**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина) Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: Алгоритм А\***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент гр. 7383 | |  | Сычевский Р.А. |  |
| Преподаватель | |  | Жангиров Т.Р. |  |
|  |  |
|  |  |  |  |  |

Санкт-Петербург

2019

**Содержание**

Цель работы 3

Реализация задачи 4

Исследование алгоритма 5

Тестирование 6

1. Процесс тестирования 6

2. Результаты тестирования 6

Вывод 7

Приложение А. Тестовые случаи 8

Приложение Б. Исходный код 9

**Цель работы**

Цель работы: познакомиться с алгоритмом поиска А\*, создать программу, осуществляющую поиск кратчайшего пути в графе с помощью алгоритма А\*.

Формулировка задачи: Разработайте программу, которая решает задачу построения кратчайшего пути в ориентированном графе методом А\*. Каждая вершина в графе имеет буквенное обозначение (“a”, “b”, “c” ...), каждое ребро имеет неотрицательный вес. Вариант 4с: Списки смежности. Модификация A\* с двумя финишами.

**Реализация задачи**

Программу было решено писать на языке программирования C++.

Для реализации поставленной задачи был создан класс vertex.

class vertex{

public:

int length;

char last;

int painted;

vertex(){

length = -1;

painted = 0;

last = '0';

}

};

Класс vertex хранит вершину, а именно её имя, путь до неё и вспомогательную переменную painted, использующуюся в матрице смежности.

Исходный код программы представлен в приложении Б.

**Исследование алгоритма**

Сложность алгоритма составляет O(|V|·|E|), где |V| – количество вершин в графе, |E| - количество ребер в графе. При обработке каждое вершины алгоритм обрабатывает все ребра графа, в наихудшем случае алгоритм обработает каждую вершину.

**Тестирование**

1. **Процесс тестирования**

Программа собрана в операционной системе Ubuntu 18.04.2 LTS bionic компилятором g++ version 7.3.0 (Ubuntu 7.3.0-27ubuntu1~18.04). В других ОС

* компиляторах тестирование не проводилось.
  1. **Результаты тестирования**
* результате тестирования было выявлено и исправлено некорректное поведение программы в ситуации, когда после нахождения пути программа прекращала поиск и, порой, выводила некорректный ответ. Тестовые случаи представлены в приложении А.

**Вывод**

* ходе выполнения данной работы был изучен метод поиска кратчайшего пути А\*. Была написана программа, применяющая метод А\* для поиска кратчайшего пути в графе. Сложность реализованного алгоритма составляет O(|V|·|E|).

**ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ**

|  |  |  |
| --- | --- | --- |
| Входные данные | Результат |  |
|  |  |  |
| a e |  |  |
| a b 3.0 |  |  |
| b c 1.0 | ade |  |
| c d 1.0 |  |
|  |  |
| a d 5.0 |  |  |
| d e 1.0 |  |  |
|  |  |  |
| a e |  |  |
| a b 1.0 |  |  |
| b c 1.0 | abe |  |
| c d 1.0 |  |
|  |  |
| d b 1.0 |  |  |
| b e 10.0 |  |  |
|  |  |  |

**ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД**

#include <iostream>

#include <cmath>

using namespace std;

#define ARR\_SIZE 27

class vertex{

public:

int length;

char last;

int painted;

vertex(){

length = -1;

painted = 0;

last = '0';

}

};

template <typename T>

void full\_elem( T\* a, T b)

{

\*(a) = b;

return;

}

int evr\_f(char a, char b){

return abs(a-b);

}

int main(){

vertex arr[ARR\_SIZE][ARR\_SIZE];

for(int i = 0; i < ARR\_SIZE; i++)

for(int j = 0; j < ARR\_SIZE; j++){

if(i == j)

arr[i][j].length = 0;

arr[i][j].last = '0';

}

vertex start;

cin >> start.last;

vertex end;

cin >> end.last;

char tmp1;

char tmp2;

int tmp\_length = 0;

double tmp\_buf = 0;

cin >> tmp1;

cin >> tmp2;

cin >> tmp\_buf;

tmp\_length = tmp\_buf;

full\_elem(&arr[1][0].last, tmp1);

full\_elem(&arr[1][0].length, 0);

full\_elem(&arr[0][1].last, tmp1);

full\_elem(&arr[0][1].length, 0);

full\_elem(&arr[2][0].last, tmp2);

full\_elem(&arr[2][0].length, 0);

full\_elem(&arr[0][2].last, tmp2);

full\_elem(&arr[0][2].length, 0);

full\_elem(&arr[1][2].length, tmp\_length);

int i = 1;

int j = 1;

int count = 1;

while(cin >> tmp1){

cin >> tmp2;

cin >> tmp\_buf;

tmp\_length = tmp\_buf;

count++;

for(i; arr[i][0].last != tmp1; i++){

if(arr[i][0].length == -1){

arr[i][0].last = tmp1;

arr[i][0].length = 0;

arr[0][i].last = tmp1;

arr[0][i].length = 0;

break;

}

}

for(j; arr[0][j].last != tmp2; j++){

if(arr[0][j].length == -1){

arr[0][j].last = tmp2;

arr[0][j].length = 0;

arr[j][0].last = tmp2;

arr[j][0].length = 0;

break;

}

}

arr[i][j].length = tmp\_length;

i = 1;

j = 1;

}

for(i = 1; arr[i][0].length != -1; i++)

if(arr[i][0].last == start.last){

start.length = i;

arr[i][0].painted = 1;

arr[0][i].painted = 1;

}

for(i = 1; arr[i][0].length != -1; i++)

if(arr[i][0].last == end.last){

end.length = i;

}

if(start.last == end.last){

cout << start.last;

return 0;

}

int min = 0;

vertex tmp\_ver;

vertex need\_to\_add;

int add\_min = 0;

for(i = 1; i <= count; i++){

if(arr[start.length][i].length > 0)

arr[i][i].length = start.length;

}

while(arr[end.length][0].painted != 1){

for(i = 1; i <= count; i++){

if(arr[i][0].painted == 1)

continue;

if(arr[start.length][i].length < 1)

continue;

if(!min && arr[start.length][i].length > 0){

min = arr[start.length][i].length;

tmp\_ver.painted = i;

tmp\_ver.length = evr\_f(end.last, arr[i][0].last);

tmp\_ver.last = arr[i][0].last;

continue;

}

if(min && arr[start.length][i].length > 0){

if((min + tmp\_ver.length) >= (arr[start.length][i].length + evr\_f(end.last, arr[i][0].last))){

if(tmp\_ver.last - arr[i][0].last < 0){

min = arr[start.length][i].length;

tmp\_ver.painted = i;

tmp\_ver.length = evr\_f(end.last, arr[i][0].last);

tmp\_ver.last = arr[i][0].last;

continue;

}

}

}

}

for(j = 1; j <= count; j++){

if(arr[tmp\_ver.painted][j].length > -1 && tmp\_ver.painted!=j){

if(arr[start.length][j].length < 0 || arr[start.length][j].length >= (arr[tmp\_ver.painted][j].length + min)){

arr[start.length][j].length = arr[tmp\_ver.painted][j].length + min;

arr[j][j].length = tmp\_ver.painted;

}

}

}

arr[tmp\_ver.painted][0].painted = 1;

arr[0][tmp\_ver.painted].painted = 1;

tmp\_ver.painted = 0;

tmp\_ver.length = 0;

min = 0;

}

char list[count];

i = 0;

int tmp\_int = end.length;

char tmp\_char = '-';

while(tmp\_char != start.last){

tmp\_char = arr[tmp\_int][0].last;

list[i] = tmp\_char;

tmp\_int = arr[tmp\_int][tmp\_int].length;

i++;

}

i--;

while(i > -1){

cout << list[i];

i--;

}

return 0;

}