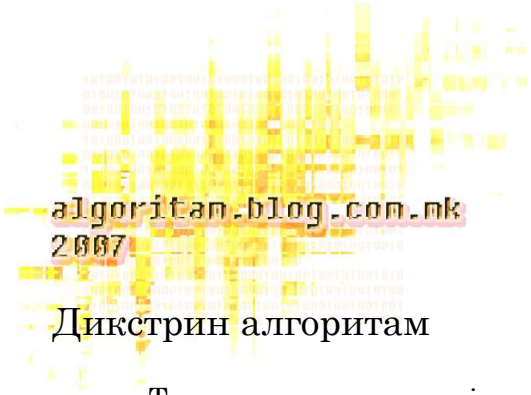


Дикстрин алгоритам

Тоа е алгоритам кој за даден граф (V,E) и дадено теме ***u*** ги одредува најкратките(минималните) патишта од темето ***u*** до секое останато теме. Кај овој алгоритам се манипулира со два податока: множество темиња до кои е одреден најкраткиот пат (ова множество ќе го означиме со U), и за секое теме ***v*** од множеството ***V\U***, должината на најкраткиот пат од темето ***u*** до темето ***v***, кој минува низ темиња од множеството ***U*** (се мисли на темињата меѓу ***u*** и ***v***, а не и на темето ***v***). Ќе ја означиме должината на тој пат со ***s(v)***. Доколку не постои нити еден пат до темето ***v***, бројот ***s(v)*** ќе биде некој доволно голем број, така што нема да постои ниеден пат чија должина е подолга од таа вредност. Патиштата кои ќе поминуваат низ темиња од множеството ***U***, така што само крајот на патиштата ќе биде надвор од множеството ***U***, ќе ги нарекуваме *специјални патишта*.

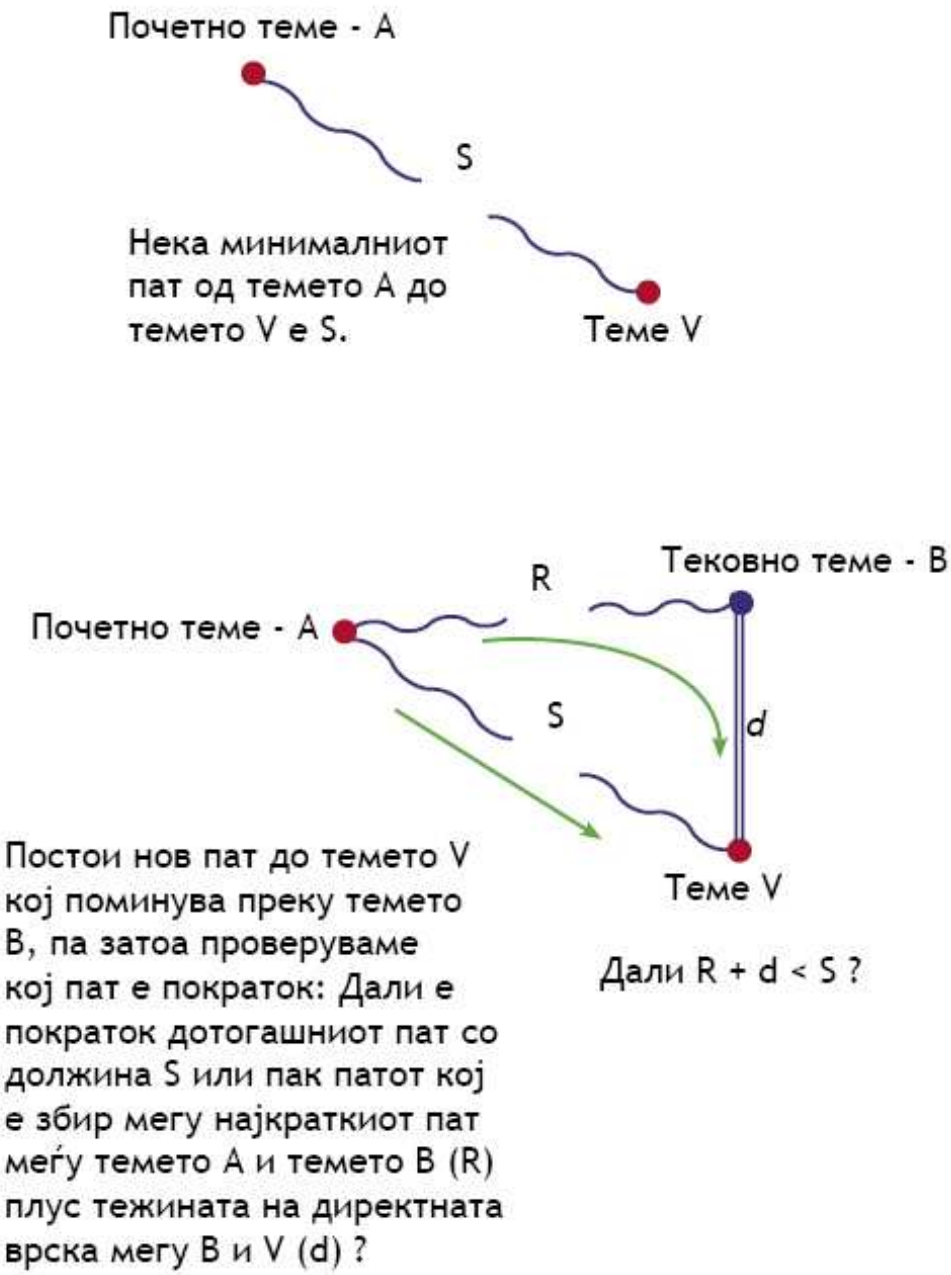
Во почетниот момент ***U={u}***. Специјални патишта постојат само до темињата кои се директно поврзани со темето ***u*** и нивните должини се еднакви на должините на ребрата кои ги поврзуваат овие темиња со ***u***.

Во секој чекор се изведуваат следниве подчекори:

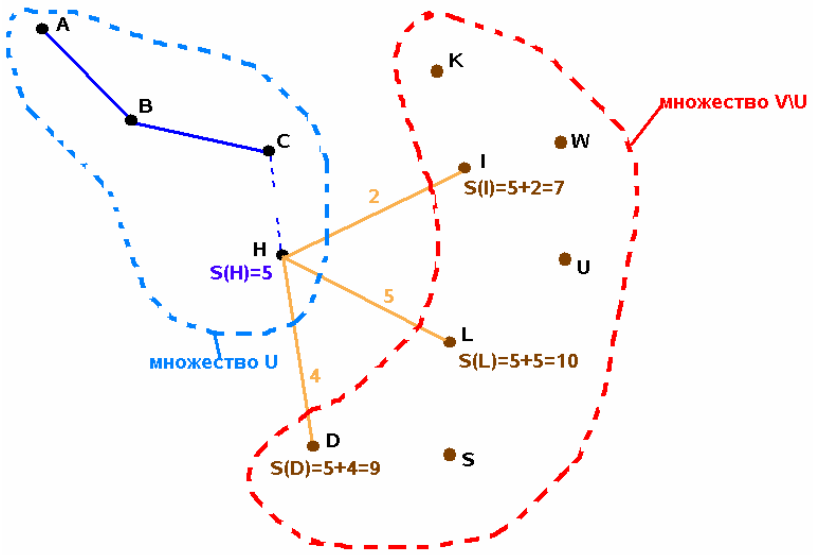
1. Се одредува теме ***w*** од множеството ***V\U*** таков што за него, должината на специјален пат е минимална. Го вклучуваме тоа теме ***w*** во множеството ***U*** (**U = U** U **{w}**) . Со тоа сме го определиле најкраткиот пат од темето ***u*** до темето ***w*** и неговата должина е ***s(w)***.
2. За сите темиња ***v*** од множеството ***V\U***, го ажурираме бројот s така што ***s(v)=min{s(v),s(w)+d(w,v)},*** каде ***d(w,v)***, е еднаков на должината на реброто кое ги спојува темињата ***w*** и ***v***. Јасно е дека најкраткиот специјален пат е дотогашниот најкраток специјален пат, или најкраткиот специјален пат до темето ***w*** продолжен за реброто од темето ***w*** до темето ***v***.

Оваа постапка се повторува се додека сите темиња не ги приклучиме во множеството ***U***.

Пред да дадеме конкретна реализација на Дикстриниот алгоритам, ќе дадеме уште некои објаснувања.
Кога се задава графот преку темиња и ребра со тежини, тежините не мора секогаш да претставуваат оддалеченост. На пример, јазлите може да претставуваат системи, а ребрата меѓу нив, директни врски. Потоа тежината кај секое ребро може да означува загуба на квалитет на сигнал. Па притоа Дикстриниот алгоритам можеме да го употребиме да се најде пат меѓу две темиња со ***најмала(минимална)*** сумарна загуба кај квалитетот на сигнал. Притоа може да се наложуваат и други ограничувања кои би го направиле проблемот нереален.



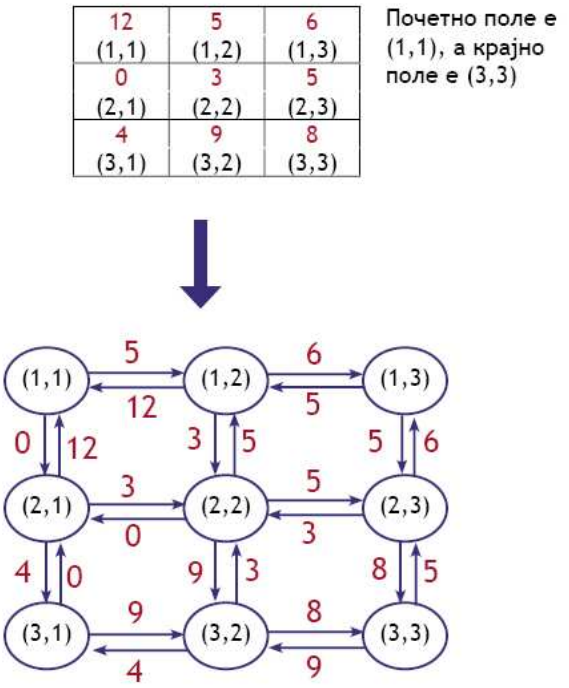
Слика 1: Главната идеја кај Дикстриниот алгоритам



Слика 2: Чекор кај Дикстринот алгоритам

Објаснување на слика 2.
Значи темето H сме го вклучиле во множеството U, па според тоа најкраткиот пат од темето A до H е веќе определен и изнесува 5. Потоа за секое теме од V\U повторно ги калкулираме нивните S, и како следно теме за во U, го бираме она теме од V\U, кое ќе има најмала вредност за S меѓу темињата од V\U.

Друга употреба на Дикстриниот алгоритам е кај следниов проблем:
На пример, имаме матрица со димензии m x n. Во секое поле од матрицата имаме некаква позитивна вредност. Притоа, сакаме да тргнеме од некое поле со индекси i, j и да дојдеме до поле со индекси p, q но така што збирот од вредностите по тој пат да биде минимален. Овој алгоритам е применлив во наведениов проблем бидејќи матрицата може да се претстави(трансформира) преку граф на следниов начин:



Слика 3: Трансформација на матрица во граф

Има уште една работа која е карактеристична и многу важна за Дикстриниот алгоритам. Тежините на ребрата мора да бидат ненегативни вредности т.е. поголеми од или еднакви на 0. Во случај на негативни тежини се користи друг алгоритам, како што е Флојдовиот алгоритам (види во [2],[3],[4]).

Сложеноста на Дикстриниот алгоритам е од ред $O(n^2)$, каде n е број на темиња во графот.

Реализација на Дикстриниот алгоритам во програмскиот јазик Pascal:

```
program dikstra;
uses crt;
var
   tezini:array [1..100,1..100] of integer;{matrica za tezinite na vrskite megu
teminjata}
   pominato:array[1..100] of boolean;{niza od indikatori za teminjata, koi kazuvaat
   dali do odredeno teme e najden minimalniot pat od temeto poc}
   minPat:array[1..100] of longint;{niza za sledenje na minimalniot pat od poc temeto
do sekoe ostanato teme}
   odKade:array[1..100] of integer;{niza od vrednosti za sekoe teme koja kazuva od
koe teme poteknuva minimalniot pat}
   brTem,poc:integer;
   vlez:text;

procedure citaj;
var
   vrski,i,j,k,d:integer;
begin
     fillchar(pominato,sizeof(pominato),false);{se popolnuva celanata niza indikatori
so false}
     assign(vlez,'vlez.dat');
     reset(vlez);
     readln(vlez,brTem);{se cita brojot na teminja}
     for i:=1 to brTem do
         begin
             for j:=1 to brTem do
                 tezini[i,j]:=maxint;{za sekoja tezina megu teminjata se postavuva
najgolem broj sto postoi}
             minPat[i]:=maxlongint;{za minimalen pat se stava najgolem mozen broj sto
postoi}
         end;
     readln(vlez,vrski);{se cita brojot na rebra-vrski megu teminjata}
     for k:=1 to vrski do{se cita megu koi teminja postoi vrska i so kolkava tezina}
         begin
              readln(vlez,i,j,d);
              tezini[i,j]:=d;{bidejki grafot ne e orientiran vrskata megu i-toto teme
i j-toto teme e istata vrska megu j-toto teme i i-toto teme}
              tezini[j,i]:=d;
         end;
     readln(vlez,poc);{se cita pocetnoto teme vo odnos na koe
     ke se opredeluva najkratok pat do drugite teminja}
     close(vlez);
end;

procedure run;
var
   i,tek,lmin:integer;
   pon:boolean;
begin
     tek:=poc;{kako prvo teme se zema pocetnoto}
     fillchar(odKade,sizeof(odKade),poc);{cela niza odKade se polni so vrednosti poc}
```

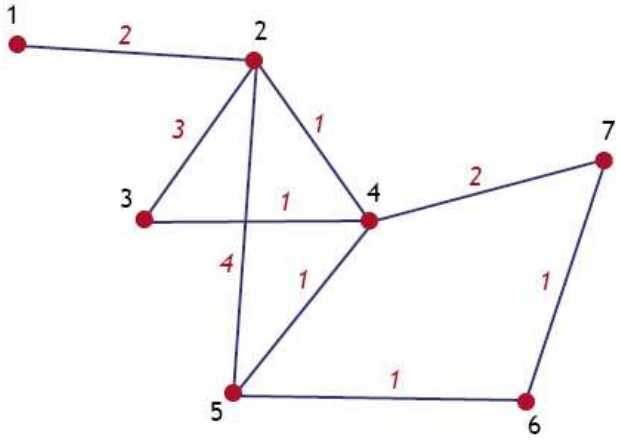
```
     minPat[tek]:=0;{minimaleniot pat od temeto poc do istoto teme poc e 0}
     pominato[tek]:=true;{znaci minimalniot pat do temeto poc od temeto poc e veke
odredeno; pa zatoa go oznacuvame kako pominato teme}
     pon:=true;
     while pon do
       begin
             pon:=false;
             lmin:=0;
             for i:=1 to brTem do{se pominuvaat site teminja}
                 begin
                      if  (not(pominato[i])  and  (tezini[i,tek]<maxint)){ako  i-toto
teme e nepominato i pritoa postoi vrska megu i-toto i tekovnoto teme}
                         then
                             begin
                                  if    minPat[i]>minPat[tek]+tezini[i,tek]    then{se
proveruva dali noviot minimalen pat, no koj sega doga preku tekovnoto teme e pokratok
od stariot}
                                     begin
                                          minPat[i]:=minPat[tek]+tezini[i,tek];{ako
toa e tocno se zapisuva vrednosta na noviot pat}
                                          odKade[i]:=tek;{koj  doaga  preku  tekovnoto
teme}
                                     end;
                             end;
                       if (not pominato[i]){za da go odredime slednoto teme so koe ke
prodolzi algoritmot od site nepominato teminja go birame ona so najmal minimalen pat
od temeto poc}
                         then
                             begin
                                  if lmin=0
                                     then
                                         lmin:=i
                                     else
                                         if minPat[lmin]>minPat[i]
                                            then
                                                lmin:=i;
                                  pon:=true;
                             end;
                 end;
             if pon
                then
                    begin
                         pominato[lmin]:=true;{go   oznacuvame   temeto   lmin   kako
pominato i toa e temeto koe ke go smetame kako tekovno vo slednata iteracija }
                         tek:=lmin;
                    end;
       end;
end;

procedure pecati;
var
   i:integer;
begin
     for i:=1 to brTem do
         writeln('Najkratkiot  pat  od  ',poc,'-to  teme  do  ',i,'-to  teme  iznesuva
',minPat[i]);
end;

begin
     clrscr;
     citaj;
     run;
```

```
     pecati;
end.
```

За влезна датотека



VLEZ.DAT

```
7
9
1 2 2
2 3 3
2 4 1
3 4 1
2 5 4
5 4 1
4 7 2
5 6 1
6 7 1
1
```

се добива следниов излез:

```
Najkratkiot pat od 1-to teme do 1-to teme iznesuva 0
Najkratkiot pat od 1-to teme do 2-to teme iznesuva 2
Najkratkiot pat od 1-to teme do 3-to teme iznesuva 4
Najkratkiot pat od 1-to teme do 4-to teme iznesuva 3
Najkratkiot pat od 1-to teme do 5-to teme iznesuva 4
Najkratkiot pat od 1-to teme do 6-to teme iznesuva 5
Najkratkiot pat od 1-to teme do 7-to teme iznesuva 5
```

Во приложената програма, за секое теме се памти информација за тоа од каде, поточно преку кое соседно теме доаѓа најкраткиот пат. На пример, за графот прикажан на сликата на оваа страна, најкраткиот пат од темето 1 до темето со број 7 изнесува 5. А тоа е всушност патот определен со темињата 1 – 2 – 4 – 7. Значи за темето со број 7, најкраткиот пат за ова теме доаѓа преку темето со број 4 т.е. odKade[7] = 4. Аналогно odKade[4] = 2 и odKade[2] = 1. Овие информации можат да ни послужат како водич при трасирање на најкраткиот (најоптималниот) пат меѓу две темиња.

Код за трасирање на најкраток пат меѓу темињата нумерирани со A1(почетно теме) и A2(крајно теме):

```
writeln(A2);
tek:=A2;
while (tek <> A1) do
      begin
            writeln(odKade[tek]);
            tek:=odKade[tek];
      end;
```

Доколку крајното теме до кое треба да се определи најоптималниот пат, е експлицитно зададено, тогаш Дикстриниот алгоритам завршува во моментот кога крајното теме ќе биде означено(pominato).

Користена литература

[1] *Takmičenja iz informatike od 1995 do 1998 godine* – Dragan Urošević, Krug 1998;
[2] *Introduction to algorithms* – Thomas H. Corman, Charles E. Leiseron, Ronald L. Rivest, Clifford Stein; MIT Press, 2001;
[3] *Теорија на графови* – Душан Чакмаков, ИнФорма 2002;
[4] *Discrete mathematics* – Kenneth A. Ross, Charles R.B. Wright, Prantice Hall.