**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра информатики и прикладной математики**

Алгоритмы и структуры данных

Лабораторная работа №2

«Нахождение кратчайшего пути в графе».

Выполнил Кудряшов А.А.

Группа 2121

Проверил

 Зинчик Александр Адольфович

2013 г.

# Задание

Вариант 2

* А − алгоритм Дейкстры с использованием бинарной кучи,
* В − алгоритм Дейкстры с использованием меток;

 n = 104+1, m = 0, … ,107 с шагом 105, q = 1, r = 106 (нарисовать графики функций TА(m) и ТВ(m) );

Результат выполнения



public static HeapElement[] DijkstraWithBinaryHeap(List<Edge> edges, List<int>[] vertexes, short indexVertexFrom)

 {

 List<HeapElement> tempResultList = new List<HeapElement>();

 HeapElement[] resultList = new HeapElement[vertexes.Length];

 List<HeapElement> tempHeap = new List<HeapElement>();

 for (short i = 0; i < vertexes.Length; ++i)

 {

 tempHeap.Add(new HeapElement(i));

 }

 tempHeap[indexVertexFrom].Value = 0;

 BinaryHeap heap = new BinaryHeap(tempHeap);

 for (int i = 0; i < vertexes.Length; ++i)

 {

 HeapElement currentHeapElement = heap.GetMin();

 tempResultList.Add(currentHeapElement);

 List<int> edgesOfCurrentVertex = vertexes[currentHeapElement.VertexIndex];

 int newWayWeight;

 int oldWayWeight;

 for (int i2 = 0; i2 < edgesOfCurrentVertex.Count; ++i2)

 {

 if (heap.IndexCorrespondanceTable[edges[edgesOfCurrentVertex[i2]].WereVertex] >= heap.list.Count) continue;

 newWayWeight = currentHeapElement.Value + edges[edgesOfCurrentVertex[i2]].Weight;

 oldWayWeight = heap.list[heap.IndexCorrespondanceTable[edges[edgesOfCurrentVertex[i2]].WereVertex]].Value;

 if (newWayWeight < 0) newWayWeight = int.MaxValue;

 if (oldWayWeight > newWayWeight)

 {

 heap.list[heap.IndexCorrespondanceTable[edges[edgesOfCurrentVertex[i2]].WereVertex]].Value =

 currentHeapElement.Value + edges[edgesOfCurrentVertex[i2]].Weight;

 heap.list[heap.IndexCorrespondanceTable[edges[edgesOfCurrentVertex[i2]].WereVertex]].LastVertexIndex = currentHeapElement.VertexIndex;

 heap.Heapify(heap.IndexCorrespondanceTable[edges[edgesOfCurrentVertex[i2]].WereVertex]);

 }

 }

 }

 for (int i = 0; i < tempResultList.Count; ++i)

 {

 resultList[tempResultList[i].VertexIndex] = tempResultList[i];

 }

 return resultList;

 }

 public static HeapElement[] DijkstraWithMark(List<Edge> edges, List<int>[] vertexes, int indexVertexFrom)

 {

 HeapElement[] resultList = new HeapElement[vertexes.Length];

 int inf = 1000001;

 int[] toVertexWay = new int[vertexes.Length];

 short[] lastVertex = new short[vertexes.Length];

 for (int i = 0; i < toVertexWay.Length; ++i)

 {

 toVertexWay[i] = inf;

 }

 toVertexWay[indexVertexFrom] = 0;

 for (int i = 0; i < vertexes.Length; ++i)

 {

 int tempMinValue = toVertexWay[0];

 int MinVertexIndex = 0;

 for (short i1 = 1; i1 < vertexes.Length; ++i1)

 {

 if (toVertexWay[i1] < tempMinValue)

 {

 tempMinValue = toVertexWay[i1];

 MinVertexIndex = i1;

 }

 }

 for (int i1 = 0; i1 < vertexes[MinVertexIndex].Count; ++i1)

 {

 if (toVertexWay[edges[vertexes[MinVertexIndex][i1]].WereVertex] == inf + 1) continue;

 if (toVertexWay[edges[vertexes[MinVertexIndex][i1]].WereVertex] >

 toVertexWay[MinVertexIndex] + edges[vertexes[MinVertexIndex][i1]].Weight)

 {

 toVertexWay[edges[vertexes[MinVertexIndex][i1]].WereVertex] =

 toVertexWay[MinVertexIndex] + edges[vertexes[MinVertexIndex][i1]].Weight;

 lastVertex[edges[vertexes[MinVertexIndex][i1]].WereVertex] = (short)MinVertexIndex;

 }

 }

 resultList[MinVertexIndex] = new HeapElement((short)MinVertexIndex, toVertexWay[MinVertexIndex], lastVertex[MinVertexIndex]);

 toVertexWay[MinVertexIndex] = inf + 1;

 }

 return resultList;

 }

 }

}

Вывод:

В данной реализации, как видно из графиков, оба алгоритма зависят от количества ребер в графе линейно, алгоритм Дейкстры с использованием бинарной кучи быстрее, чем реализация без нее с использованием меток.