algoritam.blog.com.mk
2007

Примов алгоритам

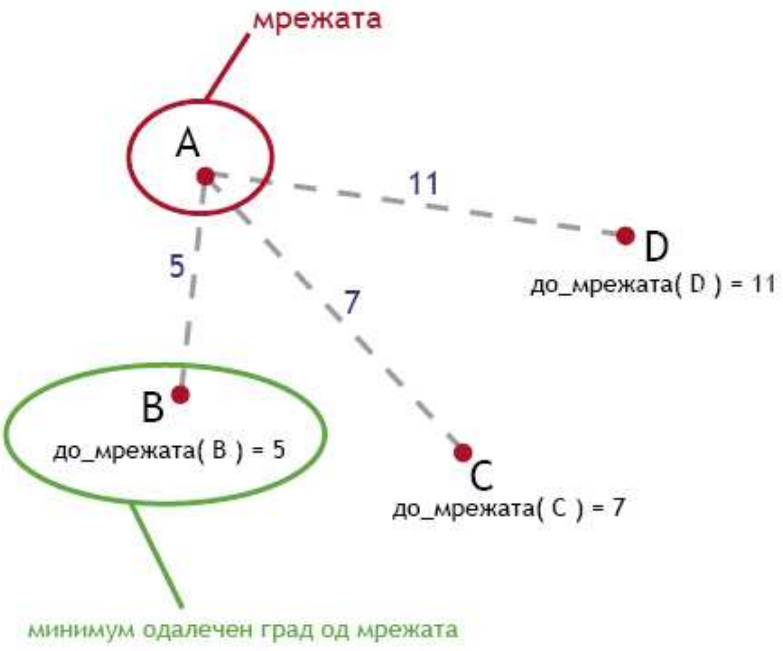
Примовиот алгоритам ќе го разгледаме преку следниов проблем:

Дадени се градови низ Македонија. Владата сака да изгради најефтина мрежа од патишта, така што сите градови ќе бидат поврзани. Патиштата се двонасочни. Значи, мрежата може да има произволна структура, важно е сумата пари што ќе се потрошат да биде минимална, а целта да биде исполнета – градовите да бидат поврзани.

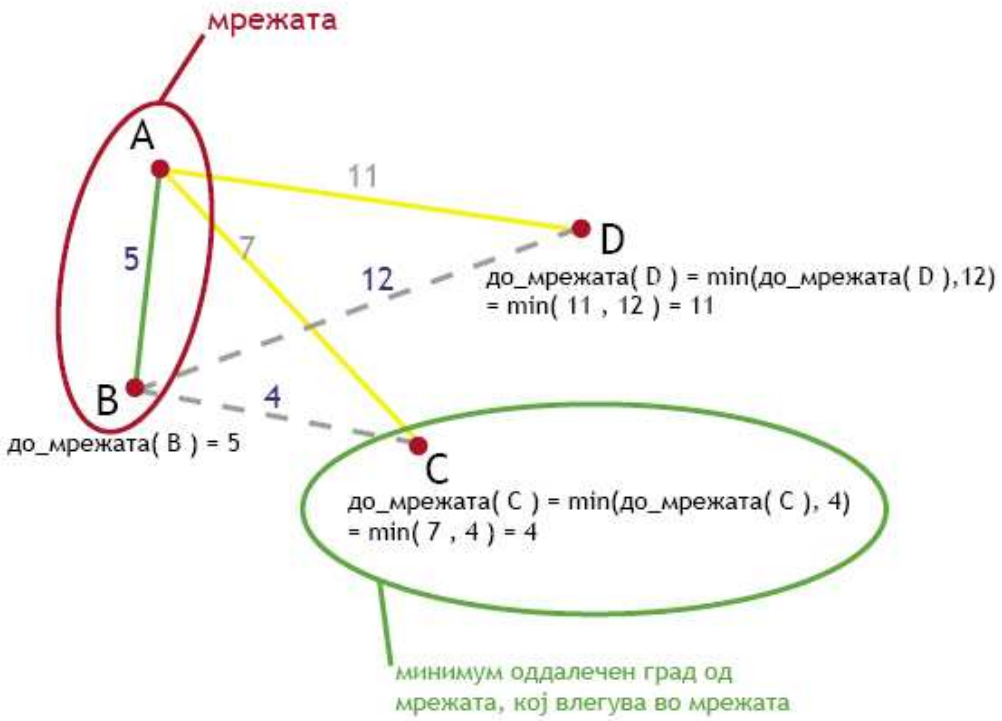
Решение:

Патиштата може да бидат изградени помеѓу сите можни парови на градови. Значи на располагање ги имаме сите можни врски. Но ние треба да одбереме некои од нив, за да ја направиме мрежата.

Нека се дадени вкупно четири града и тоа: **A**, **B**, **C**, **D**.



На почеток, во мрежата го поставуваме само еден од понудените градови. Нека тоа биде градот **A**. И бидејќи градот **A** е единствениот град во мрежата, за другите темиња кои не се во мрежата, го пресметуваме растојанието до градот **A**, односно до мрежата (бидејќи во мрежата се наоѓа само градот **A** !). Значи секој град, со себе ќе носи променлива **ДО_МРЕЖА** која ќе ни кажува, колкаво е минималното растојание од тој град до мрежата.



Откако сме ги пресметале променливите **ДО_МРЕЖА** кај сите од оние градови кои не се во мрежата, кон мрежата ќе го приклучиме оној град кој има минимална вредност за одалеченост до мрежата односно оној град кој има минимална вредност на **ДО_МРЕЖА**. Да речеме дека тоа ќе биде градот(темето) **B**. Бидејќи градовите **D** и **C** веќе имаат вредности за нивното минимално растојание до мрежата (само до градот **A**), но бидејќи и градот **B** станува дел од мрежата, мора вредностите на **ДО_МРЕЖА** за секој од темињата **D** и **C** да бидат повторно пресметани. (На пример, темето **D**: **ДО_МРЕЖА(D)** претходно има вредност - растојание од **A** до **D**, но бидејќи ставивме нов град во мрежата, **ДО_МРЕЖА(D) = MIN(ДО_МРЕЖА(D),BD)**).

Постапката продолжува се додека сите градови не станат дел од мрежата.

Реализација на алгоритмот во програмскиот јазик Паскал:

```
program primov_alg;
uses crt;
type
    teme=record
               x:integer;{x-koordinata na temeto}
               y:integer;{y- koordinata na temeto}
               kade:integer;{indeksot na temeto do koe ova teme e najmalku odaleceno
do tekovniot moment na proverka}
               do_mreza:real;{rastojanieto megu i-toto teme i temeto so indeks kade}
               vo_mreza:boolean;{dali temeto e vo mrezata ili ne}
    end;

var
   tocki:array[1..100] of teme;
   vk_pat:real;
   n:integer;

procedure citaj;
```

```
var
   i:integer;
begin
     writeln('Vnesi go brojot na teminja: ');
     readln(n);
     writeln('Vnesi gi teminjata kako podredeni praovi: ');
     for i:=1 to n do
         begin
              readln(tocki[i].x,tocki[i].y);
              tocki[i].do_mreza:=0;
              tocki[i].kade:=0;
              tocki[i].vo_mreza:=false;
         end;
end;

function dest(i,j:integer):real;
begin
     dest:=sqrt(sqr(tocki[i].x-tocki[j].x)+sqr(tocki[i].y-tocki[j].y));
end;

procedure izvrsi;
var
   pos,l_min,i:integer;
   brojac:boolean;
begin
     tocki[1].vo_mreza:=true;{kako poceto teme vo mrezata go zemame prvo teme}
     pos:=1;{prvoto teme e posledno teme vo mrezata}
     vk_pat:=0;{vkupniot pat, na pocetok iznesuva 0}
     uste_tocki:=true;{postojat uste teminja koi treba da se stavat vo mrezata}
     for i:=2 to n do{na pocetok, za ostanatite teminja, go prsmetuvame rastojanieto
do poslednoto staveno teme vo mrezata}
         begin
               tocki[i].do_mreza:=dest(pos,i);
               tocki[i].kade:=pos;{rastojanieto do tekovnata mreza(samo teme 1) e
rastojanie megu i-toto teme i temeto so indeks pos}
         end;
     while uste_tocki do{dodeka ima uste teminja}
           begin
                uste_tocki:=false;{pretpostavuvame deka nema teminja koi treba da se
stavat vo mrezata}
                l_min:=0;{neka na pocetok, indeksot na temeto koe e najmalku
odaleceno od mrezata e 0}
                for i:=1 to n do
                    if not tocki[i].vo_mreza{ako i-toto teme ne e vo mrezata}
                       then
                           begin
                                if (tocki[i].do_mreza>dest(pos,i)){ako dotogasnoto
rastojanie do mrezata e pogolemo od rastojanieto megu poslednoto teme vo mrezata i i-
toto teme}
                                   then
                                       begin
                                            tocki[i].do_mreza:=dest(pos,i);
{rastojanieto na i-toto teme do mrezata prima nova vrednost}
                                            tocki[i].kade:=pos;{se zapisuva do koe
teme e vsusnost toa rastojanie}
                                       end;
                                 if uste_tocki
                                   then
                                    if (tocki[l_min].do_mreza>tocki[i].do_mreza)
{ako l_min temeto e podaleku od mrezata otkolku i-toto teme i imame barem edno
pominato teme}
```

```
                                    then
                                        l_min:=i;{indeksot na najmalku odalecenato
teme ja zema vrednosta i}
                                  if not uste_tocki{ako ova e prvo teme na ispituvanje,
bidejki l_min ima vrednost 0 togas}
                                    then
                                        begin
                                             l_min:=i;
                                             uste_tocki:=true;{veke pominavme edno
teme koe ne e vo mrezata}
                                        end;

                              end;
                  if uste_tocki
                    then
                          begin
                               vk_pat:=vk_pat+tocki[l_min].do_mreza;{na vkupniot pat
go dodavame rastojanieto megu najmalku odalecenoto teme do mrezata}
                               pos:=l_min;
                               tocki[l_min].vo_mreza:=true;{ona teme sto bese najmalku
odaleceno go stavame vo mrezata}
                          end;
             end;
end;

procedure pecati;
begin
     writeln('Najeftinata mreza na patista iznesuva ',vk_pat:7:2,' kilometri');
end;

begin
     clrscr;
     citaj;
     izvrsi;
     pecati;
     readln;
end.
```