**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра информатики и прикладной математики**

Алгоритмы и структуры данных

Лабораторная работа №2

Вариант 3

Выполнил: Гхази Даниэль

Группа P3218

Преподаватель: Зинчик А.А.

2017 г.

**Текст задания**

1. Написать программу, реализующую алгоритм А и алгоритм В.
2. Написать программу, реализующую алгоритм А и алгоритм В, для проведения экспериментов, в которых можно выбирать:

* число n вершин и число m ребер графа,
* натуральные числа q и r, являющиеся соответственно нижней и верхней границей для весов ребер графа.

Выходом данной программы должно быть время работы ТА алгоритма А и время работы ТВ алгоритма В в секундах.

1. Провести эксперименты на основе следующих данных:
   1. n = 1, … ,104+1 с шагом 100, q = 1, r =106, количество ребер: а) m ≈ n2/10, б) m ≈ n2 (нарисовать графики функций TА(n) и ТВ(n) для обоих случаев);
   2. n = 101, … ,104+1 с шагом 100, q = 1, r  = 106, количество ребер: а) m ≈ 100⋅n, б) m ≈ 1000⋅n (нарисовать графики функций TА(n) и ТВ(n) для обоих случаев);
   3. n = 104+1, m = 0, … ,107 с шагом 105, q = 1, r = 106 (нарисовать графики функций TА(m) и ТВ(m) );
   4. n = 104+1, q = 1, r = 1, … ,200 с шагом 1, количество ребер: а) m ≈ n2, б) m ≈ 1000⋅n (нарисовать графики функций TА(r) и ТВ(r) для обоих случаев).
2. Сформулировать и обосновать вывод о том, в каких случаях целесообразно применять алгоритм А, а в каких − алгоритм В.

**Текст программы**

#include "../Headers/shortestPathFinding.h"

// Finds shortest path in graph using Dijkstra's algorithm with d-heap priority queue.

void ldgDijkstraDHeap(std::vector<ElementOfAdjacencyList\*> &ADJ, std::vector<unsigned long long> &dist,

std::vector<unsigned long> &up, GraphParameters \*graphParameters)

{

up.resize(graphParameters->numberOfVertices + 1);

dist.resize(graphParameters->numberOfVertices + 1);

// Array of vertices' names. Name of vertex equals its index.

std::vector<unsigned int> name(graphParameters->numberOfVertices + 1);

// Represents current estimation of distance from vertex 's' to vertex 'name[i]'.

std::vector<unsigned long long> key(graphParameters->numberOfVertices + 1);

// Represents names' indices. Is supposed to be correct at that: index[name[i]] = i

std::vector<unsigned int> index(graphParameters->numberOfVertices + 1);

// Initializing data keepers.

for (int i = 1; i <= graphParameters->numberOfVertices; i++)

{

up[i] = 0;

dist[i] = ULLONG\_MAX;

index[i] = i;

name[i] = i;

key[i] = ULLONG\_MAX;

}

key[graphParameters->startingVertex] = 0;

int nq = graphParameters->numberOfVertices;

PriorityQueue priorityQueue(index, name, key, nq, graphParameters->d);

unsigned int name1 = name[1];

unsigned long long key1 = key[1];

while (nq > 0)

{

priorityQueue.getMin(index, &name1, &key1, name, key, &nq, graphParameters->d);

int i = name1;

dist[i] = key1;

ElementOfAdjacencyList \*p = ADJ[i];

while (p != NULL)

{

unsigned int j = p->name;

unsigned int jq = index[j];

if (dist[i] != ULLONG\_MAX)

{

if (key[jq] > dist[i] + p->weight)

{

key[jq] = dist[i] + p->weight;

priorityQueue.emerge(index, jq, name, key, nq, graphParameters->d);

up[j] = i;

}

}

p = p->next;

}

}

}

**Результаты эксперимента**

**Вывод**

В результате выполнения лабораторной работы был создан генератор случайных графов и реализован алгоритм Дейстры, а также проведены эксперименты для сравнения двух алгоритмов с разными значениями параметра d у d-кучи.

В ходе эксперимента явного доминирования ни один из алгоритмов не показал. Они оба выполняются за соразмерное время. Вероятно, для выявления наиболее эффективного алгоритма необходимо протестировать алгоритм с другими значениями d.