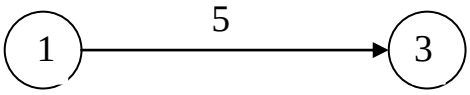
Знаходження найкоротшого шляху у графі за алгоритмом Дійкстри .

До уваги читачів пропонується алгоритм та текст програми знаходження найкоротшого шляху у зваженому орієнтованому графі .

Нагадаємо, що граф називається зваженим, якщо кожному його ребру відповідає певне числове начення . Наприклад ребру (1,3) відповідає вага 5. (див. мал. 1) .

Мал. 1. 

Будемо називати граф орієнтованим, якщо вказано напрям переходу від однієї вершини графа до іншої (див. мал. 1) .

Перший ефективний алгоритм побудови найкоротшого шляху у графі з невід'ємною вагою запропонував Е . Дійкстра у 1959 році . Суть цього алгоритму полягає у слідуючому .

На початку розглянемо термінологію, яку будемо використовувати при описі алгоритму .

Кожна вершина графа може мати три стани : виділена, невиділена та активна. **Виділеною** вершиною будемо називати будь-яку вершину, яка при виконанні алгоритму пройшла певну обробку і вподальшому не використовується . **Активною** вершиною будемо називати вершину з якої шукається шлях до іншої вершини .

Всі вершини, які не є виділеними і не є активною вершиною будемо вважати **невиділеними .**

Позначення кожного стану вершин показані на малюнку 2 .

Мал .2.

Крім того кожна вершина графа має дві характеристики : вага вершини та НВП .

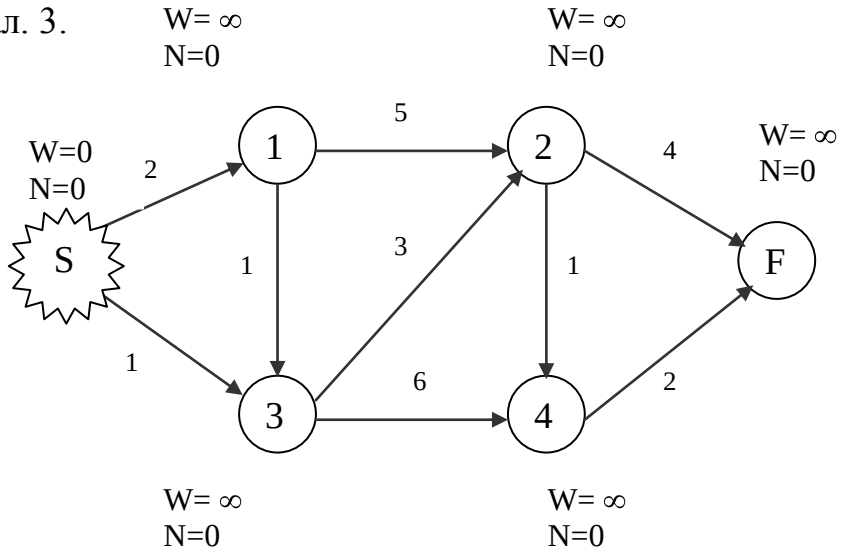
Вага вершини – це шлях , пройдений від початкової вершини до даної вершини .

НВП – номер вершини, з якої відбувся перехід у дану вершину .

Домовимося вагу вершини позначати W(v), НВП – N(v), де v – номер вершини .

Тепер перейдемо безпосередньо до опису кроків алгоритму Дійсктри .

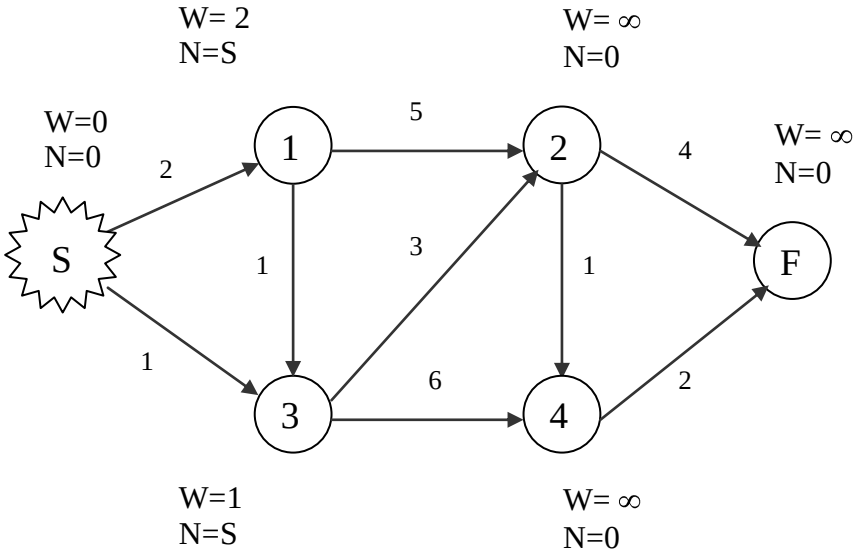
Мал. 3.



1.Прийняти НВП кожної вершини рівною нулю . Прийняти вагу початкової вершини S рівною нулю, а вагу усіх інших вершин рівною нескінченності.Вважати початкову вершину активною (див. мал. 3).

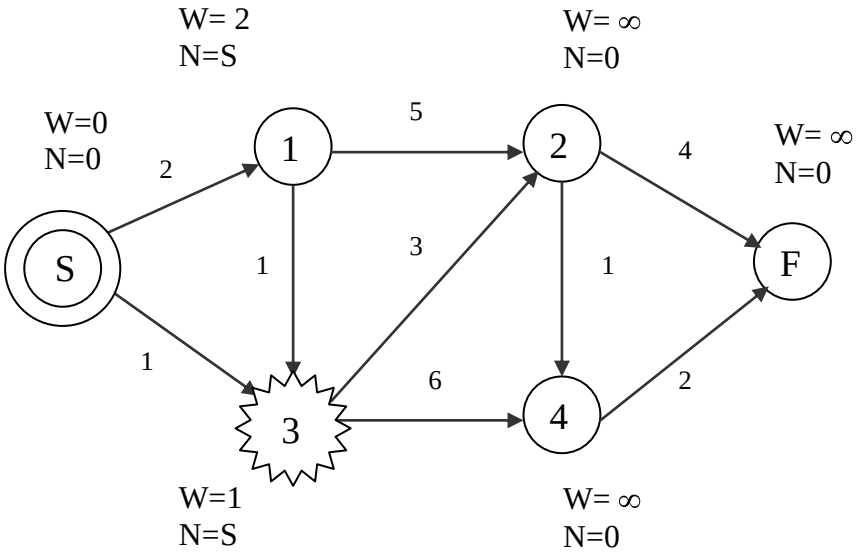
2.Для усіх невиділених вершин, до яких можна потрапити з активної вершини, виконати : якщо W(v)>W(a)+m(a,v), то W(v):=W(a)+m(a,v) і N(v):=a. Інакше W(v) і N(v) не змінювати. (тут a- номер активної вершини ; m(a,v) – вага ребра , що з'єднує вершини a та v). (див. мал. 4).

Мал. 4.



3.Серед множини невиділених вершин знайти вершину з найменшою вагою . Активну вершину зробити виділеною . Вершину з найменшою вагою зробити активною . (див. мал. 5) .

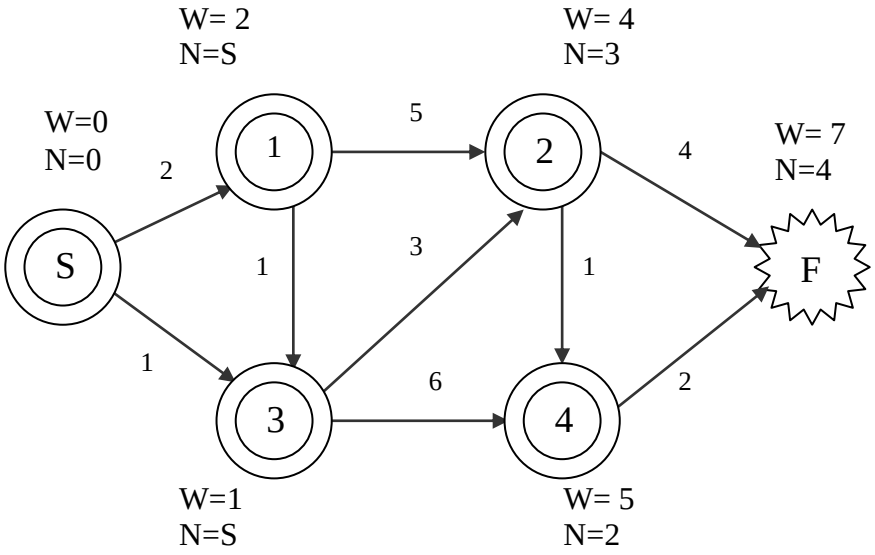
Мал. 5.



4.Якщо активною вершиною є кінцева вершина f , то перейти до пункту 5, інакше перейти до пункту 2.

5. Кінець .

Мал. 6.



Після закінчення виконання кроків алгоритму значення довжини найкоротшого шляху дорівнює вазі кінцевої вершини F .

Користуючись значеннями НВП для кожної вершини легко встановити найкоротший шлях від початкової вершини S до кінцевої вершини f .

На малюнках 3, 4, 5 проілюстровано виконання кроків алгоритму Дійсктри.

На мал. 6. зображено кінцевий результат дії алгоритму . З малюнка видно, що порядок обходу вершин по найкоротшому шляху такий : S, 3, 2, 4, F ; довжина шляху дорівнює 7 .

Перш ніж перейти до написання програми, необхідно домовитись щодо способу кодування інформації про даний граф .

Один із способів кодування полягає у створенні так званої матриці ваги для даного графу . Матриця ваги являє собою квадратну матрицю у якої кількість рядків (стовпчиків) дорівнює кількості вершин графа . Елементи матриці визначають вагу відповідних ребер . Наприклад (див. мал. 1) m(1,3)=5 означає , що вага орієнтованого ребра , яке зєднує вершини 1 та 3 дорівнює 5 .
Запис m(3,1)=∞ означає , що перехід від вершини 3 до вершини 1 неможливий .
Позначення m(i,i)=0 , де i=1,2,3… означає неможливість переходу із даної вершини в саму себе .

Враховуючи все вище сказане, матриця ваги для графа, зображеного на мал. 3 набуде такого вигляду :

	S	1	2	3	4	F
S	0	2	∞	1	∞	∞
1	∞	0	5	∞	∞	∞
2	∞	∞	0	∞	1	4
3	∞	∞	3	0	6	∞
4	∞	∞	∞	∞	0	2
F	∞	∞	∞	∞	∞	0

Тепер можна перейти до розгляду процедури, що знаходить у зваженому орієнтованому графі, заданому матрицею ваги, найкоротшого шляху від початкової вершини S до кінцевої вершини F за алгоритмом Дійкстра.

```
  . . .
const
  hmv=10;
Type
  Matrix=array[0..hmv,0..hmv] of integer;
  Vector=array[0..hmv] of integer;
  . . .
procedure minpath(m:integer;MatrixWigh:Matrix;
                        var dmas:Vector;
                        var length,hmWeb:integer);
var
   W,N : Vector;
   SelectWeb : array[0..hmv+1] of boolean;
   ActiveWeb : integer;
   i,j,min : integer;
begin

  for i:=1 to m+1 do
    begin
      W[i]:=10000;N[i]:=0;
```

```
    SelectWeb[i]:=false;
   end;
 SelectWeb[0]:=true;ActiveWeb:=0;N[0]:=0;
 W[0]:=0;

 repeat
   for i:=1 to m+1 do
    if (MatrixWigh[ActiveWeb,i]<>10000) and (ActiveWeb<>i) and
      (W[i]>W[ActiveWeb]+MatrixWigh[ActiveWeb,i])
      then
        begin
          W[i]:=W[ActiveWeb]+MatrixWigh[ActiveWeb,i];
          N[i]:=ActiveWeb;
        end;
    min:=10000;
   for i:=1 to m+1 do
    if (not(SelectWeb[i])) and (min>W[i]) then
      begin
        min:=W[i];ActiveWeb:=i;
      end;
   SelectWeb[ActiveWeb]:=true;
 until ActiveWeb=m+1;

  j:=m+1;i:=1;
 repeat
   dmas[i]:=N[j];j:=N[j];inc(i);
 until j=0;
    hmWeb:=i;
 for j:=1 to round((i-1)/2) do
   begin
     min:=dmas[j];
     dmas[j]:=dmas[i-j];
     dmas[i-j]:=min;
   end;
    dmas[i]:=m+1;
   length:=W[m+1];
end;
```

У даній процедурі у змінну m записується кількість вершин графа, крім початкової та кінцевої вершин. Масив MatrixWigh являє собою матрицю ваги графа. Масив dmas містить номери вершин графа у порядку їх обходу по мінімальному шляху; кількість елементів (вершин) цього масиву записується у змінну hmWeb. У змінну length записується значення довжини найкоротшого шляху.

На думку автора описані алгоритм та процедура не є складними для розуміння учнями старших класів загальноосвітньої школи.Даний алгоритм не потребує ніяких спеціальних знань, разом з тим вміння користуватись цим алгоритмом поповнює багаж знань учня, розвиває його логічне мислення, стимулює до процесу пізнання оточуючого світу.