Algoritmul lui Prim

Roxana Macarie
Diana Serbanescu
clasa a XI-a F

Algoritmul lui Prim

Roxana Macarie
Diana Serbanescu
clasa a XI-a F

Algoritmul lui Prim

Algoritmul lui Prim este un algoritm din teoria grafurilor care gaseste arborele partial de cost minim al unui graf conex ponderat.

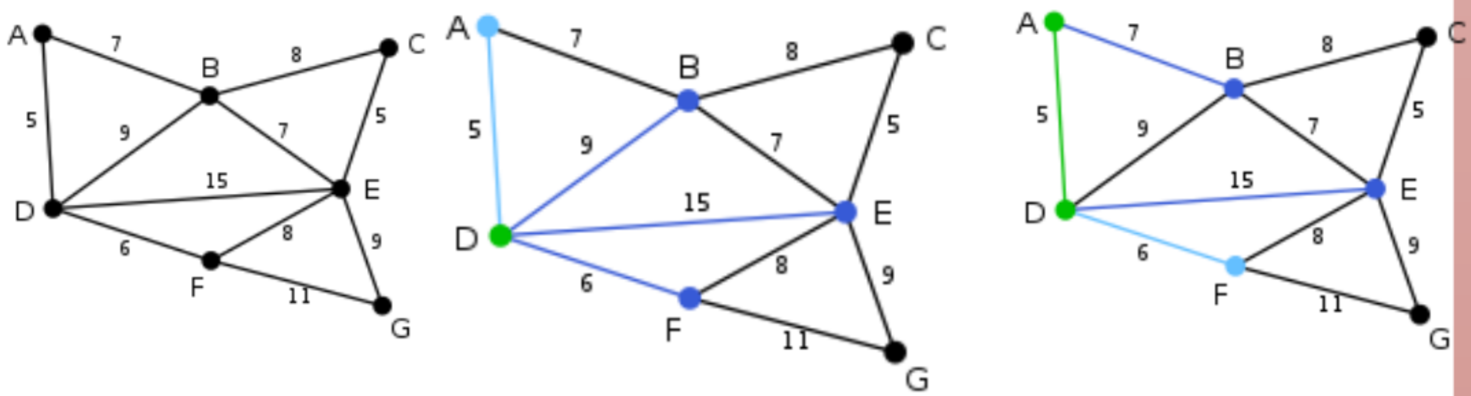
Se numeste graf ponderat o structura (V,E,W), unde:

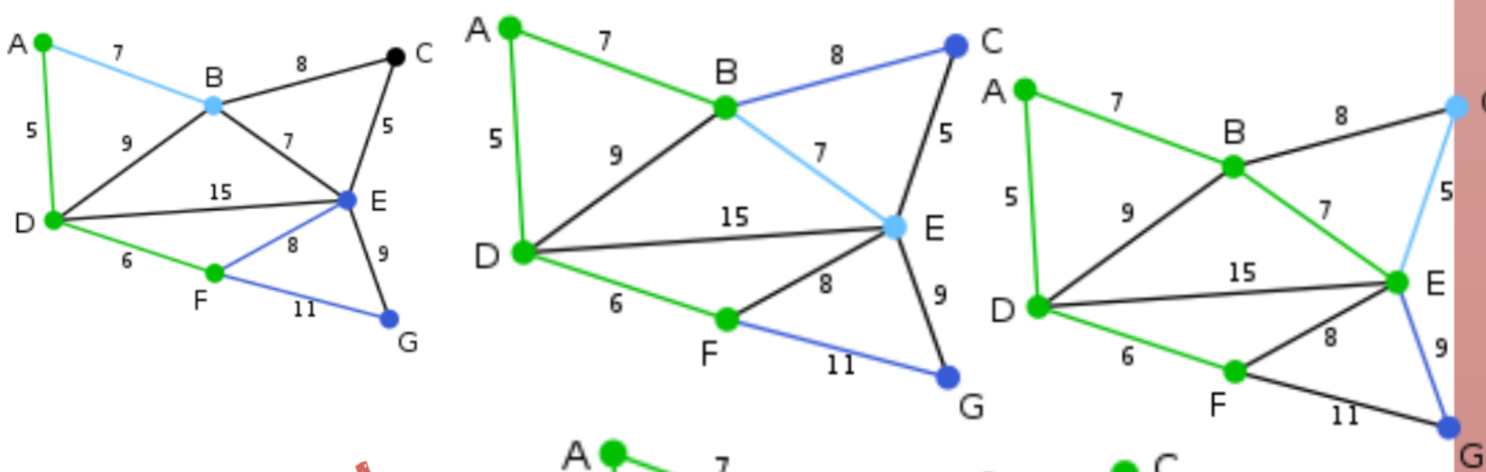
- G=(V,E) este graf
- W este o functie numita pondere, care asociaza fiecarei muchii a grafului un cost/castig al parcurgerii ei.

Cun functioneaza?

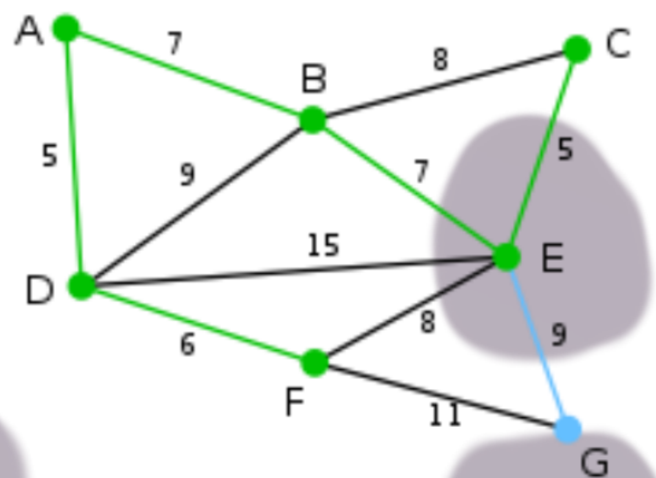
Arborele porneste de la un vârf arbitrar r si creste pana acopera toate vârfurile din V. La fiecare pas se adauga arborelui o muchie de lungime minima care uneste un vârf al arborelui A cu un vârf neselectat pâna la acel pas.

Asociem fiecarui vârf neselectat o cheie reprezentând costul minim al unei muchii care îl uneste de un vârf deja selectat. La fiecare selectare a unui nou vârf pentru a fi adugat în arbore, se actualizeaza cheile vârfurilor adiacente cu el ramase neselectate.


A B C D E F G
7 8 5 9 7 5 15 6 8 9 11
A B C D E F G
7 8 5 9 7 5 15 6 8 9 11
A B C D E F G
7 8 5 9 7 5 15 6 8 9 11


A B C D E F G
7 8 5 9 7 5 15 6 8 9 11
A B C D E F G
7 8 5 9 7 5 15 6 8 9 11
A B C D E F G
7 8 5 9 7 5 15 6 8 9 11

Mai mult...


A B C D E F G
7 8 5 9 7 5 15 6 8 9 11

Sa intelegem mai bine!

G- graf conex
r- radacina
cheie[v]- minimul costurilor muchiilor care unesc pe v cu vârfurile selectate (din arborele A)
p[v]- varful din arbore pentru care se realizeaza acest minim

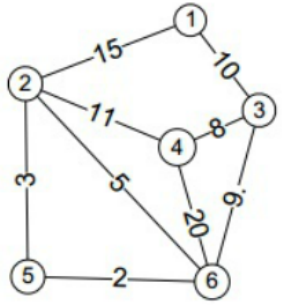
Astfel, la fiecare pas, muchia (v, p[v]) va fi muchia de cost minim care il uneste pe v de un vârf selectat, iar lungimea acestei muchii este cheie[v] si reprezinta distanta minima de la v la arborele A.

cheie[v]=+oo daca nu exista nici o muchie care sa il uneasca pe v de un vârf selectat.

PSeudoCod

```
creare(A,v[G])
pentru fiecare u din A executa:
    cheie[u]=+oo
cheie[r]=0
p[r]=null
cat timp A diferit de 0 executa:
    u=extragereMinim(A)
    pentru fiecare v din A executa:
        daca v apartine A si
        w(u,v)<cheie[v] atunci
            p[v]=u
            cheie(v)=w(u,v)
```

Exemplu:



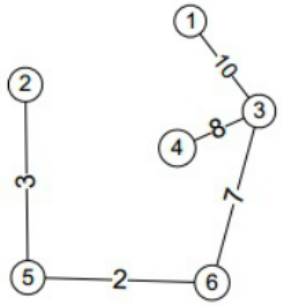
Vârf de pornire – 1
Marcăm cu * vârful selectat la pasul curent

vârf	1	2	3	4	5	6
Cheie/p	0/null *	∞/null	∞/null	∞/null	∞/null	∞/null
	-	15/1	10/1 *	∞/null	∞/null	∞/null
	-	15/1	-	8/3 *	∞/null	9/3
	-	11/4	-	-	∞/null	9/3 *
	-	5/6	-	-	2/6 *	-
		3/5 *				

După ce au fost selectate toate vârfurile, muchiile se pot reconstitui ținând cont de părintele p la momentul selectării. Astfel, muchiile vor fi

(5, 2) - deoarece p[2] = 5
(1, 3) - deoarece p[3] = 1
(3, 4) - deoarece p[4] = 3
(6, 5) - deoarece p[5] = 6
(3, 6) - deoarece p[6] = 3

Obținem arborele următor, cu costul 30.



Algoritm C++

```
#include<stdio.h>
int main()
{int n,i,j,x;
freopen("prim.in","r",stdin);
freopen("prim.out","w",stdout);
scanf("%d",&n);
scanf("\n");
   int c[n][n],ni[n];
   for(i=1;i<=n;i++)
                  {
                  for(j=1;j<=n;j++)scanf("%d",&c[i][j]);
                  scanf("\n");
                  }
   for(i=1;i<=n;i++)
                  for(j=i+1;j<=n;j++)
                                 if(c[i][j]>0){
                                        printf("%d -- %d   cost=%d
                                 \n",i,j,c[i][j]);
                                        }

   int nr[n];
   for(i=1;i<=n;i++)
     {
     nr[i]=0;
     for(j=1;j<=n;j++)
                  if(c[i][j]!=0)
                        nr[i]++;
     }

   int max=nr[1],init=1; // ni - nod initial,
   for(i=1;i<=n;i++)if(nr[i]>max)
                                 {
                                 init=i;
                                 max=nr[i];
                                 }
```

```
 ni[1]=init;
   int nrn=1, cf=0; // cf - cost final, nrn - este nr de noduri care se vor
include la fiecare pas,
                  // se autoincrementeaza la fiecare nod inclus

   do
    {
    int min=0, x=0, y=0; // x - nodul de plecare ,y - nodul de sosire
    for(i=1;i<=nrn;i++)
        if(c[ni[i]][j]!=0&&min==0)
        { //gasirea primei muchii a nodului curent care il uneste cu alt nod
         for(j=1;j<=n;j++)
                  { min=c[ni[i]][j];
                    y=j; // y pastreaza numele nodului gasit
                    x=i;
                    continue;
                  }
        if(c[ni[i]][j]!=0&&c[ni[i]][j]<min){
                  min=c[ni[i]][j];
                  y=j;
                  x=i;
                  }
        }
    if(min>0)
        { printf("%d -- %d   cost=%d \n",ni[x],y,c[ni[x]][y]);
          cf+=c[ni[x]][y];
          for(i=1;i<=nrn;i++){
                  c[ni[i]][y]=0;
                  c[y][ni[i]]=0;
                  }
         nrn++;
         ni[nrn]=y;
        }
   }while(nrn<n);
printf("Cost final=%d",cf);
return 0;
}
```

```
#include<stdio.h>
int main()
{int n,i,j,x;
freopen("prim.in","r",stdin);
freopen("prim.out","w",stdout);
scanf("%d",&n);
scanf("\n");
    int c[n][n],ni[n];
    for(i=1;i<=n;i++)
                    {
                    for(j=1;j<=n;j++)scanf("%d",&c[i][j]);
                    scanf("\n");
                    }
    for(i=1;i<=n;i++)
                    for(j=i+1;j<=n;j++)
                                    if(c[i][j]>0){
                                            printf("%d -- %d   cost=%d
                                     \n",i,j,c[i][j]);
                                                    }
    int nr[n];
    for(i=1;i<=n;i++)
       {
       nr[i]=0;
       for(j=1;j<=n;j++)
                    if(c[i][j]!=0)
                        nr[i]++;
       }

    int max=nr[1],init=1; // ni - nod initial,
    for(i=1;i<=n;i++)if(nr[i]>max)
                                {
                                init=i;
                                max=nr[i];
                                }
```

```
 ni[1]=init;
   int nrn=1, cf=0; // cf - cost final, nrn - este nr de noduri care se vor
include la fiecare pas,
                  // se autoincrementeaza la fiecare nod inclus

   do
    {
    int min=0, x=0, y=0; // x - nodul de plecare ,y - nodul de sosire
    for(i=1;i<=nrn;i++)
       if(c[ni[i]][j]!=0&&min==0)
       { //gasirea primei muchii a nodului curent care il uneste cu alt nod
        for(j=1;j<=n;j++)
              { min=c[ni[i]][j];
                y=j; // y pastreaza numele nodului gasit
                x=i;
                continue;
              }
       if(c[ni[i]][j]!=0&&c[ni[i]][j]<min){
              min=c[ni[i]][j];
              y=j;
              x=i;
              }
        }
    if(min>0)
        { printf("%d -- %d   cost=%d \n",ni[x],y,c[ni[x]][y]);
          cf+=c[ni[x]][y];
          for(i=1;i<=nrn;i++){
               c[ni[i]][y]=0;
               c[y][ni[i]]=0;
               }
          nrn++;
          ni[nrn]=y;
        }
   }while(nrn<n);
 printf("Cost final=%d",cf);
 return 0;
}
```