**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 |  | Кирьянов Д.И. |
| Преподаватель |  | Размочаева Н.В. |

Санкт-Петербург

2020

## Цель работы.

Разработать программу, которая решает задачу построения кратчайшего пути в ориентированно*м* графе методом А\*. 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.

**Вариант 6.**

Реализация очереди с приоритетами, используемой в А\*, через двоичную кучу.

## Описание алгоритма.

В процессе работы алгоритма для вершин рассчитывается функция f(v) = g(v) + h(v), где

g(v) - наименьшая стоимость пути в v из стартовой вершины,

h(v) - эвристическое приближение стоимости пути от v до конечной цели.

Фактически, функция f(v) - длина пути до цели, которая складывается из пройденного расстояния g(v) и оставшегося расстояния h(v). Исходя из этого, чем меньше значение f(v), тем раньше мы откроем вершину v, так как через неё мы предположительно достигнем расстояние до цели быстрее всего. Открытые алгоритмом вершины хранятся в очереди с приоритетом по значению f(v).

Сложность алгоритма: O(|v|\*|v| + |k|), где

v- множество вершин,

k - множество ребер.

## Описание основных структур данных и функций.

std::map<char, std::vector<std::pair<char, int>>> card;

- словарь для хранения графа. Для каждой вершины хранится вектор с парами, в которых хранится вершина, в которую можно перейти, и расстояние до нее.

std::priority\_queue<std::pair<int, char>, std::vector<std::pair<int, char>>, std::greater<std::pair<int, char>>> priorities;

- очередь с приоритетами через двоичную кучу. Хранит пары с названием вершины для перехода и ее приоритетом. В качестве перегрузки используем std::greater, чтобы функцией top() возвращалось значение с минимальным приоритетом, а не максимальным.

std::map<char, char> prev;

- словарь для хранения вершины, из которой мы пришли в текущую.

std::map<char, int> cost;

- словарь для хранения расстояния от старта до текущей вершины.

int heuristic(char& first, char& second);

- эвристическая функция, подсчитывающая близость символов, обозначающих вершины графа, в таблице ASCII.

void print(char& start, char& finish, std::map<char, char>& prev, int& menu);

- функция, выводящая конечный результат. В зависимости от значения menu происходит вывод либо в консоль, либо в файл.

**Тестирование.**

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| a e  a b 3.0  b c 1.0  c d 1.0  a d 5.0  d e 1.0 | Result: ade |
| a g  a b 3.0  a c 1.0  b d 2.0  b e 3.0  d e 4.0  e a 1.0  e f 2.0  a g 8.0  f g 1.0 | Result: ag |
| a f  a c 1.0  a b 1.0  c d 2.0  b e 2.0  d f 3.0  e f 3.0 | Result: abef |
| a d  a b 1.0  b c 9.0  c d 3.0  a d 9.0  a e 1.0  e d 3.0 | Result: aed |
| b e  a b 1.0  a c 2.0  b d 7.0  b e 8.0  a g 2.0  b g 6.0  c e 4.0  d e 4.0  g e 1.0 | Result: bge |

# Вывод.

В ходе выполнения данной работы была написана программа, которая решает задачу построения кратчайшего пути в ориентированном графе методом А\*.

**ПРИЛОЖЕНИЕ А.  
ИСХОДНЫЙ КОД**

#include <iostream>

#include <vector>

#include <map>

#include <algorithm>

#include <queue>

#include <fstream>

#include <string>

std::string input\_way = "C:/Users/Danielka/source/repos/paa\_lr2\_2/input";//пути до файлов

std::string output\_way = "C:/Users/Danielka/source/repos/paa\_lr2\_2/output";

int heuristic(char& first, char& second){//эвристика

return std::abs(first-second);

}

void print(char& start, char& finish, std::map<char, char>& prev, int& menu) {//вывод результата

std::vector<char> result;

char current = finish;

result.push\_back(current);

while (current != start) {//проходим от конца до начала

current = prev[current];

result.push\_back(current);//записываем результат в вектор

}

if (menu == 1) {//вывод в консоль

std::cout << std::endl << "Result: ";

for (unsigned long int i = 0; i < result.size(); ++i) {

std::cout << result[result.size() - i - 1];

}

}

else {//вывод в файл

std::ofstream file;

file.open(output\_way);

if (!file.is\_open()) {

std::cout << "Can't open file!\n";

}

file << "Result: ";

for (unsigned long int i = 0; i < result.size(); ++i) {

file << result[result.size() - i - 1];

}

}

}

int main() {

std::cout << "How do you want to input?" << std::endl << std::endl

<< "Press 1 to input from console." << std::endl//выбор как считать

<< "Press 2 to input from file." << std::endl;

char start, finish='0';

std::map<char, std::vector<std::pair<char, int>>> card;//хранение графа

char first, second;

float len;

int menu=0;

while (menu != 1 && menu != 2) {//пока не введется нужная цифра

std::cin >> menu;

if (menu == 1) {//считывание с консоли

std::cin >> start >> finish;

while (std::cin >> first >> second >> len) {

if (len == -1)

break;

card[first].push\_back(std::make\_pair(second, len));

}

}

else if (menu == 2) {//считывание из файла

std::ifstream file;

file.open(input\_way);

if (!file.is\_open()) {

std::cout << "Can't open file!" << std::endl;

return 0;

}

file >> start >> finish;

while (file >> first >> second >> len) {

card[first].push\_back(std::make\_pair(second, len));

}

}

else {//если неверно введена цифра, выводится сообщение

std::cout << std::endl << "Wrong choice! Try again!" << std::endl;

}//а цикл продолжается

}

std::priority\_queue<std::pair<int, char>, std::vector<std::pair<int, char>>, std::greater<std::pair<int, char>>> priorities;

priorities.push(std::make\_pair(0, start));//объявление очереди через двоичную кучу, вносим начальную вершину

std::map<char, char> prev;//откуда пришли в вершину

std::map<char, int> cost;//стоимость всего пути до вершины

prev[start] = start;

cost[start] = 0;

while (!priorities.empty()) {//пока очередь не станет пустой

char current = priorities.top().second;//берем вершину с наименьшим приоритетом

priorities.pop();

if (current == finish)//если доходим до конца

break;//то завершаем цикл

for (auto& next : card[current]) {//для текущей вершины прогоняем все возможные пути

int new\_cost = cost[current] + next.second;//считаем стоимость пути в каждую из новых вершин

if (!cost.count(next.first) || new\_cost < cost[next.first]) {//если впервые в этой вершине

cost[next.first] = new\_cost;//или новый возможный путь короче, то обновляем стоимость

int priority = new\_cost + heuristic(next.first, finish);//считаем приоритет

priorities.push(std::make\_pair(priority, next.first));//заносим в очередь

prev[next.first] = current;//в следующую вершину пришли из текущей

}

}

}

menu = 0;

std::cout << std::endl << "How do you want to output?" << std::endl << std::endl

<< "Press 1 to output by console." << std::endl//выбор как вывести

<< "Press 2 to output into file." << std::endl;

while (menu != 1 && menu != 2) {//пока не введется нужная цифра

std::cin >> menu;

if (menu != 1 && menu != 2)//если неверно, то выводится сообщение и цикл продолжается

std::cout << std::endl << "Wrong choice! Try again!" << std::endl;

}

print(start, finish, prev, menu);//функция вывода результата

return 0;

}