Министерство образования Республики Белару сь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

Учебная дисциплина «Структуры и алгоритмы обработки данных»

**Инструкция**

по выполнению лабораторной работы

«Реализация основных операций над графами»

Минск

2018

**Лабораторная работа № 9**

**Тема работы: «Реализация основных операций над графами»**

**1. Цель работы**

Формирование умений применения основных операций над графами.

**2. Задание**

1. Разработать процедуры ввода графа в виде матрицы инцидентности, матрицы смежности и списка смежности.

2. Для заданного графа определить:

Вариант 1: смежность вершин v и w; веса ребра, заданного (v, w); перечислить ребра, инцидентные вершине v.

Вариант 2: перечислить все вершины смежные с вершиной v; вес вершины v; перечислить все ребра (v, w).

**3. Оснащение работы**

ПК, Pascal.

**4. Основные теоретические сведения**

Есть два основных способа представить граф: в виде списков смежных вершин и матрицы смежности. Первый предпочтительнее для разреженных графов, в которых количество ребер E намного больше V^2.

Представление графа в виде **списков смежных вершин** использует массив из **V** списков – по одному списку на вершину. Для каждой вершины список содержит в произвольном порядке все смежные с ней вершины.

*Вот как эта структура данных будет выглядеть в Паскале*

**const**

**VMax  = 100;**

**type**

**Spisok = ^List;**

**List = record**

**weight:integer;**

**next:spisok;**

**end;**

**var**

**a:array[1..VMax] of spisok;**

При использовании матрицы смежности мы нумеруем вершины графа числами **1,2,...,V** и рассматриваем матрицу **A**, в которой **a[i,j]=1** если есть ребро между вершинами i и j и **a[i,j]=0** в противном случае. В случае взвешенного графа, вместо единицы в **a[i,j]** удобно хранить вес ребра.

*В Паскале матрица будет выглядеть так:*

**const**

**VMax=100;**

**var**

**b:array[1..VMax,1..VMax] of integer;**

*Заполнение матрицы смежности из файла на Паскале*

**assign(fin,paramstr(1)); reset(fin);**

**readln(fin,n);**

**while not EOF(fin) do**

**begin**

**read(fin,i); readln(fin,j);**

**g[i,j]:=1;**

**end;**

**close(fin);**

сначала из первой строки файла считываем количество вершин графа. Затем до конца файла в строках по два числа **i** и **j** что означает что есть ребро из i и j. Эта информация заносится в матрицу.

Примеры некоторых операций с графами с помощью матрицы смежности имеют следующий вид:

*Предполагается, что вершины графа пронумерованы от 1 до* ***N****, а ребра – от 1 до* ***M****. Каждому ребру и каждой вершине может быть сопоставлен вес – целое положительное число.*

**const**

**N = 6; //Число вершин**

**a: array[1..6, 1..6] of integer = //Матрица смежности**

**((0, 1, 0, 0, 1, 0),**

**(1, 0, 1, 0, 1, 0),**

**(0, 1, 0, 1, 0, 0),**

**(0, 0, 1, 0, 1, 1),**

**(1, 1, 0, 1, 0, 0),**

**(0, 0, 0, 1, 0, 0));**

**function proverka\_smezhnosti(v, w: integer): boolean := a[v, w] > 0;** *проверка*

*смежности вершин v и w*

**procedure perechislenie\_smezhnih\_vershin(v: integer);** *перечисление всех*

*вершин смежных с v*

**begin**

**write('Вершины, смежные с ', v, '-той вершиной: ');**

**for var w := 1 to N do**

**if proverka\_smezhnosti(v, w) then print(w);**

**writeln;**

**end;**

**function ves\_rebra(v, w: integer): integer := a[v, w];** *определение веса*

*ребра (v, w);*

**function stepen\_vershini(v: integer): integer;** *определение веса*

*вершины v;*

**begin**

**Result := 0;**

**for var w := 1 to N do**

**if proverka\_smezhnosti(v, w) then Result += 1;**

**end;**

**procedure perechislenie\_vseh\_reber;** *перечисление всех*

*ребер (v, w)*

**begin**

**write('Ребра графа: ');**

**for var v := 1 to N do**

**for var w := v + 1 to N do**

**if proverka\_smezhnosti(v, w) then print('(', v, '; ', w, ')');**

**end;**

**procedure perechislenie\_incidentnih\_reber(v: integer);** *перечисление ребер,*

*инцидентных вершине v*

*begin*

**write('Ребра, инцидентные с ', v, '-той вершиной: ');**

**for var w := v + 1 to N do**

**if proverka\_smezhnosti(v, w) then print('(', v, '; ', w, ')');**

**end;**

**procedure perechislenie\_incidentnih\_vershin(v, w: integer) := writeln('Вершины, инцидентные ребру (', v, '; ', w, '): ', v, ' ', w);** *перечисление вершин,*

*инцидентных ребру s*

Пример процедуры ввода графа в виде матрицы инцидентности, матрицы смежности и списка смежности с возможностью корректировки введенных данных, а также процедуры преобразования различных форм хранения графа:

-- из матрицы смежности в список смежности и обратно;

-- из матрицы инцидентности в список смежности и обратно;

**program mgraf;**

**uses crt;**

**type pgraf=^graf;**

**graf=record**

**num:integer;**

**next:pgraf;**

**end;**

**var**

**start,current,prev:pgraf;**

**k,f,l:integer;**

**mat,mat1:array[0..100,0..100] of integer;**

**procedure vmi;**

**var a,b,c,i,j,d:integer;**

**begin**

**a:=0;b:=0; d:=0;**

**writeln('vvedite razmernosti grafa');**

**write('colvo reber ');readln(a);**

**write('colvo versin ');readln(b);**

**write('orgraf (1-yes,0-no) ');readln(f);**

**if f=1 then f:=-1**

**else f:=1;**

**for i:=0 to a-1 do**

**begin**

**write('rebro ',i+1,': ');**

**for j:=0 to b-1 do**

**read(mat[i,j]);**

**end;**

**start:=nil;**

**for i:=0 to b-1 do**

**begin**

**new(current);**

**if d=0 then**

**begin**

**start:=current; d:=1;prev:=current; end**

**else**

**begin**

**prev^.next:=current;**

**prev:=current;**

**end;**

**current^.num:=i+1;**

**current^.next:=nil;**

**for j:=0 to a-1 do**

**if mat[j,i]=f then**

**for c:=0 to b-1 do**

**begin**

**if (mat[j,c]=1) and (i<>c) then**

**begin**

**new(current); prev^.next:=current; current^.num:=c+1; current^.next:=nil;**

**prev:=current;**

**end;**

**end;**

**new(current); prev^.next:=current; current^.num:=-1; current^.next:=nil;**

**prev:=current;**

**end;**

**end;**

**procedure vms;**

**var i,j,b,d:integer;**

**begin**

**d:=0;**

**writeln('vvedite razmernosti grafa');**

**write('colvo versin ');readln(b);**

**write('orgraf (1-yes,0-no) ');readln(f);**

**if f=1 then f:=-1**

**else f:=1;**

**for i:=0 to b-1 do**

**begin**

**write('versina ',i+1,' : ');**

**for j:=0 to b-1 do**

**read(mat[i,j]);**

**end;**

**for i:=0 to b-1 do**

**begin**

**new(current);**

**if d=0 then begin start:=current; prev:=current;d:=1; end**

**else begin prev^.next:=current; prev:=current; end;**

**current^.num:=i+1;**

**for j:=0 to b-1 do**

**if mat[i,j]=1 then**

**begin**

**new(current);**

**prev^.next:=current; prev:=current; current^.num:=j+1;**

**end;**

**new(current);**

**prev^.next:=current; prev:=current; current^.num:=-1;current^.next:=nil;**

**end;**

**end;**

**procedure vss;**

**var b,i,d,s:integer;**

**begin**

**d:=0;**

**writeln('vvedite razmernosti grafa');**

**write('colvo versin ');readln(b);**

**write('orgraf (1-yes,0-no) ');readln(f);**

**if f=1 then f:=-1**

**else f:=1;**

**for i:=0 to b-1 do**

**begin**

**new(current);**

**if d=0 then begin start:=current; prev:=current;d:=1; end**

**else begin prev^.next:=current; prev:=current; end;**

**current^.num:=i+1;**

**repeat**

**write('vvesti versinu smejnuiu versine ',i+1,'? (1-da) '); read(s);**

**if s=1 then**

**begin**

**new(current);**

**prev^.next:=current; prev:=current;read(current^.num);**

**end;**

**until s<>1;**

**new(current);**

**prev^.next:=current; prev:=current; current^.num:=-1;current^.next:=nil;**

**end;**

**end;**

**procedure oss;**

**var e:integer;**

**begin**

**e:=0;**

**current:=start;**

**while current^.next<>nil do**

**if current^.num=-1 then**

**begin**

**writeln(''); current:=current^.next; e:=0;**

**end**

**else**

**begin**

**if e=0 then**

**begin**

**write(' ',current^.num,'-');current:=current^.next;e:=1;**

**end**

**else**

**begin**

**write(' ',current^.num);current:=current^.next;**

**end;**

**end;**

**end;**

**procedure oms;**

**var i,j,a:integer;**

**begin**

**a:=0;**

**current:=start;**

**while current^.next<>nil do**

**begin**

**if current^.num=-1 then a:=a+1;**

**current:=current^.next;**

**end;**

**a:=a+1;**

**for i:=0 to a-1 do**

**for j:=0 to a-1 do**

**mat[i,j]:=0;**

**current:=start;**

**i:=0;j:=0;**

**i:=current^.num-1; current:=current^.next;**

**repeat**

**if current^.num=-1 then**

**begin**

**if current^.next<>nil then**

**begin**

**current:=current^.next; i:=current^.num-1; current:=current^.next;**

**end;**

**end**

**else**

**begin mat[i,current^.num-1]:=1; current:=current^.next; end;**

**until current^.next=nil;**

**for i:=0 to a-1 do**

**begin**

**write ('versina ',i+1,': ' );**

**for j:=0 to a-1 do**

**write(mat[i,j],' ');**

**writeln('');**

**end;**

**end;**

**procedure omi;**

**var i,j,a,d,m,c:integer;**

**begin**

**m:=0;**

**a:=0;**

**d:=0;**

**current:=start;**

**while current^.next<>nil do**

**begin**

**if current^.num=-1 then a:=a+1;**

**current:=current^.next;**

**end;**

**a:=a+1;**

**for i:=0 to a-1 do**

**for j:=0 to a-1 do**

**mat[i,j]:=0;**

**current:=start;**

**i:=0;j:=0;**

**i:=current^.num-1; current:=current^.next;**

**repeat**

**if current^.num=-1 then**

**begin**

**if current^.next<>nil then**

**begin**

**current:=current^.next; i:=current^.num-1; current:=current^.next;**

**end;**

**end**

**else**

**begin mat[i,current^.num-1]:=1; current:=current^.next; end;**

**until current^.next=nil;**

**for i:=0 to a-1 do**

**for j:=0 to a-1 do**

**if mat[i,j]=1 then**

**begin**

**for c:=0 to a-1 do**

**mat1[d,c]:=0;**

**mat1[d,i]:=f; mat1[d,j]:=1;**

**mat[j,i]:=0;d:=d+1;**

**end;**

**for i:=0 to d-1 do**

**begin**

**write ('rebro ',i+1,': ' );**

**for j:=0 to a-1 do**

**write(mat1[i,j],' ');**

**writeln('');**

**end;**

**end;**

**begin**

**repeat**

**clrscr;**

**writeln ('format vvoda');**

**write ('1-matrita int, 2-matrita smejn, 3-spisok smejn: ');**

**readln(k);**

**case k of**

**1:vmi;**

**3:vss;**

**2:vms;**

**end;**

**writeln('format vivida');**

**write ('1-matrita int, 2-matrita smejn, 3-spisok smejn: ');**

**readln(k);**

**case k of**

**1:omi;**

**2:oms;**

**3:oss;**

**end;**

**writeln('');**

**write('povtoriti? (1-yes,0-no) ');**

**read(l);**

**until l=0;**

**end.**

**Для выполнения каждой из поставленных перед программой задач используется одна из 6 процедур.**

**5. Порядок выполнения работы**

**6. Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7. Контрольные вопросы и задания**

1. Что представляет собой граф?

2. Какие вершины в графе называются смежными?

3. В каком случае вершины инцидентны некоторому ребру?

4. Перечислите способы представления графов в памяти ЭВМ?

**8. Рекомендуемая литература**

1.Ахо, А.В.Структуры данных и алгоритмы / А.В. Ахо, Дж.Хопкрофт, Дж. Д.Ульман. – пер. с англ. – М.: Издательский дом "Вильямс", 2007.-400 с.

2. Вирт,Н. Алгоритмы и структуры данных / Н. Вирт. –СПб.:Невский диалект, 2008. – 352с.

3. Вирт, Н. Алгоритмы и структуры данных с примерами на Паскале / Н. Вирт[перевод с английского Д. Б. Подшивалова]; – 2-е изд., испр., – СПб.: Невский Диалект, 2005. – 352с.

4. Гагарина, Л.Г. Алгоритмы и структуры данных / Л.Г. Гагарина, В.Д. Колдаев. – учеб.пособие – М: Финансы и статистика, 2009. – 304с.

5. Котов, В.М. Алгоритмы и структуры данных: учеб.пособие / В.М. Котов, Е.П. Соболевская, А.А. Толстиков – Минск: БГУ, 2011. – 267с.

6. Макконнелл, Дж. Основы современных алгоритмов / Дж. Макконнелл – 2-е дополненное издание – М.:Техносфера, 2006. – 368с.

7. Окулов, С.М. Программирование в алгоритмах / С.М. Окулов. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2007. – 383с.