***1.Постановка задачи.***

Основные задачи работы:

* ознакомление со способами представления графов в памяти ЭВМ
* реализация чтения графа из файла со списком вершин и ребер
* графическое отображение графа (возможно – на основе данных о коордитнатах вершин);
* реализация алгоритмов поиска в глубину и поиска в ширину с графическим выделением исходной вершины и пошаговым графическим выделением посещенных вершин и ребер;
* анализ различия алгоритмов поиска в глубину и поиска в ширину.

***2.Теория***

**O.1. Граф -** совокупность множества вершин **V** и множества дуг **E**, которое является

подмножеством декартового квадрата множества вершин.

**О.2.** Пусть **V** - непустое множество, **V2** - множество всех его двухэлементных подмножеств.

Пара **(V,E)**, где **E** - произвольное подмножество **V2**, называется *неориентированным графом*.

Элементы множества **V** называются **вершинами**, а элементы множества **E** - **ребрами**.

Если **{v1,v2}** - ребро, то вершины **v1** и **v2** будем называть концами ребра.

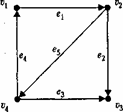
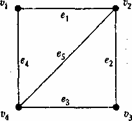
    Будем использовать символы |**V**| и |**E**| для обозначения, соответственно, числа вершин и числа

ребер в графе **(V,E)**.

**Представление графов в ЭВМ**

Известны различные способы представления графов в памяти компьютера, которые различаются объемом занимаемой памяти и скоростью выполнения oneраций над графами. Представление выбирается, исходя из потребностей конкретной задачи. Далее приведены четыре наиболее часто используемых представления с указанием характеристики **n(р, q)** — объема памяти для каждого представления. Здесь **р** — число вершин, a **q** — число ребер.

Представления иллюстрируются на конкретных примерах графа **G** и орграфа **D** диаграммы которых представлены на рис:



**Рис.** Диаграммы графа (слева) и орграфа (справа), используемых в качестве примеров

**Матрица смежности**

Представление графа с помощью квадратной булевской матрицы

**М** : **array** [l..p, l..p] of 0..1,

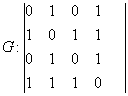
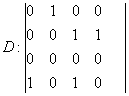
 отражающей смежность вершин, называется ***матрицей смежности,*** где

http://www.pgtk.edu.ru/legacy/lection/discret_math/graf/6_4.files/image006.gifесли вершина **vi** смежна с вершиной **vj**

если вершины **vi**  и **vj** не смежны.

Для матрицы смежности **n(p,q) = О(р2**)

**Пример:**



**Матрица инциденций**

Представление графа с помощью матрицы **Н :** **array [l..p, l..q] of 0..1** (для орграфов

**H** : **array [l..p,l..q] of -1..1),** отражающей инцидентность вершин и ребер, называется ***матрицей инциденций*,** где для неориентированного графа

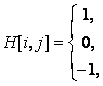
|  |
| --- |
|  |
|  |  |

http://www.pgtk.edu.ru/legacy/lection/discret_math/graf/6_4.files/image012.gif   если вершина **vi** инцидентна ребру **ej**

                     в противном случае,

а для ориенти

рованного графа

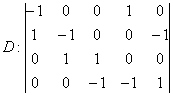
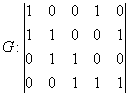
 если вершина **vi** инцидентна ребру **ej** и является его концом

 если вершина **vi** и ребро **ej** не инцидиентны,

 если вершина **vi** инцидентна ребру **ej** и является его началом.

Для матрицы инциденций **n(p,q)=O(pq).**

**Пример:**

****

**Списки смежности**

Представление графа с помощью списочной структуры, отражающей смежность вершин и состоящей из массива указателей **Г** : **array [l..p] of ­ N** на списки смежных вершин (элемент списка представлен структурой **N** : **record v : l..p; n :­ N** endrecord), называется ***списком смежности.*** В случае представления неориентированных графов списками смежности **n(p,q) = O(p + 2q),** а в случае ориентированных графов **n(р, q) = О(р + q).**

**Пример**

Списки смежности для графа **G** и орграфа **D** представлены на рис:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | http://www.pgtk.edu.ru/legacy/lection/discret_math/graf/6_4.files/image021.gif |  | http://www.pgtk.edu.ru/legacy/lection/discret_math/graf/6_4.files/image022.gif |
|  |  |  |

**Рис.** Списки смежности для графа G (слева) и орграфа D (справа)

**Массив дуг**

Представление графа с помощью массива структур **E**: **array** **[1..p]** **of record** **b,e: 1..p** **endrecord,**отражающего список пар смежных верщин, называется ***массивом ребер*** (или, для орграфов, ***массивом дуг***). Для массива **ребер** (или дуг) **n(p,q)=O(2q).**

Пример

Представление с помощью ***массива ребер*** (дуг) показано в следующей таблице (для графа **G** слева, а для **орграфа D** справа).

|  |  |  |  |
| --- | --- | --- | --- |
| b | е | b | е |
| 1 | 2 | 1 | 2 |
| 1 | 4 | 2 | 3 |
| 2 | 3 | 2 | 4 |
| 2 | 4 | 4 | 1 |
| 3 | 4 | 4 | з |

В данной лабораторной работе был использован способ представления графа с помощью матрицы смежности.

**Обходы графов**

***Обход*** графа — это

1. перебор всех вершин, в которых существует путь из заданной вершины с нахождением одного из путей

либо

1. перебор всех путей, начинающихся в заданной вершинe.

Наибольший интерес представляют ***обходы***, использующие локальную информацию (***списки смежности).*** Среди всех обходов наиболее известны ***поиск в******ширин****у* и ***в глубину***. Алгоритмы поиска в ширину и в глубину лежат в основе многих конкретных алгоритмов на графах.

**Поиск в глубину (depth-first search).** Один из методов систематического обхода вершин графа называется **поиском в глубину**. Стратегия поиска в глубину, как следует из ее названия, состоит в том, чтобы идти "вглубь" графа, насколько это возможно. При выполнении поиска в глубину исследуются все ребра, выходящие из вершины, открытой последней, и покидает вершину, только когда не остается неисследованных ребер — при этом происходит возврат в вершину, из которой была открыта вершина v. Этот процесс продолжается до тех пор, пока не будут открыты все вершины, достижимые из исходной. Когда вершина v открывается в процессе сканирования списка смежности уже открытой вершины и, процедура поиска записывает это событие, устанавливая поле предшественника v prev[v] равным u.

**Поиск в ширину (breadth-first search).** Один из методов систематического обхода вершин графа называется **поиском в ширину**. Он получил свое название из-за того, что при достижении во время обхода любой вершины v далее рассматриваются все вершины, смежные с вершиной v, т.е. осуществляется просмотр вершин "в ширину". Этот метод можно применить и к ориентированным графам.

Пусть задан граф G = (V, Е) и выделена исходная (source) вершина S. Алгоритм поиска в ширину систематически обходит все ребра G для "открытия" всех вершин, которые можно достичь. Кроме того, в процессе обхода строится "дерево поиска в ширину" с корнем s, содержащее все достижимые вершины. Для каждой достижимой из s вершины v путь в дереве поиска в ширину соответствует кратчайшему (т.е. содержащему наименьшее количество ребер) пути от s до v в G.

Формирование путей происходит с помощью массива prev, который хранит индексы, предшествующие каждой из текущих вершин, путем сравнения индексов вершины, заданной пользователем и вершины, которая рассматривается в данный момент. Все пути рекурсивно восстанавливаются до тех пор, пока не достигнут индекс вершины, с которой начинался весь обход.

***3.Руководство программиста***

Это программа разработана на языке С++ и содержит следующие функции.

***void outxy(int x, int y, int var)*  -** *функция, которая преобразует элементы массива, в текстовую форму и выводит по заданным координатам. Параметры: координаты места вывода на экран, элемент кучи.*

***void Creation(Queue \*Q) -*** *функция, перемещает элемент кучи вниз. Параметры: индекс элемента.*

***void Creation(Queue \*Q)*** *–функция создания создание очереди.*

***bool Full(Queue \*Q)*** *-функция проверки очереди на пустоту.*

***void Add(Queue \*Q,int i)*** *–функция добавления элемента в очередь.*

***void Delete(Queue \*Q)*** *–функция удаления элемента из очереди.*

***int Top(Queue \*Q)*** *–функция вывода начального элемента из очереди.*

***void void showPath(int index1, int index2, int \*p) -*** *функция, которая используются для вывода путей при поисках в ширину и в глубину.* ***Параметры****: Индексы вершин графа, указатель на массив prev.*

***void gphDFS(int i) –*** *функция, графического отображения поиска в глубину.* ***Параметры****: индекс вершины*

***void DFS(int vert, int n, int num, int \*prev1)*** - *функция, содержащая рекурсию. Осуществляет поиск в глубину.* ***Параметры****: вершина, количество ребер, указатель на массив prev.*

***void gphBFS(int j) –*** *функция, графического отображения поиска в ширину.* ***Параметры****: индекс вершины*

***void BFS(int n, int tmp, int num, Queue Q) –*** *функция, осуществляет поиск в ширину.* ***Параметры****: количество ребер, количество ребер, очередь, вершина.*

***void graph() –*** *инициирует графический режим.*

***void matrica(int \*ind1,int \*ind2,int n) –*** *функция построения матрицы смежности.* ***Параметры****: входные файлы, количество вершин.*

***int ineterfies(int a, int b, int \*m, int \*n, int \*p, int \*p1) –*** *функция графического отображения графа.* ***Параметры:*** *количество ребер, количество вершин, указатели на массивы координат, указатели на входные текстовые файлы в которых храниться информация.*

***void searchElem(int i, int n, int \*a, int \*b, int \*c) -***  *функция графического отображения найденного элемента перед поиском в ширину или в глубину.* ***Параметры:*** *цвет, индекс выбранной вершины, указатели на массивы координат, указатель на массив значений.*

***int main() –*** *основная функция программы. Запрашивает у пользователя номер вершины, с которой осуществляются поиск в ширину и поиск в глубину. Осуществляет чтение данных из текстовых файлов, выполняет поиск в ширину и обращается к функции поиска в глубину.*

***4.Тестирование***

**Оценка временной сложности.**

### Оценка сложности   поиска в глубину

Ниже приведен псевдокод алгоритма поиска в глубину.

**DFS(G)**

1 **for** (Для) каждой вершины u є V[G)

2 **do**color [u] «- WHITE

3 prev[u] «- NIL

4 time «— О

5 **for** (Для) каждой вершины u Е V[G]

6 **doif** color[u] = white

7  **then**DFS\_Visit(u)

**DFS\_Visit(u)**

1 color[u] «— GRAY // Открытабелаявершинаu

2 time«— time+1

3 d[u] «— time

4 **for** (Для) каждой вершины v є Adj[u] // Исследование ребра (u, v).

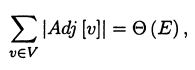
5  **do if** color [v] = white

6 **then** prev[v] «— u

1 DFS\_Visit(v)

8 color [u] «— BLACK // Завершение

9 f[u] «— time«— time+1

Циклы в строках 1-3 и 5-7 процедуры DFS выполняются за время o(|V|), исключая время, необходимое для вызова процедуры DFS\_Visit. Процедура DFS\_VISIT вызывается ровно по одному разу для каждой вершины v є V, так как она вызывается только для белых вершин, и первое, что она делает, — это окрашивает переданную в качестве параметра вершину в серый цвет. В процессе выполнения DFS\_Visit(v) цикл в строках 4-7 выполняется |Adj[v]| раз. Поскольку   
  
общая стоимость выполнения строк 4-7 процедуры DFS\_Visit равна o(|Е|).

Время работы процедуры DFS, таким образом, равно **o(|V |+ |Е|)**.

### Оценка сложности    поиска в ширину

Ниже приведен псевдокод алгоритма поиска в ширину.

**BFS(G, s)**

1 **for** (для) каждой вершины u є V[G] — s

2 **do**color[u] «— WHITE

3 d[u] «- infinity

4 previous[u] «- NIL

5 Color[s] «— GRAY

6 d[s] «- 0

7 previous[s] «- NIL

8 Q - пустаяочередь

9 Enqueue(Q, s)

10 **while** Q непустая

11 **do** u «— Dequeue(Q)

12 **for** (для) каждойvєAdj[u]

13 **do if**color[v] = white

14 **then**color[v] «— GRAY

15 d[v] «- d[u] + 1

16 prev[v] «— u

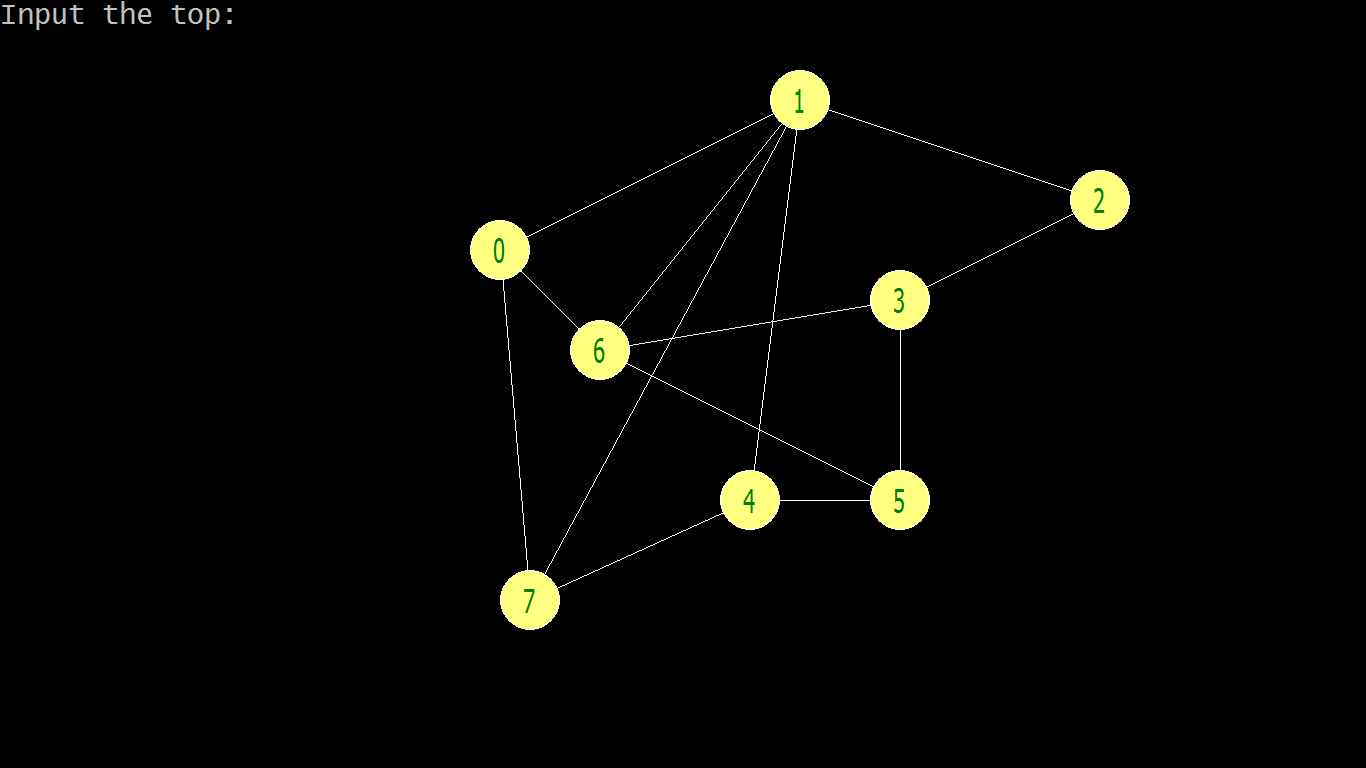
17 Enqueue(Q,v)

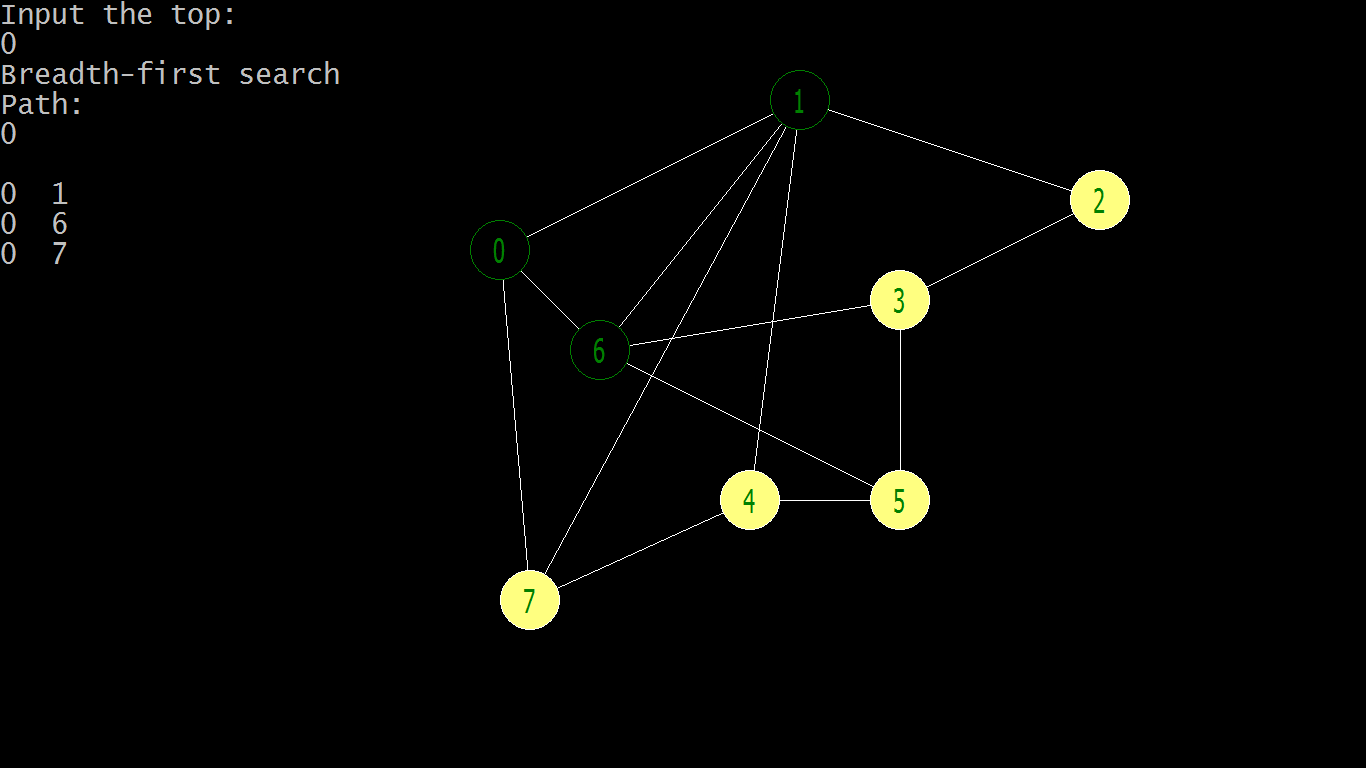
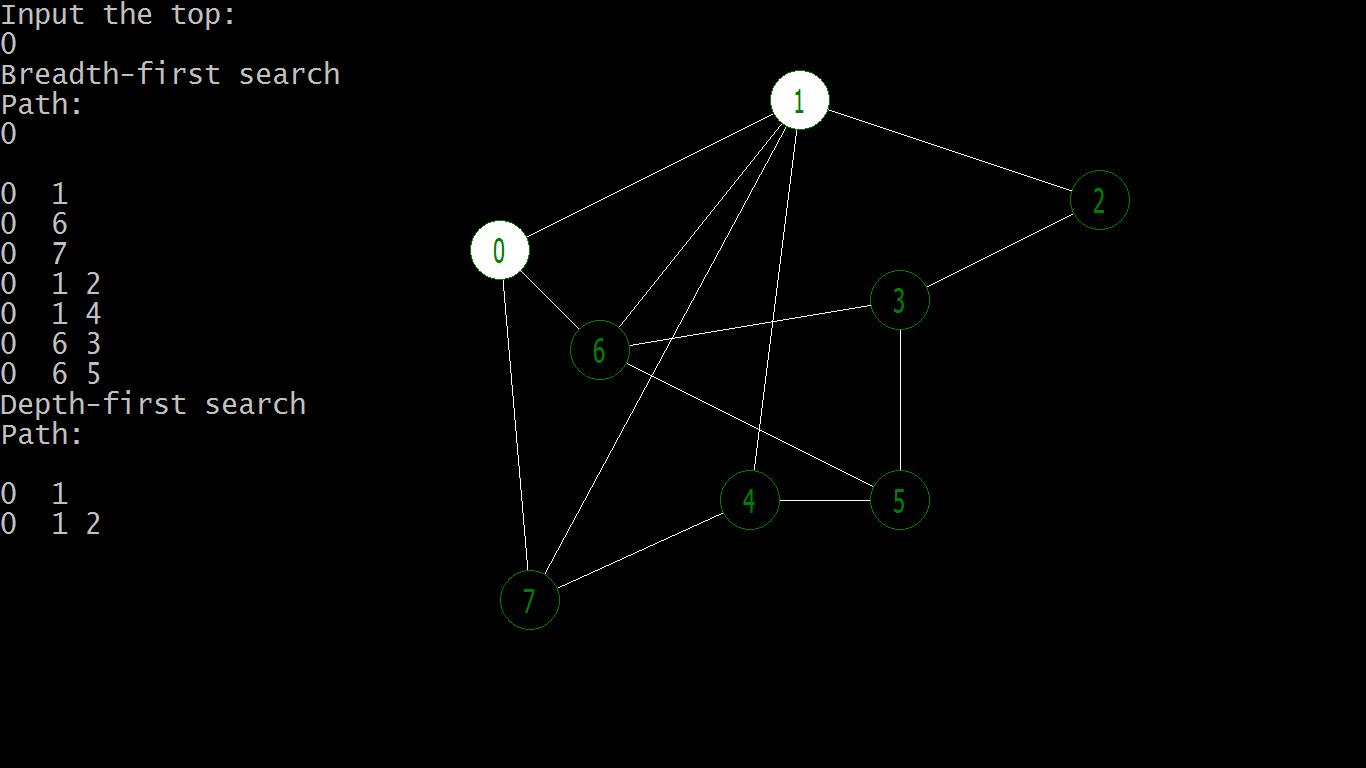
18 color[u] «— BLACK

После инициализации ни одна вершина не окрашивается в белый цвет, поэтому проверка в строке 13 гарантирует, что каждая вершина вносится в очередь не более одного раза, а следовательно, и удаляется из очереди она не более одного раза. Операции внесения в очередь и удаления из нее требуют 0(1) времени, так что общее время операций с очередью составляет О (|V|). Поскольку каждый список смежности сканируется только при удалении соответствующей вершины из очереди, каждый список сканируется не более одного раза. Oбщее время, необходимое для сканирования списков, равно 0(|Е|). Накладные расходы на инициализацию равны 0(V), так что общее время работы процедуры BFS составляет **О (|V| + |Е|)**. Таким образом, время поиска в ширину линейно зависит от размера представления графа G с использованием списков смежности..

В результате тестирования программы, не было обнаружено отклонений от поставленной задачи, а это означает, что программа исправна и готова к использованию.

1. При запуске программы пользователя просят ввести номер вершины, с которой будет осуществляться обход графа.



1. Далее последовательно осуществляется поиск в ширину, выводится информация о путях обхода графа.
2. После выполнения поиска в ширину, с выбранной же вершины осуществляется поиск в глубину , и также выводится информация о путях.

По завершению выполнения обходов при нажатии любой клавиши программа завершает свою работу.