamo sve grane koje izlaze iz čvora 0. Neka je (0, v) najkraća među njima za neki čvor v. Sigurno znamo da je to najkraći put do v, što nam služi kao početak. Pokušavamo da napravimo sledeći korak. Postepeno proširujemo skup čvorova za koje sigurno znamo najkraća rastojanja i te čvorove označavamo. Naime u svakoj iteraciji od neoznačenih čvorova biramo onaj sa najmanjom dužinom od tog skupa. Može se pkazati ispravnost ovog algoritma.

Za ovo nam su nam potrebna tri niza:
c - niz indikatora obiđenosti
d - niz rastojanja čvorova od početnog
p - niz prethodnika za optimalan put
U svakoj iteraciji nalazimo novi čvor i njegovu minimalnu dužinu i ažuriramo niz d i p.



Složenost rešenja je $O(n^2)$, jer imamo dve ugnježdene petlje.

```
function Dist (i, j: Integer): Real;
begin
  Dist := Sqrt ((x [i] - x [j]) * (x [i] - x [j]) + (y [i] - y [j]) *
(y [i] - y [j]));
end;

procedure Jovica;
var
  tmp, i, j, v: Integer;
  Min: Real;
begin
{ inicijalizacija nizova}
  for i := 0 to n + 1 do begin
    d [i] := -1; p [i] := -1;
    c [i] := False;
  end;
  d [0] := 0;

  for i := 0 to n + 1 do begin
{ nalazenje najblizeg neobidjenog cvora}
    Min := 1e10;
    for j := 0 to n + 1 do
      if (not c [j]) and (d [j] > 0) and (Min > d [j]) then begin
        Min := d [j];
        v := j;
     end;
      if (v = n + 1) then
        Break;
{ azuriranje nizova d i p}
      c [v] := True;
      for j := 0 to n + 1 do
        if (not c [j]) then begin
          if ((v = 0) or (j = n + 1)) and ((d [j] < 0)
             or (Dist (v, j) / v1 + d [v] < d [j]))
          then begin
            d [j] := d [v] + Dist (v, j) / v1;
            p [j] := v;
          end;
          if a [v, j] and ((d [j] < 0)
             or (Dist (v, j) / v2 + d [v] < d [j]))
          then begin
            d [j] := d [v] + Dist (v, j) / v2;
            p [j] := v;
          end;
        end;
  end;
{ nalazenje najkraceg puta do cvora n+1}
  Sol := d [n + 1];
  k := -1; tmp := n + 1;
  while (tmp <> 0) do begin
    Inc (k);
    s [k] := tmp;
    tmp := p [tmp];
  end;
end;
```