ЛАБОРАТОРНАЯ РАБОТА № 4
ПОИСК ПУТЕЙ ВО ВЗВЕШЕННОМ ГРАФЕ

Цель работы: Изучить алгоритм Дейкстры поиска путей во взвешенном 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:

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

Алгоритм Дей*кстры* (1959 г.) находит кратчайшие пути от заданной начальной вершины до всех остальных вершин графа.

Основная идея алгоритма: на каждом шаге пытаемся уменьшить кратчайшее расстояние до непросмотренных вершин, используя очередную вершину, длину пути до которой уменьшить уже нельзя.

Алгоритм Дейкстры:

1. Задается множество непросмотренных вершин.
2. Первоначально в нем содержатся все вершины графа, кроме начальной.
3. На каждом шаге из этого множества выбирается та из вершин, расстояние до которой от начальной меньше, чем для других оставшихся вершин.
4. Текущие кратчайшие расстояния от начальной до соответствующей вершины хранятся в массиве.

5. Далее пробуем с помощью ребер выбранной вершины уменьшить длину пути до оставшихся непросмотренными вершин. Если это удается, то массив расстояний корректируется.

В реализации используется матрица расстояний – является копией матрицы смежности графа, только если дуги (ребра) не существует, то вместо нуля в соответствующую ячейку записывается большое число, равное «машинной бесконечности».

На каждом шаге выбирается еще не просмотренная вершина *edgeMin*, расстояние до которой от начальной вершины наименьшее из всех необработанных вершин.

Затем при помощи дуг (ребер) вершины *edgeMin* уменьшается расстояние до непросмотренных вершин (если это возможно).

Длину пути до тех вершин, которые уже просмотрены, уменьшить нельзя: для всех u, где u – уже просмотренная вершина, имеем

$$distance[u] < distance[edgeMin] + matr[u, edgeMin],$$

так как distance[u] < distance[edgeMin] – по алгоритму выбора очередной вершины, а все дуги имеют неотрицательный вес)

Код алгоритма приведен в примере:

```
public DijkstraAlgm(graph g, int vBegin)
{
  Distance = new int[g.numEdges];
  parent = new int[g.numEdges];
  int[,] matr = new int[g.numEdges, g.numEdges];

// инициализация
for (int i = 0; i < g.adjacency.GetLength(0); i++)
  for (int j = 0; j < g.numEdges; j++)
  {
    matr[i, j] = g.adjacency[i, j];
    if (g.adjacency[i, j] == 0)
    matr[i, j] = int.MaxValue;
  }
```

```
HashSet<int> edges= new HashSet<int>();
for (int i = 0; i < g.numEdges; i++)
{
  edges.Add(i);

  distance[i] = matr[vBegin, i];
  if (distance[i] < int.MaxValue)
    parent[i] = vBegin;
}
distance[vBegin] = 0;
parent[vBegin] = -1;
edges.Remove(vBegin);
while (edges.Count != 0)
{
  int minDistance = int.MaxValue;
  int minEdge = -1;
  foreach (int u in edges)
  {
    if (distance[u] < minDistance)
    {
      minDistance = distance[u];
      minEdge = u;
    }
  }
  if (minEdge != -1)
  edges.Remove(minEdge);
foreach (int u in edges)
{
  if (matr[minEdge, u] < int.MaxValue)
  {
    distance[u] = Math.Min(
                 Distance[u],
                 Distance[minEdge] + matr[minEdge, u]
                );
    if (distance[u] ==
                (distance[minEdge] + matr[minEdge, u])
        )
```

```
    {
      parent[u] = minEdge;
    }
  }
}

}
```

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 взвешенной матрицы смежности
2. Реализовать алгоритм Дейкстры обхода графа для поиска кратчайших путей из заданной пользователем вершины.
3. Программа должна выводить на экран путь от начальной вершины до всех остальных, а также длины этих путей.