Takmičarski zadaci na okružnom 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. Niz cifara

Prirodni brojevi su zapisani jedan iza drugog tako da čine beskonačni niz cifara:
12345678910111213141516171819202122232 4...
Za dati broj n, naći n - tu cifru u ovom nizu.

Ulazni podaci nalaze se u tekstualnom fajlu ZAD1.DAT. U prvom redu ulaznog fajla se nalazi ceo broj n ($1 \le n \le 2 \cdot 10^9$).
Izlazne podatke treba upisati u tekstualni fajl ZAD1.RES. U prvi red fajla upisati cifru koja se nalazi na n - tom mestu.

Primer:

ZAD1.DAT	ZAD1.RES
20	1

Odredimo 1000001-tu cifru na tabli:

jednocifreni brojevi	$9 \cdot 1$ cifara
dvocifreni brojevi	$90 \cdot 2$ cifara
trocifreni brojevi	$900 \cdot 3$ cifara
četvorocifreni brojevi	$9000 \cdot 4$ cifara
petocifreni brojevi	$90000 \cdot 5$ cifara
ukupno	488889 cifara

Tražena cifra se nalazi u broju koji ima 6 cifara, jer je $488889 + 900000 \cdot 6 > 1000001 - 1$. Dakle, među svim šestocifrenim brojevima mi tražimo $1000000 - 488889 = 511111$ - tu cifru. To je broj $100000 + [511111 / 6] = 185185$. Pre broja 185185 ima ukupno $85185 \cdot 6 = 511110$, što kaže da u datom broju trebamo prvu cifru. Rešenje je 1.

Rešenje zadatka 1.

Ovo je zadatak iz matematike, koji se pojavljivao na takmičenjima u osnovnoj školi.

Za generisanje svih brojeva redom i traženja n-te cifre se dobija 30 poena.

Pametniji algoritam je sledeći: Odrediti koliko cifara ima broj koji sadrži traženu cifru, a zatim odrediti koji je to broj i koju njegovu cifru tražimo. Važna činjenica je da k - tocifrenih brojeva ima ukupno $9 \cdot 10^k$.
Najlakše je zbog ostataka prenumerisati cifre datog broja tako da počinju od 0. Zatim oduzimamo redom jednocifrene brojeve, dvocifrene brojeve, ..., sve dok ne dobijemo da dati broj ima d cifara. Sada prosto celobrojno podelimo n sa d i dobijemo broj koji sadrži n-tu cifru. Ostaje još da odredimo koja je tražena cifra sa suprotnog kraja, pošto se cifre nekog broja određuju sa suprotne strane.

Vremenska složenost je O (log N). Sledi kod u PASCAL-u, gde je n uneti broj, a Sol rešenje.

```
procedure Zad1;
var
  x, y, d, i: LongInt;
begin
  Dec (n);
  x := 9;
  d := 1;
{ nalazenje duzine broja u kome se nalazi n-ta cifra}
  while (d <= 8) and (d * x <= n) do begin
    n := n - d * x;
    Inc (d);
    x := x * 10;
  end;
  x := deg [d - 1] + n div d;
  y := d - n mod d - 1;
{ nalazenje y-te cifre u broju x}
  while (x > 0) and (y >= 0) do begin
    Sol := x mod 10;
    x := x div 10;
    Dec (y);
  end;
end;
```

Zadatak 2. Tenis

Jedini domaći internacionalni stonoteniski sudija Đura bio je pozvan da sudi finale na čuvenom teniskom truniru SCG Open 2004. Međutim, kako su se organizatori kasnije uverili, Đura nije baš najbolje upoznat sa sistemom bodovanja u tenisu, koji se razlikuje od onog u stonom tenisu. Sve što je ostalo su Đurine beleške na osnovu kojih organizatori žele da rekonstruišu meč (koji je prekinut zbog incidenta).
Tenis se igra po sledećim pravilima. Igrač koji osvoji bar 4 poena i ima bar 2 više od protivnika osvojio je gem. Igrač koji osvoji bar 6 gemova i ima bar 2 gema više od protivnika osvojio je set (na SCG Open - u se igra bez tajbrejka). Jedan igrač neprestano servira u jednom gemu, sve do kraja tog gema. Igrači se smenjuju na servisu posle svakog gema.

Ulazani podaci se nalaze u tekstualnom fajlu ZAD2.DAT. Ulaz ima tačno jedan red koji se sastoji od velikih slova S i P bez praznina. Ako na i-tom mestu stoji slovo S, to znači da je igrač koji je servirao za i-ti poen osvojio taj poen. Ako na i-tom mestu stoji slovo P, tada je i-ti poen osvojio igrač koji je primao servis. U meču je odigrano najviše 1000 poena.
U izlazni fajl ZAD2.RES treba ispisati rezultat po setovima u trenutku kada je meč prekinut. Rezultat svakog seta (tj. broj gemova koje su osvojili igrači) treba ispisati u posebnom redu. Znači u svakom redu se nalaze dva cela broja razdvojena prazninom. Prvi od ta dva broja je broj gemova koje je u tom setu osvojio prvi igrač, a drugi je broj gemova drugog igrača. Uvek treba prvo ispisati broj gemova igrača koji je prvi servirao u prvom gemu meča.

Primer:

ZAD2.DAT	ZAD2.RES
SSSSSSSSSSSSSS	2 1

Rešenje zadatka 2.

Ovaj zadatak je bio najlakši zadatak na takmičenju.

Takmičare označimo 1 i 2, i za svakog pamtimo broj poena u trenutnom gemu i broj gemova u trenutnom setu. Na početku iniciramo sve vrednosti na 0 i servera (igrača koji trenutno servira) na 1. Pri učitavanju karaktera tj. takmičara koji je osvojio poen, proverimo da li je neki od takmičara napravio bar 4 poena i bar 2 više od svog protivnika. Ukoliko jeste, povećamo rezultat u gemovima i promenimo servera. Ako je rezultat u gemovima nekog igrača veći bar za 2 od protivnika i veći ili jednak od 6, to ispisujemo rezultat u izlazni fajl.

U ovom zadatku simultano čitamo iz ulazne datoteke i pišemo rešenja. Ako je trenutni igrač x, onda je protivnik 3 - x.

Složenost je O(n). Sledi kod u PASCAL-u.

```
procedure Zad2;
var i, Server, n, Sol: LongInt;
  a: array [1..MaxN] of LongInt;
  PSet, PGem: array [1..2] of LongInt;
  c: Char;
begin
  PSet [1] := 0; PSet [2] := 0;
  PGem [1] := 0; PGem [2] := 0;
  Server := 1;
  while (not EoLn (f)) do begin
    Read (f, c);
{ ucitavanje karaktera i azuriranje rezultata u poenima}
    if (c = 'S') then
      Inc (PGem [Server])
    else
     Inc (PGem [3 - Server]);
    for i := 1 to 2 do
{ provera da li je igrac pobedio u gemu ili setu}
      if (PGem [i] >= 4) and (PGem [i] >= PGem [3 - i] + 2)
      then begin
        Server := 3 - Server;
        Inc (PSet [i]);
          if (PSet [i] >= 6) and (PSet [i] >= PSet [3 - i] + 2)
          then begin
            WriteLn (g, PSet [1], ' ', PSet [2]);
            PSet [1] := 0;
            PSet [2] := 0;
          end;
        PGem [1] := 0;
        PGem [2] := 0;
     end;
  end;
  if (PGem[1]>0) or (PGem[2]>0) or (PSet[1]>0) or (PSet[2]>0) then
    WriteLn (g, PSet [1], ' ', PSet [2]);
end;
```

Zadatak 3. Molekuli

Hemičar radi na projektu stvaranja super-molekula od četiri zadata molekula jednake dužine. Molekul se predstavlja kao niz atoma koji ga sačinjavaju. Atomi su predstavljeni velikim slovima abecede. Model super molekul može se predstaviti u dvodimenzionalnom prostoru tako da su bilo koja dva, od četiri zadata molekula vertikalna, a preostala dva horizontalna. Pri tome svaki od horizontalno postavljenih molekula sa svakim od vertikalno postavljenih molekula ima tačno jedan zajednički (presečni) atom.

Pravila sastavljenja super molekula su sledeća:

- svaki molekul mora biti upotrebljen tačno jedanput
- bilo koji od četiri zadata molekula može biti smešten u bilo koji horizontalni i bilo koji vertikalni red super molekula
- ako je molekul horizontalno smešten, on mora biti orijentisan s leva na desno
- ako je molekul vertikalno smešten, on mora biti orijentisan odozgo na dole
- površina pravougaonika koji se nalazi u centru super molekula treba da bude što je moguće veća. Ova površina nikada neće biti nula (Presečni atom se ne računa u dužinu stranice pravougaonika)
- prvi i poslednji atom bilo kog molekula ne može biti presečni atom

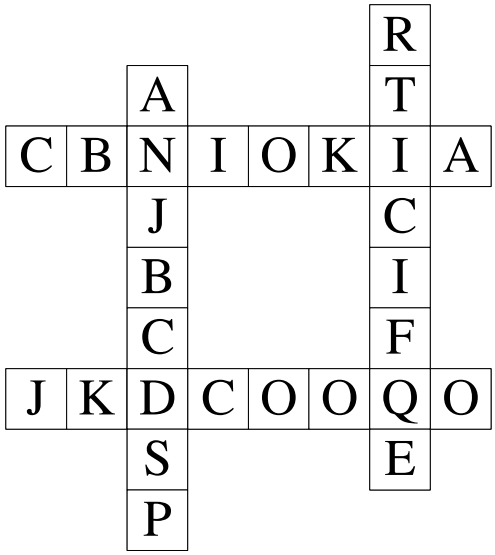
Pomozite hemičaru da sastavi super molekul koji zadovoljava navedena pravila.

Ulazni podaci se nalaze u tekstualnom fajlu ZAD3.DAT. U prvom redu fajla nalazi se ceo broj n ($5 \leq n \leq 20$) koji predstavlja dužinu molekula. U naredna četiri reda nalze se zapisi četiri molekula velikim abecednim slovima bez razmaka između slova.
Izlazni podaci se ispisuju u tekstualni fajl ZAD3.RES. U prvom redu fajla ispisati kolika je maksimalna površina super molekula koji zadovoljava navedena pravila. Ako ne postoji super-molekul koji ispunjava zadata pravila ispisati nulu.

Primer:

ZAD3.DAT
```
8
ANJBCDSP
CBNIOKIA
JKDCOOQO
RTICIFQE
```

ZAD3.RES
```
9
```



Rešenje zadatka 3.

Ovo je klasičan primer backtracking-a, ili probanja svih mogućnosti sa vraćanjem. Zadatak nema nikakvu ideju, već je samo težak za programiranje. Dakle, fiksiramo jedan niz, i onda pokušamo da preklopimo drugi i treći niz popreko (horizontalno). Za svako takvo preklapanje, četvrti niz pomeramo odozgo nadole i pamtimo kada su se poklopili karakteri. Naravno uvek ažuriramo rešenje.

Dakle, imamo ugnježdene petlje, jer tako režemo grane u grafu obilaska. U okviru dve petlje po brojačima i i j, imamo još dve petlje po k i l, koje proveravaju da li se odgovarajući elementi poklapaju. Zatim odredimo moguću širinu pravougaonika, stavimo poslednji niz paralelno prvim i proverimo da li smo u preseku dobili iste karaktere.

Permutacije nalazimo rekurzijom, uz jedan niz indikatora. Moguće je sve permutacije od 4 elemenata zapamtiti u konstantan niz, čime neznatno ubrzavamo program. Složenost je O $(24 \cdot 20^5)$.

```
procedure Zad3 (pr: LongInt);
var i, j, x, y, z, Maxx, k, l: LongInt;
begin
  if (pr = 5) then begin
{ provera za trenutnu permutaciju }
    for i := 2 to n - 3 do
      for j := i + 2 to n - 1 do begin
        for k := 2 to n - 3 do
          if (a [p [4], i] = a [p [1], k]) then
           for l := 2 to n - 3 do
            if (a [p [4], j] = a [p [2], l]) then begin
                y := j - i - 1;
                Maxx := n - k - 2;
                if (n - l - 2 < Maxx) then
                  Maxx := n - l - 2;
                if (Sol >= Maxx * y) then Break;
{ za fiksirane nizove 4, 1 i 2, provera da li se stap 3 uklapa}
                for x := 1 to Maxx do
                  for z := 2 to n - y - 2 do
                      if (a [p [1], k + x + 1] = a [p [3], z]) and
(a [p [2], l + x + 1] = a [p [3], z + y + 1]) then begin
                      if (Sol < x * y) then Sol := x * y;
                   end;
              end;
      end;
  end
  else begin { generisanje nove permutacije nizova }
    for i := 1 to 4 do
      if (c [i] = 0) then begin
        c [i] := 1; p [pr] := i;
        Zad3 (pr + 1);
        c [i] := 0; p [pr] := 0;
      end;
  end;
end;
```

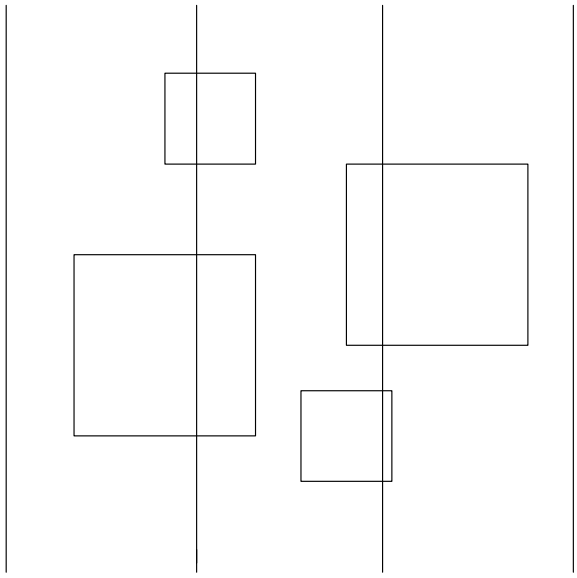
Zadatak 4. Jagode

Posle dugoročne saradnje na plantažama jagoda, farmeri Šomi, Draganče i Đurica su se posvađali, te su poželeli da podele plantaže na jednake delove. Pošto Viskonsin (država iz koje su naši farmeri) poznata po tome da ima močvarno zemljište farmeri su pravili plantaže u obliku kvadrata. Da bi spasili papirologije odlučili su da zemljište podele linijama koje idu od severa ka jugu tako da svako dobije neke od plantaža (i neke delove ostalih). Njihov komšija Zlatević je uzeo plan u ruke i rekao da je problem lako rešiti. Međutim u međuvremenu farmer Zlatević se izgubio jureći zlatnu žabu u šumi. Tako da na vama ostaje da pomognete farmerima da reše svoj problem.

Ulazni podaci se nalaze u tekstualnom fajlu ZAD4.DAT. U prvom redu nalzi se ceo broj N ($1 \leq N \leq 200$), koji predstavlja ukupan broj plantaža koje poseduju Šomi, Draganče i Đurica. U svakom od sledećih redova nalaze se tri realna broja X, Y, A ($0 \leq X, Y, A \leq 30000$). X i Y predstavljaju koordinate početka plantaže na mapi (jugozapadni ćošak ili donji levi ugao plantaže), dok A predstavlja dužinu strane plantaže. Stranice plantaže su uvek paralelne koordinatnim osama.
Izlazni podaci se ispisuju u tekstrualnom fajlu ZAD4.RES. U fajlu treba ispisati dva realna broja, sa tačnošću na dve decimale (tj. sa dve cifre iz decimalne tačke) u dva zasebna reda. Oni predstavljaju X koordinatu zamišljenih linija koje dele zemljište. Rešenja su u svakom test primeru jedinstvena.

Primer:

ZAD4.DAT	ZAD4.RES
2	67.67
1 1 100	233.33
200 200 100	



Rešenje zadatka 4.

Ovo je jednostavna i elegantna primena binarnog pretraživanja. Osnovna ideja je podela prostora pretraživanja na dva približno jednaka dela postavljanjem samo jednog pitanja.

Dat je sortirani niz x, tj. $x_1<x_2<...<x_n$. Treba ispitati da li se zadati broj z pojavljuje u nizu.
Neka je levi indeks podniza koji trenutno obrađujemo l, a desni r. Proverimo srednji element u nizu tj. onaj sa indeksom m = (l + r) div 2. Ako je z > a [m], onda z ne može biti u prvoj polovini, a ako je z < a [m], tada se z ne nalazi u drugoj polovini. Pronalaženje z u prvoj ili drugoj polovini niza je problem veličine n/2, koji se rešava rekurzivno. Bazni slučaj za n = 1 ili n = 2, se rešava neposrednim upoređivanjem elementa niza sa z.

Dakle, u našem problemu polazimo od koordinata l = 0 i r = 30000, a pretragu završavamo kada je razlika r - l manja od 0.001, zbog zaokruživanja. Kada je postavljena prava x = m, izračunamo zbir površina levo od prave jednim prolaskom kroz niz kvadrata.
Složenost je $O(\log(3\cdot10^8)\cdot N)$. Rešenje u PASCAL-u izgleda ovako:

```
function Find (l, r, k: Double): Double;
var p, m: Double;
  i: LongInt;
begin
  if (r - l <= 0.001) then begin
{ resenje je nadjeno na 3 decimale }
    Find := (l + r) / 2;
  end
  else begin { racunanje povrsine poseda levo od prave x = m }
    m := (l + r) / 2; p := 0;
    for i := 1 to n do
      if (x [i] >= m) then
        p := p + a [i] * a [i]
      else
        if (x [i] <= m) and (x [i] + a [i] >= m) then
          p := p + a [i] * (x [i] + a [i] - m);
{ da li je prava x = m levo ili desno od trazene }
    if (Area * k >= p) then
      Find := Find (l, m, k)
    else
      Find := Find (m, r, k);
  end;
end;

procedure Zad4;
var i: LongInt;
begin { racunanje ukupne povrsine kvadrata}
  l := 0; r := 30000;
  Area := 0;
  for i := 1 to n do
    Area := Area + a [i] * a [i];
  x1 := Find (l, r, 2 / 3);
  x2 := Find (l, r, 1 / 3);
end;
```