Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Л и ОА в ИЗ»

на тему «Поиск расстояний в графе»

**Выполнили:**

студенты группы 19ВВ2

Сидоров Н. Р.

Карамышев А. А.

Горбунов Д.А.

**Приняли:**

Юрова О. В.

Митрохин М. А.

Пенза 2020

**Название:**

Поиск расстояний в графе

**Цель работы:**

Научиться применять алгоритм поиска расстояний в графе на основе алгоритма поиска в ширину.

**Лабораторные задания:**

### ***Задание 1***

1. Сгенерировали (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G. Вывели матрицу на экран.

2. Для сгенерированного графа осуществили процедуру поиска

расстояний.

При реализации алгоритма в качестве очереди использовали класс queue из

стандартной библиотеки С++.

**Описание метода решения задач:**

Реализация состоит из подготовительной части, в которой все вершины

помечаются как не поcещенные. В отличие от алгоритма BFS не

посещенные вершины помечаем -1, т.к. значение 0 и 1 могут быть

расстояниями. Расстояние 0 – от исходной вершины до самой себя.

В самой процедуре как и в алгоритме BFS сначала создается пустая

очередь, в которую помещается исходная вершина, из которой начат

обход. Расстояние до этой вершины устанавливается равным 0

(расстояние до самой себя).

Далее итерационно, пока очередь не опустеет, из нее извлекается

первый элемент, который становится текущей вершиной. Затем в

цикле просматривается v-я строка матрицы смежности графа G(v,i). Как

только алгоритм встречает смежную с v не посещенную вершину, эта

вершина помещается в очередь и для нее обновляется вектор

расстояния. Расстояние до новой i-й вершины вычисляется как

расстояние до текущей v-й вершины плюс 1 (так как ребра нашего графа не

взвешенные).

После просмотра строки матрицы смежности алгоритм делает

следующую итерацию цикла 2.4 или заканчивает работу, если очередь пуста.

Таким образом, если вершина помещается в очередь при просмотре

сроки матрицы смежности на 1-й итерации, то они находятся на 1 уровне

удаленности и расстояние до этих вершин будет равным 1.

DIST [ i ] = DIST [ v ] + 1, где DIST [ v ] = 0 – расстояние от исходной

вершины до самой себя.

Далее, начинают просматриваться вершины первого уровня и

соответствующие им строки матрицы смежности. При добавлении смежных

с вершинами первого уровня вершин, расстояния до них будут равны 2.

DIST [ i ] = DIST [ v ] + 1, где DIST [ v ] = 1 – расстояние от исходной

вершины до вершин 1 уровня.

После того, как все вершины первого уровня будут просмотрены и

извлечены из очереди, начнется просмотр вершин 2 уровня и

соответствующих им строк матрицы смежности. При добавлении смежных с

вершинами второго уровня вершин, расстояния до них будут равны 3.

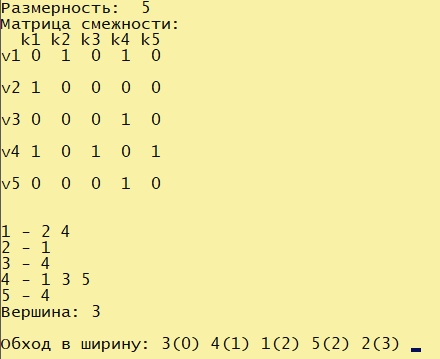
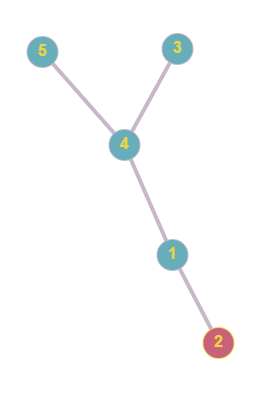
DIST [ i ] = DIST [ v ] + 1, где DIST [ v ] = 2 – расстояние от исходной

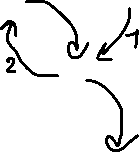
вершины до вершин 2 уровня.

И так далее, алгоритм проходит вершины по уровням, пока очередь не

опустеет.

**Результат работы программы:**

 ****



Ручной подсчет. Начинаем с вершины 3(0), от этой вершины есть только одна смежная (4), значит расстояние от 3 до 4 вершины составляет всего 1 шаг: 4(1). От 4-ой вершины есть две смежные: 5 и 1. Идем в 5, а это еще 1 шаг, значит расстояние до 5-ой вершины 1 + 1 = 2: 5(2). Далее возвращаемся в 4 и идем в вершину 1, это 1 шаг от вершины 4 (не считая, когда возвращались из 5-ой), 1(2), из этой вершины есть путь только в вершину 2, а это еще один шаг 2(3).



Посмотрим результат ручного просчета:

3(0) 4(1) 1(2) 5(2) 2(3)

Исходя из результатов работы программы и ручного подсчета можно сделать вывод, что расстояния в том и другом случае совпадают и программа работает правильно.

**Вывод:**

Научились применять алгоритм поиска расстояний в графе на основе алгоритма обхода в ширину на примере матричного представления графа, а также визуализировать пути в виде обыкновенного графа.

Листинг 1:

|  |
| --- |
| #include <time.h> |
|  | #include <iostream> |
|  | #include <stack> |
|  | #include <queue> |
|  | #include <conio.h> |
|  | //работа сделана |
|  | using namespace std; |
|  | queue <int> Q; |
|  | int i, j, m, n, v; |
|  | int\* vis; |
|  | int\*\* graph; |
|  |  |
|  | void BFS(int s) { |
|  | s--; |
|  | for (int i = 0; i < m; i++) |
|  | vis[i] = -1; |
|  |  |
|  | Q.push(s); |
|  |  |
|  | vis[s] = 0; |
|  |  |
|  | while (!Q.empty()) { |
|  |  |
|  | s = Q.front(); |
|  | Q.pop(); |
|  |  |
|  | printf("%d(%d) ", s + 1, vis[s]); |
|  |  |
|  | for (int r = 0; r < n; r++) { |
|  | if ((graph[s][r] == 1) && (vis[r] == -1)) |
|  | { |
|  | Q.push(r); |
|  | vis[r] = vis[s] + graph[s][r]; |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void DFS\_main() |
|  | { |
|  | setlocale(LC\_ALL, "Rus"); |
|  | printf("Размерность: "); |
|  | scanf\_s("%d", &m); |
|  | n = m; |
|  | graph = new int\* [m]; |
|  | vis = (int\*)malloc(m \* sizeof(int)); |
|  | for (int i = 0; i < m; i++) { |
|  | graph[i] = new int[m]; |
|  | } |
|  | hah: |
|  | //генерация матрицы |
|  | int start; |
|  | printf("Матрица смежности: \n"); |
|  | srand(time(NULL)); |
|  | for (i = 0; i < m; ++i) { |
|  | for (j = i; j < m; ++j) { |
|  | graph[i][j] = graph[j][i] = rand() % 2; |
|  | } |
|  | graph[i][i] = 0; //обнуление вершины |
|  | } |
|  | //нумерование столбцов |
|  | printf(" "); |
|  | for (j = 0; j < m; j++) |
|  | { |
|  | printf("k%d ", j + 1); |
|  | } |
|  |  |
|  | printf("\n"); |
|  |  |
|  | //нумерование строк |
|  | for (i = 0; i < m; ++i) |
|  | { |
|  | printf("v%d ", i + 1); |
|  | vis[i] = 0; |
|  | for (j = 0; j < m; ++j) |
|  | printf("%d ", graph[i][j]); |
|  | printf("\n\n"); |
|  | } |
|  | //вывод списков смежности |
|  | for (i = 0; i < m; i++) |
|  | { |
|  | printf("\n%d - ", i + 1); |
|  | for (j = 0; j < m; j++) { |
|  | if (graph[i][j] == 1) printf("%d ", j + 1); |
|  | } |
|  | } |
|  | // |
|  | printf("\nВершина: "); |
|  | scanf\_s("%d", &start); |
|  | if (start == 777) { |
|  | system("CLS"); |
|  | goto hah; |
|  | } |
|  |  |
|  | while (start > m) { |
|  | printf("\nТакой вершины нет!\n"); |
|  | printf("\nВершина: "); |
|  | scanf\_s("%d", &start); |
|  | } |
|  |  |
|  | printf("\nОбход в ширину: "); |
|  | BFS(start); |
|  | \_getch(); |
|  | } |
|  |  |
|  | int Menu(void) |
|  | { |
|  | system("CLS"); |
|  | int c = 0; |
|  | setlocale(LC\_ALL, "Rus"); |
|  | while (c < '0' || c > '1') |
|  | { |
|  | printf( |
|  | "::=================================::\n" |
|  | "|| 1 : Обход в ширину ||\n" |
|  | "|| 0 : Выход ||\n" |
|  | "::=================================::\n" |
|  | "> "); |
|  | c = \_getch(); |
|  | system("cls"); |
|  | } |
|  | return c; |
|  | } |
|  |  |
|  | void main(void) |
|  | { |
|  | int Select; |
|  | while ((Select = Menu()) != '0' && Select != 27) |
|  | switch (Select) |
|  | { |
|  | case '1': |
|  | DFS\_main(); |
|  | break; |
|  | } |
|  | } |