*Национальный технический университет Украины «КПИ»*

*Факультет информатики и вычислительной техники*

*Кафедра вычислительной техники*

***Лабораторная работа №7***

***По курсу Дискретной математики***

***Вариант №4***

*Выполнила студентка*

*1 курса ФИВТ гр. ИВ-92*

*ГлуШко Ольга*

*Проверил Флеров А.И.*

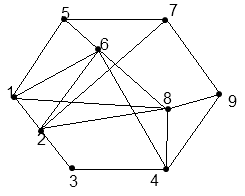
Киев 2010

**Вариант №4**

**Задание**: построить остов графа G2 методом поиска в ширину (граф в лабораторной работе №6). Разработать алгоритм и программу выполнения задания в соответствии с вариантом. Результаты выполнения задания представить графически или таблично.

**Выполнение работы**

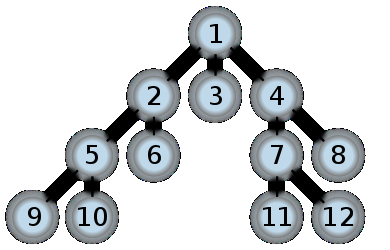
Изобразим заданный граф



Остовом графа G называют граф, не содержащий циклов и состоящий из ребер исходного графа и всех его вершин. Таким образом, остов графа является деревом. Число ребер остова равно рангу графа (v\*=n-k). Число остовов графа можно вычислить по матрице Кирхгофа.

Дерево — это связный граф (то есть такой граф, между любой парой вершин которого существует по крайней мере один путь), не содержащий циклов (то есть ациклический граф). Ацикличность означает, что в дереве существует только по одному пути между парами вершин.

Рассмотрим метод поиска в графе, называемый поиском в ширину (англ: breadth first search). Прежде чем описать его, отметим, что при поиске в глубину чем позднее будет посещена вершина, тем раньше она будет использована — точнее, так будет при допущении, что вторая вершина посещается перед использованием первой. Это прямое следствие того факта, что просмотренные, но еще не использованные вершины скапливаются в стеке.

 Порядок обхода дерева в ширину

**Алгоритм**

Поиск в ширину выполняется в следующем порядке: началу обхода s приписывается метка 0, смежным с ней вершинам — метка 1. Затем поочередно рассматривается окружение всех вершин с метками 1, и каждой из входящих в эти окружения вершин приписываем метку 2 и т. д.

Поиск в ширину, грубо говоря, основывается на замене стека очередью. После такой модификации, чем раньше посещается вершина (помещается в очередь), тем раньше она используется (удаляется из очереди). Использование вершины происходит с помощью просмотра сразу всех еще не просмотренных соседей этой вершины. Вся процедура поиска представлена ниже.

PROGRAM O\_s\_t\_o\_v\_G\_r\_a\_p\_h;

type Lref = ^Leader; { Тип: указатель на заголовочный узел }

Tref = ^Trailer; { Тип: указатель на дуговой узел }

{ Описание типа заголовочного узла }

Leader=Record

Key : Integer; { Имя заголовочного узла }

Count: Integer; { Количество предшественников }

Flag : Boolean; { Флаг посещения узла при обходе }

Trail: Tref; { Указатель на список смежности }

Next : Lref { Указатель на следующий узел в }

{ списке заголовочных узлов }

end;

{ Описание типа дугового узла }

Trailer = Record

Id : Lref;

Next: Tref

end;

var Head: Lref; { Указатель на голову списка }

{ заголовочных узлов }

Tail: Lref; { Указатель на фиктивный элемент }

{ в конце списка заголовочных узлов }

t : Lref; { Рабочий указатель для перемещения }

{ по списку заголовочных звеньев }

{ ---------------------------------------------------------- }

FUNCTION S\_e\_a\_r\_c\_h\_G\_r\_a\_p\_h (w: Integer; Head: Lref): Lref;

{ Функция возвращает указатель на заголовочный узел с ключом }

{ w в графе, заданном структурой Вирта с указателем Head }

var h: Lref;

BEGIN

h:=Head; Tail^.Key:=w;

While h^.Key<>w do h:=h^.Next;

If h=Tail

{ В списке заголовочных узлов нет узла с ключом w }

then begin

New (Tail); h^.Count:=0;

h^.Trail:=Nil; h^.Next:=Tail

end;

S\_e\_a\_r\_c\_h\_G\_r\_a\_p\_h:=h

END;

{ -------------------------------------------- }

PROCEDURE M\_a\_k\_e\_G\_r\_a\_p\_h (var Head: Lref);

{ Процедура возвращает указатель Head на структуру }

{ Вирта, соответствующую ориентированному графу }

var x,y: Integer;

p,q: Lref; { Рабочие указатели }

t,r: Tref; { Рабочие указатели }

Res: Boolean; { Флаг наличия дуги }

BEGIN

Write ('Вводите начальную вершину дуги: '); Read (x);

While x<>0 do

begin

Write ('Вводите конечную вершину дуги: ');

ReadLn (y);

{ Определим, существует ли в графе дуга (x,y)? }

p:=S\_e\_a\_r\_c\_h\_G\_r\_a\_p\_h (x,Head);

q:=S\_e\_a\_r\_c\_h\_G\_r\_a\_p\_h (y,Head);

r:=p^.Trail; Res:=FALSE;

While (r<>Nil) AND (NOT Res) do

If r^.Id=q then Res:=TRUE else r:=r^.Next;

If (NOT Res)

{Если дуга не существует, то поместим ее в граф}

then begin

New (t); t^.Id:=q; t^.Next:=p^.Trail;

p^.Trail:=t; q^.Count:=q^.Count+1

end;

Write ('Вводите начальную вершину дуги: '); Read (x)

end;

END;

{ ------------------------------- }

procedure Pw(v:integer);

var Og:array[1..Maxn]of 0..Maxn;

yk1,yk2:integer;

j:integer;

begin

fillchar(Og,sizeof(Og),0);yk2:=0;

yk1:=1;Og[yk1]:=v;Nnew[v]:=false;

while yk2 < yk1 do

begin

inc(yk2);v:=Og[yk2];

write(v:2);

for j:=1 to n do

if (a[v,j] <> 0) and Nnew[j] then

begin

inc(yk1);Og[yk1]:=j;Nnew[j]:=false;

end;

end;

end;BEGIN

{ Инициализация списка заголовочных узлов }

New (Head); Tail:=Head;

{ Построение графа }

M\_a\_k\_e\_G\_r\_a\_p\_h (Head);

{ Построение стягивающего дерева (остова) }

Write ('Остов: '); t:=Head;

While t<>Tail do

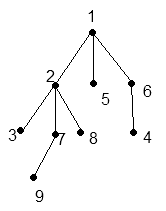
begin t^.Flag:=TRUE; t:=t^.Next end;

Pw (Head);

WriteLn

END.

Предоставим графическую интерпретацию результата, полученного в ходу программной реализации.



Вывод: в ходе лабораторной работы были изучены такие понятия, как остов графа, древо, методы получения оных. Программно был реализован поиск в ширину (BFS, Breadth-first search) — метод обхода и разметки вершин графа. Результат представлен графически.