Министерство науки и высшего образования Российской Федерации  
НОВОССИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕСИТЕТ   
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

Курсовая работа

Структуры данных и алгоритмы

Факультет: прикладной математики и информатики

Группа: ПМИ-82

Студент: Хайдаев К.Е.

Преподаватель: Еланцева И.Л.

### Новосибирск 2019

**1. Условие задачи**

Найти все вершины заданного графа, недостижимые от заданной вершины

**2. Анализ задачи**

**2.1. Исходные данные задачи:**

*G (V, E)* – граф (не дерево по условию)

*V* – конечное непустое множество вершин

*E –* множество ребер

**2.2. Результат:**

**Если в графе есть вершины не смежные с данной:**

*Множество не смежных вершин*

**Если в графе нет вершин не смежных с данной :**

Таких вершин нет

**Если заданной вершины не существует:**

*Не существует заданной вершины*

**Если заданный граф тривиальный:**

*Тривиальный граф*

**Если заданный граф пустой:**

*Пустой Граф*

**2.3. Решение:**

**Математическая модель:**

**Графом (неориентированным графом)** G(V,E) называется совокупность двух множеств, где V – конечное непустое множество элементов, называемых вершинами, а E – множество неупорядоченных пар различных элементов множества V (эти пары называются ребрами). Граф, состоящий из одной вершины, называется тривиальным.

**Связный граф** – граф, у которого две любые не совпадающие вершины соединены маршрутом. Для связности графа необходимо и достаточно, чтобы в нем для какой-либо фиксированной вершины *u* и каждой другой вершины *v* существовал *(u,v)* – маршрут.

Вершина *V* и ребро *Е* называются **инцидентными***,* если *V* является одним из концов ребра *Е,* и **не инцидентными**в противном случае.

**Степень вершины** – это число ребер, инцидентных ей.

Две вершины vi и vj называются **смежными**, если они являются концевыми для некоторого ребра.

**Обход графа в глубину**. Стратегия поиска в глубину состоит в том, чтобы, начиная с начальной вершины, идти «вглубь», пока это возможно (есть не пройденные ребра), и возвращаться и искать дугой путь, когда таких ребер нет. Так делается, пока не обнаружены все вершины, достижимые из исходной. Если после этого остаются необнаруженные вершины, выбирается одна из них (как начальная) и процесс повторяется. Так делается до тех пор, пока мы не обнаружим все вершины графа.

**Метод решения:**

Для того чтобы определить является ли вершина смежной с данной из графа *G (V, E)*  нужно выполнить обход в глубину и проверить есть ли “белые” марки (=0), то есть не пройденные вершины графа от заданного.

Если условие выполнено значит заданная вершина не достижима с проверенной вершиной, нужно перейти к следующей, пока не пройдем все марки.

**Формальная постановка задачи:**

В данном графе проверить выполнение условие отсутствия связности для выбранной вершины *V* с другими вершинами заданного графа*.* От выполнения этого условия зависит результат задачи.

Таким образом решение задачи заключается в поочередном рассмотрение графа, и проверки выполнения условия смежности для каждой вершины.

Основные подзадачи:

1. Ввод графа
2. Проверка достижима ли вершина от заданной.
3. Вывод результата (вершины, которые недостижимы от заданной вершины)

**3. Структура основных входных и выходных данных**

**3.1 Входные данные**

**Внешнее представление**

Граф представляется списком смежности вершин. Он организуется следующим образом:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n - количество вершин графа | | | | | | |
| номер вершины графа | степень вершины графа | номера вершин смежных с данной | | | | |
| v1 | sv v1 | v11 | v12 | v13 | … | v1n |
| v2 | sv v2 | v21 | v22 | v23 | … | v2n |
| … | … | … | … | … | … | … |
| vn | sv vn | vn1 | vn2 | vn3 | … | vnn |

Был выбран именно такой способ представления графа, так как он очень прост для ввода пользователем и алгоритмы на списках смежности широко распространены. Исходные данные хранятся в текстовом файле Graph.txt

**Внутреннее представление**

В программе граф представлен следующей структурой данных:

struct graph {int nv; int sv; int\* ss;}; – структура вершины графа, где

int nv – целое число, номер вершины

int sv – целое число, степень вершины

int\* ss – указатель на целый тип, массив смежности вершин

Граф хранится в динамическом массиве объектов структуры graph (graph\* p).

Количество вершин графа хранится в переменной типа int n.

Массив под хранение пройденных вершин bool \*used.

* 1. **Выходные данные**

Если есть вершины, которые не смежны с данной:

Последовательность целых чисел – номера вершин.

Если в графе нет вершин не смежных с данной:

Таких вершин нет

Если вершина не существует в заданном графе:

Вершина не существует

Если заданный граф тривиальный:

Тривиальный граф

Если заданный граф пустой:

Пустой Граф

1. **Укрупненный алгоритм решения задачи**

**Укрупненный алгоритм решения задачи**

**Задание графа с помощью списка смежности:**

**bool input()**

считывание fout >> n; кол-во вершин графа из файла;

если (n==0)

вывод ("Пустой граф");

если (n==1)

вывод ("Тривиальный граф") ;

p = new graph[n] выделение памяти под граф ;

used=new bool[n] выделение памяти под массив марок;

считывание fout >> p[i].nv >> p[i].sv номеров вершин графа из файла и степеней вершин графа из файла;

если (p[i].sv != 0) степень вершины не ноль (есть смежные вершины);

p[i].ss = new int[p[i].sv] выделение памяти под номера вершин смежных с данной;

пока (j = 0; j < p[i].sv; j++) не конец кол-ва вершин смежных с данной;

считывание out >> p[i].ss[j] вершин смежных с данной;

**Укрупненный алгоритм проверки связности графа:**

**Void DFS(int nv)**

Если (used[nv]==0) заданная вершина еще не пройдена

То used[nv]=1 – заданная вершина проходится

Пока (int i=0;i<p[nv].sv;i++) счетчик меньше степени вершин

Функция вызывает саму себя DFS(p[nv].ss[i])

1. **Структура программы**

**bool input()** – подпрограмма ввода графа. Возвращает 1, если ввод произошел удачно, и 0, если введенный граф некорректен или возникла ошибка при открытии файла.

**bool check\_nv(int nv) –** подпрограммапроверки существования вершины в заданном графе, возвращает 1, если вершина существует, и 0, если не существует.

int nv – заданная вершина.

**void clean\_used() –** подпрограмма, котораяделает все марки “белыми”, то есть равными 0.

**void DFS(int nv) –** подпрограмма обхода в глубину.

int nv – заданная вершина.

**int check(int i) –** подпрограмма проверки смежна ли вершина с заданной, возвращает -1, если не существует, и если существует, то не смежную вершину (проверяет равен ли элемент массива марок 0).

int i – счетчик

**bool clear() –** отчищает динамическую память.

**int main** – головная функция, связывающая между собой все подпрограммы.

**Структура программы по управлению:**



1. **Текст программы на языке Си**

#include <iostream>

#include <fstream>

using namespace std;

//структура граф

// nv- номер вершины

//sv- степень вершины

// \*ss - список смежности вершин

// \*p- указатель для блока памяти под граф

struct graph

{

int nv,sv,\*ss;

} \*p=nullptr;

int n; // колличество вершин

bool\* used=nullptr; // массив, который хранит значения пройденных вершин

bool input()

{

ifstream fout;

fout.open("Graph.txt");

if (!fout.is\_open())

{

cout << "невозможно открыть файл или он не существует!\n";

return 0;

}

fout >> n;

if (n == 0)

{

cout << "граф пуст !\n";

fout.close();

return 0;

}

else if (n == 1) // граф состоит из 1 вершины

{

p = new graph;

fout >> p->nv>>p->sv;

cout << "граф тривиален, следовательно не достижимых вершин до него не существует = 0\n ";

fout.close();

return 0;

}

p = new graph[n];

used = new bool[n];

for (int i = 0; i < n; i++)

{

fout >> p[i].nv >> p[i].sv;

if (p[i].sv != 0) // есди sv!= 0 значит есть смежные вершины

{

p[i].ss = new int[p[i].sv];

for (int j = 0; j < p[i].sv; j++)

fout >> p[i].ss[j];

}

}

fout.close();

return 1;

}

bool check\_nv(int nv) // проверка на существование вершины в заданном графе

{

int temp = 0;

for (int i = 0; i < n; i++)

if (nv == p[i].nv)

temp++;

if (temp)

return 1;

else

return 0;

}

void clean\_used()

{

for (int i = 0; i < n; i++)

used[i] = 0;

}

void DFS(int nv)

{

if (used[nv] == 0) //если марка = 0

{

used[nv] = 1; // метка равна 1

for (int i = 0; i < p[nv].sv; i++)// пока меньше степени вершины

DFS(p[nv].ss[i]); // функция вызывает сама

}

}

int check(int i)//поиск не пройденных вершин

{

if (!used[i])// если метка = 0 значит вершина не пройдена и нарушено условие связности

return p[i].nv;

else

return -1;

}

bool clear() // чистим память

{

if (n == 0)

return 0;

else if (n == 1)

{

delete p;

p = nullptr;

}

else

{

for (int i = 0; i < n; i++)

delete[] p[i].ss;

delete[] p;

delete[] used;

p = nullptr;

used = nullptr;

}

return 1;

}

int main()

{

setlocale(LC\_ALL, "ru");

int nv; // номер вершины

if (!input())

{

system("pause");

clear();

exit(0);

}

cout << "введите номер вершины:" << endl;

cin >> nv;

cout << endl;

clean\_used();

if (check\_nv(nv))

{

cout << "вершины не смежные с данной :\n";

DFS(nv);

for (int i = 0; i < n; i++)

if (check(i) != -1)

cout << check(i) << "\t";

cout << endl;

}

else

{

cout << "заданной вершины не существует в данном графе\n";

}

clear();

}

1. **Набор тестов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № введеной вершины | Содержание файла "input.txt" | Графическое представление | Результат | Пояснения |
| - | 0 |  | Пустой граф | Пустой граф |
| - | 1  0 | C:\Users\PPTTFFGG\Downloads\KYyMuGvPIvfSWrpV.png | Тривиальный граф | Тривиальный граф |
| 0,1,2 | 3  0 2 1 2  1 2 0 2  2 2 0 1 | C:\Users\PPTTFFGG\Downloads\qQnGcZEWdprGKEHc.png | Все вершины достижимы от данной | Все вершины достижимы от данной |
| 0 | 8  0 2 1 2  1 2 0 3  2 2 0 4  3 2 1 5  4 2 2 5  5 2 3 4  6 2 7 8  7 1 6  8 1 6 | C:\Users\PPTTFFGG\Downloads\gXfpTRRdFRfaLkCJ.png | Список вершин не достижимых данной :  6,7,8 | Граф содержит висячие вершины, нарушено условие связности |
| 3 | 4  0 1 1  1 1 0  2 1 3  3 1 2 | C:\Users\PPTTFFGG\Downloads\FFAgFXxpJssXiSjo.png | Список вершин не достижимых данной :  0,1 | Граф содержит висячие вершины |
| 20 | 7  0 2 1 2  1 3 0 3 4  2 3 0 5 6  3 2 1 4  4 2 1 3  5 1 2  6 1 2 | C:\Users\PPTTFFGG\Desktop\NHOzRdWsHIfXSwTi.png | заданная вершина не принадлежит  данному графу | Вершина не существует в заданном графе |
| 0 | 4  0 3 1 2 3  1 3 0 2 3  2 3 0 1 3  3 3 0 1 2 |  | Таких вершин нет | Все вершины достижимы от данной |

**8.** **Результаты работы программы**

На всех тестах программа вывела верные результаты, следовательно она работает верно.

1. **Литература**
2. Хиценко В.П., Шапошникова Т.А. Структуры данных и алгоритмы: методические указания. – Новосибирск: НГТУ, 2008.
3. Хиценко В.П., Шапошникова Т.А. Практикум на ЭВМ. Алгоритмы: учебное пособие. – Новосибирск: НГТУ, 2004.