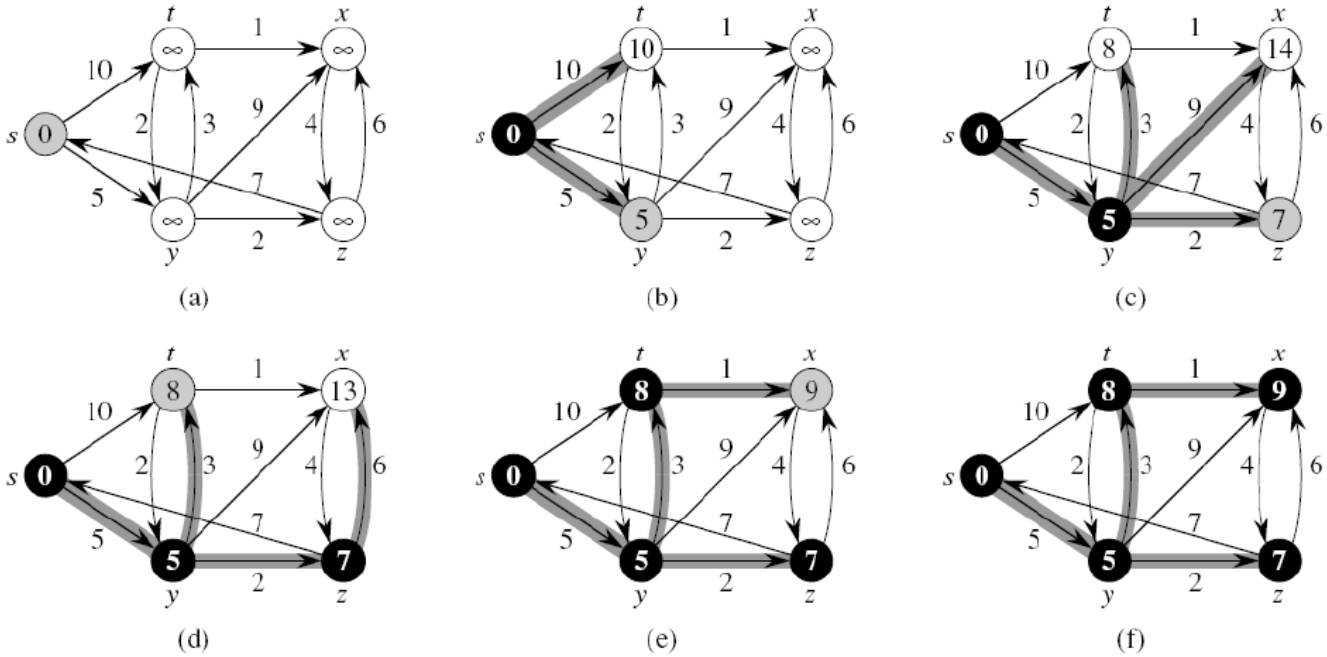
Slavisa Krstic 70/08

Dijkstra algoritam

Dijkstra algoritam resava problem najkraceg puta iz jednog polazista na tezinskom, usmerenom grafu G = (V,E) u slucaju kada su sve veze nenegativne.



Formiramo matricu rastojanja ovog grafa. Cvor od koga odredjujemo rastojanje postavimo na 0 i promenljivu provera postavmo na 1. Ostalim cvorovima promenljivu provera postavimo na 0.

Funkcija UnosMatriceN i UnosMatriceP sluzi za formiranje matrice rastojanja. UnosMatriceN se poziva ukoliko je n neparan broj. Ukoliko je paran broj poziva se UnosMatriceP zato sto program pravi razliku izmedju parnog i neparnog n.

Funkcija IspisNiza sluzi da ispisemo rastojanja izmedju prvog i svih ostali elemenata. Na prvom mestu u nizu se nalazi 0 tj. cvor za koji trazimo rastojanje do svih ostalih. Ostali brojevi u nizu predstavljaju rastojanje od pocetnog cvor do sebe.

Funkcija Zauzmi sluzi da promenljivu provera postavi na 1 i samim ti taj cvor postaje stalan. Taj cvor se ne moze vise menjati posto je on dobio najmanje rastojanje od zadatog cvora.

Funkcija Minimum odredjuje najmanji cvor u nizu i njega vraca. Najmanji cvor se postavlja za stalni cvor tako sto se pozove funkcija zauzmi.

Funkcija NoviNizU radi tako to se posalje niz u u kome se nalaze trenutno najkraca rastojanja i niz w. Od ta dva niza dobijamo novi niz U u kome se nalaze najkraca rastojanja.

Funkcija FUNKCIJA ima zadatak da poziva ostale funkcije i rezervise memoriju.

```
#include <stdio.h>
#include <stdlib.h>

struct STRUKTURA
{
  int Broj;
  int Provera;
};

void UnosMatriceN(struct STRUKTURA ***Matrica, int n)
{
  struct STRUKTURA **m;
  int i,j;
  m = (struct STRUKTURA **)malloc(n * sizeof(struct STRUKTURA *));
  for(i = 0; i < n; i++)
    m[i] = (struct STRUKTURA *)malloc(n * sizeof(struct STRUKTURA));
  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
    {
      scanf("%d",&m[i][j].Broj);
      if(m[i][j].Broj == 0)
        m[i][j].Broj = -1;
      m[i][j].Provera = 0;
    }
  *Matrica = m;
}

void UnosMatriceP(struct STRUKTURA ***Matrica, int n)
{
  struct STRUKTURA **m;
  int i,j;
  m = (struct STRUKTURA **)malloc(n * sizeof(struct STRUKTURA *));
  for(i = 0; i < n; i++)
    m[i] = (struct STRUKTURA *)malloc(n * sizeof(struct STRUKTURA));

  for(i = 0; i < n-1; i++)
    for(j = 0; j < n-1; j++)
    {
      scanf("%d",&m[i][j].Broj);
      if(m[i][j].Broj == 0)
        m[i][j].Broj = -1;
      m[i][j].Provera = 0;
    }

  for(i = n-1; i < n; i++)
    for(j = 0; j < n; j++)
    {
```

```
        m[i][j].Broj = -1;
        m[j][i].Broj = 0;
        m[i][j].Provera = 0;
        m[j][i].Provera = 0;
      }
   *Matrica = m;
}

void Zauzmi(struct STRUKTURA **Matrica, int z, int n)
{
   int i;
   for(i = 0; i < n; i++)
      Matrica[i][z].Provera = 1;
}

int Minimum(int *z,struct STRUKTURA *u, int n)
{
   int i, Min;
   for(i = 0; i < n; i++)
      if(u[i].Broj != -1 && u[i].Provera != 1);
      {
         Min = u[i].Broj;
         *z = i;
      }
    for(i = 0; i < n; i++)
       if(u[i].Broj != -1 && u[i].Provera == 0 && Min > u[i].Broj)
       {
          Min = u[i].Broj;
          *z = i;
       }
   return Min;
}

void NovNizU(struct STRUKTURA *u, struct STRUKTURA *w, int min, int n)
{
   int i;
   for(i = 0; i < n; i++)
   {
      if(w[i].Provera == 0 && w[i].Broj != -1)
      {
         if(w[i].Broj + min < u[i].Broj || u[i].Broj == -1)
            u[i].Broj = w[i].Broj + min;
      }
   }
}
```

```
void IspisNiza(struct STRUKTURA *U, int n, int m)
{
  int i;
  if(m == 0)
  {
    for(i = 0; i < n; i++)
      printf("%d ",U[i].Broj);
    printf("\n");
  }
  else
  {
    for(i = 0; i < n-1; i++)
      printf("%d ",U[i].Broj);
    printf("\n");
  }
}

void Funkcija(struct STRUKTURA **Matrica, int n, struct STRUKTURA **U)
{
  struct STRUKTURA *u, *w;
  int z = 0;
  int Vrsta = 0;
  int i, l, min;
  u = (struct STRUKTURA *)malloc(n * sizeof(struct STRUKTURA));
  w = (struct STRUKTURA *)malloc(n * sizeof(struct STRUKTURA));
  Zauzmi(Matrica, z, n);
  u[0].Broj = 0;
  u[0].Provera = 1;
  for(i = 1; i < n; i++)
  {
    u[i].Broj = Matrica[Vrsta][i].Broj;
    u[i].Provera = Matrica[Vrsta][i].Provera;
  }
  for(l = 1; l < n; l++)
  {
    min = Minimum(&z,u,n);
    Vrsta = z;
    Zauzmi(Matrica, z, n);
    u[z].Provera = 1;
    for(i = 0; i < n; i++)
    {
      w[i].Broj = Matrica[Vrsta][i].Broj;
      w[i].Provera = Matrica[Vrsta][i].Provera;
    }
    NovNizU(u,w,min,n);
  }
  *U = u;
}
```

```
main()
{
  struct STRUKTURA **Matrica;
  int n;
  int m = 0;
  printf("Unesite velicinu matrice rastojanja: ");
  scanf("%d",&n);

  if(n % 2 == 1)
  {
    UnosMatriceN(&Matrica, n);
  }
  else
  {
    n = n + 1;
    m = 1;
    UnosMatriceP(&Matrica, n);
  }

  struct STRUKTURA *U;
  Funkcija(Matrica, n, &U);
  IspisNiza(U,n,m);
}
```