МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритмы на графах

Студент гр. 8304 Птухов Д.А.

Преподаватель Фирсов М.А.

Санкт-Петербург

2019

Вариант 4.

Цель работы.

Построение и анализ алгоритма A* на основе на решения задачи о нахождении минимального пути в графе.

Основные теоретические положения.

Разработайте программу, которая решает задачу построения кратчайшего пути в *ориентированном* графе **методом A***. Каждая вершина в графе имеет буквенное обозначение ("a", "b", "c"...), каждое ребро имеет неотрицательный вес. В качестве эвристической функции следует взять близость символов, обозначающих вершины графа, в таблице ASCII.

Описание алгоритма.

Для решения поставленной задачи был реализован алгоритм A*. В качестве эвристической функции была использована функция h(c1, c2), возвращающая расстояние между двумя символами. Очередь с приоритетами была реализована на основе массиве. В начале каждой итерации в массиве ищется элемент приоритет, которого минимален, он удаляется из очереди, и начинается осмотр всех ребер выходящих из выбранного элемента. Если нашлась вершина путь до которой был больше чем найденный, то данный путь заменяется на найденный. Для хранения значений имен узлов и ребер выходящих из низ был использован словарь. И структура Node, хранящая ребра выходящие из текущей вершины, имя вершины из которой был найден минимальный путь и длина до начальной позиции. Сложность алгоритма: O(|V|*|V| + |E|), где V – множество вершин, а E – множество ребер.

Тестирование.

Таблица 1 – Результаты тестирование

Ввод	**Вывод**
a l m a b 1.000000 a f 3.000000 b c 5.000000 b g 3.000000 f g 4.000000 c d 6.000000 d m 1.000000 g e 4.000000 e h 1.000000 e n 1.000000 n m 2.000000 g i 5.000000 i j 6.000000 i k 1.000000 j l 5.000000 m j 3.000000	For end1: abgenmjl For end2: abgenm
g j m a b 1.000000 a f 3.000000 b c 5.000000 b g 3.000000 f g 4.000000 c d 6.000000 d m 1.000000 g e 4.000000 e h 1.000000 e n 1.000000 n m 2.000000 g i 5.000000 i j 6.000000 i k 1.000000 j l 5.000000 m j 3.000000	For end1: genmj For end2: genm
a f i a b 0 a c 0 b d 0 c e 0 c s 0	For end1: acef For end2: acsi

d f 0 d s 0 s i 0 e f 0	
a e b a b 3.0 b c 1.0 c d 1.0 a d 5.0 d e 1.0	For end1: ade For end2: ab

Вывод.

В ходе работы был построен и анализирован алгоритм A* на основе решения задачи о нахождении минимального пути в графе. Исходный код программы представлен в приложении 1.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <queue>
#include <map>
#include <ctime>

struct ElemInfo
{
    char prev;
    std::vector<std::pair<char, int>> ways;
    int lenToStart = std::numeric_limits<int>::max();
};

size_t h(char c1, char end1)
{
    return std::abs(c1 - end1);
}

void write(char end, char start, std::map<char, ElemInfo>& d)
{
    for (auto& i : d)
    {
        if (i.first == end && i.second.lenToStart ==
std::numeric_limits<int>::max())
        {
            std::cout << "no way\n";
            return;
        }
    }

    std::string s(1, end);
    while (true)
    {
        if (s.back() == start)
            break;
        s += d[s.back()].prev;
    }

    std::reverse(s.begin(), s.end());
    std::cout << s;
}

int main()
{
    setlocale(LC_ALL, "Russian");

    std::map<char, ElemInfo> d;

    char start = 0;
```

```
char end1 = 0;
char end2 = 0;

std::cout << "start end1 end2\n";
std::cin >> start >> end1 >> end2;

char p1 = 0;
char p2 = 0;
float len = 0;
while (std::cin >> p1 >> p2 >> len)
{
    if (len == -1)
        break;

    d[p1].ways.push_back(std::make_pair(p2, len));
    if (p1 == start)
        d[p1].lenToStart = 0;
}

std::vector<char> q;

for (auto& i : d)
    q.push_back(i.first);


auto t1 = clock();
while (!q.empty())
{
    char cur;
    size_t eraseInd;
    int min_priority = -1;
    for (size_t i = 0; i < q.size(); ++i)
    {
        if (d[q[i]].lenToStart == std::numeric_limits<int>::max())
            continue;

        size_t cur_priority = d[q[i]].lenToStart + h(q[i], end1);
        if (cur_priority < min_priority || min_priority == -1)
        {
            min_priority = d[q[i]].lenToStart + h(q[i], end1);
            eraseInd = i;
            cur = q[i];
        }
    }
    if (min_priority == -1)
        break;

    q.erase(q.begin() + eraseInd);

    for (auto& next : d[cur].ways)
    {
        int old_value = d[next.first].lenToStart;
        int new_value = d[cur].lenToStart + next.second;
        if (old_value > new_value)
        {
```

```
                d[next.first].lenToStart = new_value;
                d[next.first].prev = cur;
            }
        }
    }
    auto t2 = clock();

    std::cout << "\nВремя работы: ";
    std::cout << (double)(t2 - t1) / CLOCKS_PER_SEC << "\n";
    std::cout << "\nДля end1: ";
    write(end1, start, d);
    std::cout << "\nДля end2: ";
    write(end2, start, d);
    std::cout << "\n\nСложность алгоритма: O(|V|*|V| + |E|) V - мн-во вершин, E -
мн-во ребер\n";

    return 0;

}
```