Konstrukcija i analiza algoritama, ispit, januar 2014.

1. U ravni je dato **N**, gde je **N** paran broj, tačaka sa celobrojnim koordinatama i prava **p**. Konstruisati C program koji će naći pravu **q** koja je paralelna sa pravom **p** i koja deli dati skup tačaka na dva jednaka dela (sadrže isti broj tačaka). Ukoliko tačka pripada pravoj, može se smatrati da pripada bilo kom delu. Prvi red standardnog ulaza sadrži četiri cela broj **N**, **A**, **B** i **C** odvojena jednim znakom razmaka, (1 <= **N** <= 10^3, -10^3 <= **A**, **B**,**C** <= 10^3), koji predstavljaju broj datih tačaka i pravu **p**. Prava **p** je predstavljena oblikom **Ax+By+C=0**. U sledeća **N** reda, date su koordinate tačaka, celi brojevi **x_k** i **y_k** iz segmenta [-10^3, 10^3]. U prvi i jedini red standardnog izlaza ispisati tri realna broja **D**, **E** i **F** koji predstavljaju kooeficiente tražene prave **q** (**Dx+Ey+F=0**). Ukoliko rešenje nije jedinstveno, štampati bilo koje. Možete koeficijente štampati sa tačnošću na dve decimale. Vremensko ograničenje: 0.1s (složenost O(n)), m.ograničenje:16MB

ULAZ	IZLAZ
4 -1 1 0 0 3 1 3 2 3 1 1	-1 1 -1.50

2. Konstruisati algoritam linearne složenosti za nalaženje najmanjeg pokrivača grana datog stabla. Pokrivač grana neusmerenog grafa G = (V,E) je skup čvorova U takav da je svaka grana iz E susedna bar jednom čvoru iz U.
3. Konstruisati algoritam za određivanje maksimalnog od datih n brojeva (ne nužno različitih) na modelu CRCW, tako da vreme izvršavanja algoritma bude O(1) i da koristi najvise n^2 procesora .
4. Ako je dat algoritam za množenje dve n * n donje trougaone matrice čije vreme izvrsavanja je O(T(n)), dokazati da postoji algoritam za množenje dve proizvoljne n *n matrice čije vreme izvršavanja je $O(T(n)+n^2)$. (Može se pretpostaviti da je T(cn)=O(T(n)) za svaku konstantu c)

Rešenja:

1.

```
#include <iostream>
#include <algorithm>
#include <cstdlib>
#include <cstdio>
using namespace std;

typedef struct
{       int x;
        int y;
} Tacka;
double sc[1000]; //y=kx+n, n=slobodni clan sc

int main()
{
        int N, A, B, C;
        double a, b1;
        int i;
        Tacka t[10000];

        scanf("%d%d%d%d",&N,&A,&B,&C);

        for (i=0;i<N;i++)
        {
                scanf("%d", &t[i].x);
                scanf("%d", &t[i].y);
        }


        if (B == 0)
        {
                for (i = 0; i < N; i++)sc[i] = t[i].x;
                  nth_element(sc, sc + N / 2, sc+N);
                  b1= sc[N/2];
                  printf("1.00 0.00 %.2f\n", -b1); // rezultat je skaliran
           return 0;
```

```
        }

  if (A == 0)
        {
              for (i = 0; i < N; i++)
                    sc[i] = t[i].y;

    nth_element(sc, sc + N / 2, sc+N);
  b1= sc[N/2]+0.0;

              printf("0.00 1.00 %.2f\n", -b1); // rezultat je skaliran
          return 0;
        }
        else
        {
              a = -A*1.0/B;


              for (i=0;i<N;i++)
                sc[i] = (t[i].y - a*t[i].x)+0.0;



  nth_element(sc, sc + N / 2, sc+N);
  b1= sc[N/2];


              printf("%lf %lf %lf\n", -a, 1.0, -b1); // rezultat je skaliran
         return 0;
        }


}
```

2.

Posmatrajmo proizvoljni list v i njegovog oca w. Grana (v,w) može se pokriti čvorom v, ali je bolje pokriti čvorom w, iz razloga što čvor v pokriva samo (v,w), a čvor w pokriva i neke druge grane. Preciznije, postoji **minimalni** pokrivač grana koji sadrži w.

Zato, ako izaberemo čvor w, uklonimo ga sa svim njemu susednim granama i indukcijom rešimo preostali problem, dolazimo do minimalnog pokrivača.

Vremenska složenost algoritma je proporcionalna broju čvorova u delu gde tražimo listove i broju grana u delu gde uklanjamo naslednike oca, tj. linearna je.

3.

Koristi se n(n-1)/2 procesora.

Svakom od datih brojeva x[i] pridruzujemo se po jedna (deljena) memorijska lokacija v[i]. Svakom paru {i,j} zadatih brojeva pridružuje se procesor P[i][j].

Najpre se u sve lokacije v[i] upisuje vrednost 1.

U sledećem koraku algoritma, procesor P[i][j] poredi elemente x[i] i x[j];

Ako je jedan od njih manji onda u njegovu odgovarajuću lokaciju upisuje se vrednost 0;

ako su x[i] i x[j] jednaki, onda ako je i<j onda se 0 upisuje u v[i], a inače se 0 upisuje u v[j].

Nakon toga samo u jednoj od lokacija v[i] ostaje upisana vrednost 1.

Otkrije se kom zadatom elementu je pridružena ta lokacija i taj broj je maksimum zadatih vrednosti.

Vreme izvršavanja algoritma je O(1) i on zahteva $O(n^2)$ procesora

4.

Neka su A i B dve proizvoljne kvadratne matrice reda n .

Svaka od njih se moze predstaviti kao zbir po jedne gornje i po jednedonje trougaone matrice(T_A, B_A, T_B, B_B, redom):

$A= T_A + B_A$

$B= T_B + B_B$

Dalje je : $AB=(T_A + B_A) (T_B + B_B)=T_AT_B + B_AB_B + B_AT_B + T_AB_B$

Ako se upotrebi algoritam iz formulacije zadataka moguce je izracunati proizvod :

$$\begin{bmatrix} B_A & 0 \\ T_A & B_A \end{bmatrix} \times \begin{bmatrix} B_B & 0 \\ T_B & B_B \end{bmatrix} = \begin{bmatrix} B_A B_B & 0 \\ T_A B_B + B_A T_B & B_A B_B \end{bmatrix}$$

Kao rezultat dobija se matrica koja sadrzi blokove: $B_A B_B$, $T_A T_B$, $B_A T_B + T_A B_B$
Ovi blokovi ucestvuju u izracunavanju proizvoda $A \times B$.
Matrice $(T_A)^T$ i $(T_B)^T$ su donje trougaone matrice, te se upotrebom datog algoritma moze izracunati blok $T_A T_B$ na sledeci nacin:
$T_A T_B = ((T_B)^T (T_A)^T)^T$
Ukupno vreme izvršavanja je $O(T(2n)+n^2) = O(T(n)+n^2)$.
Na taj nacin se problem izracunavanja proizvoda proizvoljne dve matrice svodi na problem izracunavanja proizvoda dve kvadratne donje trougaone matrice.