СПб НИУ ИТМО

кафедра ИПМ

Алгоритмы и структуры данных

Лабораторная работа № 5

Задачи на графы

Вариант 9

Работу выполнил:

Студент II курса

Группы № 2120

Журавлев Виталий

Санкт-Петербург

2014 г.

**Цель работы:**

Написать программу, являющуюся решением заданной по условию задачи с вводом входных данных и выводом результата.

**Условие задачи:**

Винни-Пух и Пятачок нанялись защищать компьютерную сеть от хакеров, которые выкачивали из компьютеров секретную информацию. Компьютерная сеть Винни-Пуха и Пятачка состояла из связанных между собой больших ЭВМ, к каждой из которых подключалось несколько терминалов. Подключение к одной из больших ЭВМ позволяло получить информацию, содержащуюся в памяти этой ЭВМ, а также всю информацию, доступную для ЭВМ, к которым данная ЭВМ могла направлять запросы. Хаккеры и раньше нападали на подобные компьютерные сети и их тактика была известна. Поэтому Винни-Пух и Пятачок разработали специальную программу, которая помогла принять меры против готовившегося нападения.
***Тактика хакеров.***

При нападениях хакеры всегда получают доступ к информации всех ЭВМ сети. Добиваются они этого, захватывая некоторые ЭВМ сети, так чтобы от них можно было запросить информацию у оставшихся ЭВМ. Вариантов захвата существует множество.
Например, захватить все ЭВМ. Но хакеры всегда выбирают такой вариант, при котором суммарное количество терминалов у захваченных ЭВМ минимально.
*Примечание*: В сети Винни-Пуха ни у каких 2-х ЭВМ количество терминалов не совпадает.
***Техническое задание.***

Вам необходимо написать программу, входными данными которой было бы описание сети, а выходными - список номеров ЭВМ, которые могут быть выбраны хаккерами для захвата сети согласно их тактике. Ввод осуществляется из файла с именем INPUT.TXT.

***Формат ввода:***
Количество ЭВМ в сети : N
ЭВМ #1 имеет терминалов : T[1]
ЭВМ #2 имеет терминалов : T[2]
...
ЭВМ #N имеет терминалов : T[N]
Права на запрос :
A[1] B[1]
A[2] B[2]
...
A[K] B[K]
0 0
A[i] и В[i] - номера ЭВМ, последняя строка '0 0' обозначает конец списка прав на запрос, каждая пара A[i] B[i] обозначает, что ЭВМ с номеров A[i] имеет право запрашивать информацию у ЭВМ с номером B[i] (A[i] не равно B[i]).
При вводе числа N и T[i] - натуральные, T[i] <=1000, N<=50, K<=2450.
***Формат вывода:***
Номера захватываемых ЭВМ : С[1] C[2] ... С[M].
Количество захватываемых ЭВМ : <M>

**Теоритические основы:**

Задача сводится к нахождению компонент связности. В каждой из них необходимо найти вершину с минимальным числом терминалов и вывести ее номер.

Подграф графа называется его компонентой связности, если каждая из вершин подграфа достижима от любой из верши подграфа.

***Алгоритм поиска компонент связности в графе***

Дан неориентированный граф  с  вершинами и  рёбрами. Требуется найти в нём все компоненты связности, т.е. разбить вершины графа на несколько групп так, что внутри одной группы можно дойти от одной вершины до любой другой, а между разными группами — пути не существует. Для решения используется обход в глубину. Фактически, мы производим серию обходов: сначала запустим обход из первой вершины, и все вершины, которые он при этом обошёл — образуют первую компоненту связности. Затем найдём первую из оставшихся вершин, которые ещё не были посещены, и запустим обход из неё, найдя тем самым вторую компоненту связности. И так далее, пока все вершины не станут помеченными.

**Описание алгоритма программы:**

В программе граф задан матрицей смежностей. Поиском в глубину находим компоненту связности. Выполняем этот шаг для каждой непомеченной вершины и получаем минимальной множество компонент связности, т.е. минимальный список множеств ЭВМ, в которых необходимо “взломать” только одну машину для доступа к остальным из этого множества. Однако для оптимального выбора машин по количеству терминалов необходимо выбрать лучшую машину (с наименьшим количеством терминалов) в каждой компоненте связанности. Поэтому для каждого множества находится ЭВМ с наименьшим количеством терминалов и ее номер заносится в результирующий список, который затем выводится вместе со своей длиной.

**Код программы:**

#include "stdafx.h"

#include "iostream"

#include "algorithm"

#include "fstream"

#include "vector"

using namespace std;

const int MAX\_N = 50, MAX\_T = 1000;

void dfs(int v, vector<bool> &used, vector<vector<int>> &g, vector<int> &temp) {

 used[v] = true;

 temp.push\_back(v);

 for (int i = 0; i < g[v].size(); i++) {

 if (!used[i] && g[v][i])

 dfs(i, used, g, temp);

 }

}

void find\_comps(int N, vector<vector<int>> &g, vector<vector<int>> &Comp) {

 vector<int> temp;

 vector <bool> used(g.size());

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

 used[i] = false;

 for (int i = 0; i < N; i++) {

 if (!used[i]) {

 temp.clear();

 dfs(i, used, g, temp);

 Comp.push\_back(temp);

 }

 }

}

void find\_min(vector<int> &T, vector<int> &C, vector<vector<int>> Comp) {

 int min, min\_elem;

 for (int i = 0; i < Comp.size(); i++) {

 min = MAX\_T;

 min\_elem = MAX\_N;

 for each (int j in Comp[i]) {

 if (T[j] < min) {

 min = T[j];

 min\_elem = j;

 }

 }

 C.push\_back(++min\_elem);

 }

}

int \_tmain(int argc, \_TCHAR\* argv[]) {

 int N = 50, from, to;

 vector<int> comp;

 string in\_file = "input.txt", out\_file = "output.txt";

 ifstream in;

 ofstream out;

 in.open(in\_file);

 in >> N;

 vector<int> T(N), C;

 vector<vector<int>> g(N, vector<int>(N)), Comp;

 for (int i = 0; i < N; i++) {

 in >> T[i];

 }

 while (!in.eof()) {

 in >> from;

 in >> to;

 if (from && to)

 g[--from][--to]++;

 }

 in.close();

 find\_comps(N, g, Comp);

 find\_min(T, C, Comp);

 out.open(out\_file);

 for each (int i in C) {

 out << i << ' ';

 }

 out << endl << C.capacity();

 out.close();

 return 0;

}

**Входные данные:**

Input.txt:
10

419

467

334

500

869

724

478

358

962

464

1 3

2 1

3 4

3 9

4 1

5 8

6 5

7 6

8 7

8 10

9 1

9 3

10 7

0 0

**Результат работы программы:**

output.txt:

3 2 8

3

**Вывод:**

В ходе выполнения лабораторной работы была решена задача на графы.

Было изучено понятие компоненты связности и ее применение в нахождении минимального количества множеств подграфов, охватываемых все вершины этого графа.

Так же был рассмотрен поиск обходом по графу в глубину, изученный ранее

в лабораторной работе № 2.