ЛАБОРАТОРНАЯ РАБОТА № 3
ИЗУЧЕНИЕ АЛГОРИТМА ОБХОДА ГРАФА

Цель работы: Изучить способы описания и представления в ЭВМ графов, а также базовый алгоритм поиска путей в графе.

Теоретические сведения

Представление графа G = (V, E) с помощью матрицы смежности предполагает, что вершины перенумерованы в некотором порядке числами 1, 2, ..., n. В таком случае представление графа G с использованием матрицы смежности представляет собой матрицу M размером n × n, такую что:

$$M[i,j] = \begin{cases} 1, \text{ если вершина } v_i \text{ смежна с вершиной } v_j \\ 0, \text{ если вершина } v_i \text{ и } v_j \text{ не смежны} \end{cases}$$

Пример реализации ввода графа в виде матрицы смежности из файла:

```
public class Graph
{
 public int[,] adjacency;
 public int numEdges; // количество вершин графа

 public Graph(string fileName)
 {
 int n, j;
 string line;
 char[] delimiterChars = new char[] { ',', ' ', ';', '.' };
 using (StreamReader file = new StreamReader(fileName))
 {
  n = int.Parse(file.ReadLine()); // ввод размерности
  this.numEdges = n;
  this.adjacency = new int[n, n];
  // ввод матрицы
  for (int i = 0; (i < n) &&
                  ((line = file.ReadLine()) != null); i++) {
```

```
    string[] numbers = line.Split(delimiterChars,
                          StringSplitOptions.RemoveEmptyEntries);
    j = 0;
    foreach (string numString in numbers)
     {
     int x;
     bool canConvert = int.TryParse(numString, out x);
     if (canConvert == true)
     {
      this.adjacency [i, j] = x;
      j++;
     }
    }
   }
  }
}}
```

Все основные служебные операции при работе с графами (например, преобразование графа из одного представления в другое, распечатка или рисование графа) предполагают его систематический обход, то есть посещение каждой вершины и каждого ребра графа.

Одним из наиболее известных алгоритмов являются поиск в ширину (breadth-first search, BFS).

Ключевая идея обхода графа – пометить каждую вершину при первом ее посещении и хранить информацию о тех верши- нах, не все ребра из которых просмотрены.

В процессе обхода графа каждая его вершина будет находиться в одном из трех состояний:

– **неоткрытая** – первоначальное состояние вершины;

– **открытая** – вершина обнаружена, но инцидентные ей ребра не просмотрены;

– **обработанная** (помеченная) – все ребра, инцидентные этой вершине, посещены.

Идея алгоритма: все вершины, смежные с начальной, открываются, то есть помещаются в список, и получают единичную пометку. После этого начальная вершина обработана полностью и имеет пометку 2.

Следующей текущей вершиной становится первая вершина списка. Все ранее не помеченные вершины, смежные с текущей, заносятся в конец списка (становятся открытыми). Текущая вершина удаляется из списка и помечается числом 2. Процесс продолжается, пока список вершин не пуст. Такая организация списка данных называется очередью.

При реализации поиска в ширину используется массив меток `mark`, массив предков `parent` и очередь `Q`. Первоначально каждой вершине в массиве `mark` соответствует значение 0, то есть вершина неоткрытая. Вершина открывается при первом посещении, и ее пометка изменяется на 1. Когда все ребра, исходящие из вершины, исследованы, то она считается обработанной и имеет пометку 2.

При просмотре списка вершин, смежных с текущей, открываются новые вершины. При этом их предком считается текущая вершина. Эта информация сохраняется в массиве `parent` и позволяет восстановить дерево поиска в ширину.

Код алгоритма приведен в примере:

```
public void BFS(Graph g) {
  int[] mark = new int[g.numEdges];
  int[] parent = new int[g.numEdges];
  for (int i = 0; i < g.numEdges; i++)
  {
    mark[i] = 0;
    parent[i] = 0;
  }
  Console.WriteLine("Вершины в порядке обхода");
  Queue<int> Q = new Queue<int>();
  int v = 0;
  mark[v] = 1;
  Q.Enqueue(v);
  Console.Write("{0} ", v);
```

```
  while (Q.Count != 0)
  {
    v = Q.Dequeue();
    for (int i = 0; i < g.numEdges; i++)
    {
      if ((g.adjacency [v, i] != 0) && (mark[i] == 0))
      { // все непомеченные вершины,
        mark[i] = 1;
        Q.Enqueue(i);
        parent[i] =v;
        Console.Write("{0} ", i);
      }
    }
    mark[v] = 2;
  }
  Console.WriteLine();
}
```

У всех вершин, кроме начальной, есть предок. Даная информация хранится в массиве `parent`. Отношение предшествования формирует дерево поиска в ширину с корнем в начальной вершине. Вершины добавляются в очередь в порядке возрастающего расстояния, поэтому дерево поиска определяет кратчайший путь от начальной вершины до любой другой вершины $v \in V$. Этот путь можно воссоздать, следуя по цепи предшественников от v к корню, то есть фактически в обратном направлении.

Порядок выполнения работы

1. Составить программу, осуществляющую чтение матрицы смежности
2. Реализовать алгоритм обхода графа в ширину для поиска путей
3. Программа должна выводить на экран номера смежных вершин в порядке обхода от начальной, вводимой пользователем. Также выводить длину каждого пути.