**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Информатики

**Отчет по лабораторной работе №3**

по предмету «Кроссплатформенные технологии программирования»

на тему:

«Алгоритм А\*(A star)»

Выполнил: студент группы БВТ 1802

Молчанова А.А.

Руководитель:

Полянцева К.А

Москва 2020

**Цель работы**

Дописать код алгоритма поиска A\*. Концепция алгоритма довольно проста, начиная с исходного местоположения, алгоритм постепенно строит путь от исходной точки до места назначения, используя наикратчайший путь, чтобы сделать следующий шаг. Это гарантирует, что полный путь будет также оптимальным.

**Задания**

Cкачать исходные файлы для данной лабораторной работы. Единственные классы, которые должны быть изменить это Location и AStarState.

Для начала необходимо подготовить класс Location для совместного использования с классами коллекции Java. Поскольку будет использоваться контейнеры для хеширования для выполнения данного задания, то для этого необходимо:

* Обеспечить реализацию метода equals ().
* Обеспечить реализацию метода hashcode().

Добавить реализацию каждого из этих методов в класс Location, следуя шаблонам в классе. После этого использовать класс Location в качестве ключевого типа в контейнерах хеширования, таких как HashSet и HashMap.

После того, как класс Location готов к использованию, завершить реализацию класса AStarState.

Состояние А\* состоит из двух наборов вершин, один из открытых вершин и другой из закрытых. Чтобы упростить алгоритм, вершины будут храниться в хэш-карте, где местоположение вершин является ключом, а сами вершины являются значениями.

Добавить два (нестатических) поля в класс AStarState с таким типом, одно для "открытых вершин" и другой для "закрытых вершин". Инициализировать каждое из этих полей для ссылки на новую пустую коллекцию.

После создания и инициализации полей, реализовать следующие методы в классе AStarState:

* public int numOpenWaypoints();
* public Waypoint getMinOpenWaypoint();
* public boolean addOpenWaypoint(Waypoint newWP);
* public boolean isLocationClosed(Location loc);
* public void closeWaypoint(Location loc).

**Коды программ**

***Location.java***

import java.util.Objects;

public class Location{

public int xCoord;

public int yCoord;

public Location(int x, int y){

xCoord = x;

yCoord = y;}

public Location() {

this(0, 0);}

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Location location = (Location) o;

return xCoord == location.xCoord &&

yCoord == location.yCoord;}

public int hashCode() {

return Objects.hash(xCoord, yCoord);}}

***AStarState.java***

import java.util.\*;

public class AStarState{

private Map2D map;

private Map<Location, Waypoint> Opened = new java.util.HashMap<Location, Waypoint>();

private Map<Location, Waypoint> Closed = new java.util.HashMap<Location, Waypoint>();

public AStarState(Map2D map){

if (map == null)

throw new NullPointerException("map cannot be null");

this.map = map;}

public Map2D getMap(){

return map;}

public Waypoint getMinOpenWaypoint(){

if (Opened.size() == 0) return null;

ArrayList<Waypoint> waypoints = new ArrayList<Waypoint>(Opened.values());

float mincost = waypoints.get(0).getTotalCost();

Waypoint min = waypoints.get(0);

for (int i = 1; i < waypoints.size(); i++) {

if (waypoints.get(i).getTotalCost() < mincost) {

min = waypoints.get(i);

mincost = min.getTotalCost();}}

return min;}

public boolean addOpenWaypoint(Waypoint newWP){

if (Opened.get(newWP.getLocation()) == null ) {

Opened.put(newWP.getLocation(), newWP);

return true;}

else{

if (Opened.get(newWP.getLocation()).getPreviousCost() >

newWP.getPreviousCost()) {

Opened.put(newWP.getLocation(), newWP);

return true;}}

return false;}

public int numOpenWaypoints(){

return Opened.size();}

public void closeWaypoint(Location loc){

Closed.put(loc, Opened.remove(loc));}

public boolean isLocationClosed(Location loc){

if (Closed.containsKey(loc)) return true;

return false;}}

**Результат работы кода**

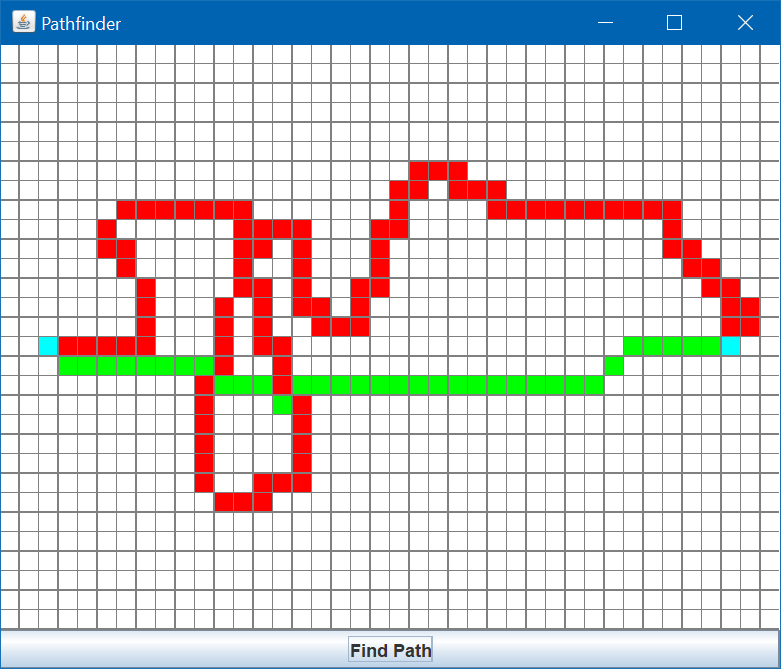


Рисунок 1 – Работа программы

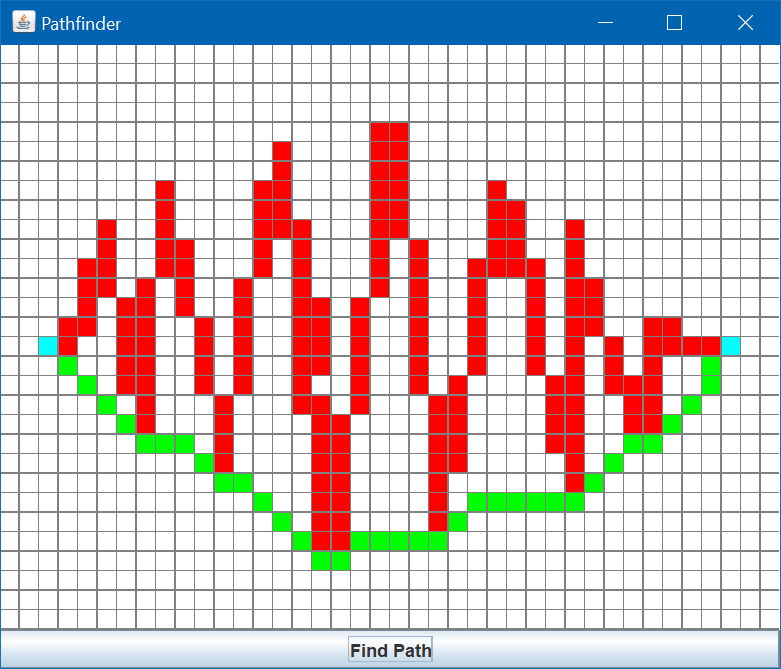


Рисунок 2 – Работа программы

**Вывод**

В ходе лабораторной работы подготовили класс Location для совместного использования с классами коллекции Java. Обеспечили реализацию метода equals (). Обеспечили реализацию метода hashcode(). После этого использовали класс Location в качестве ключевого типа в контейнерах хеширования, таких как HashSet и HashMap.

Добавили два (нестатических) поля в класс AStarState с таким типом. Инициализировали каждое из этих полей для ссылки на новую пустую коллекцию. После создания и инициализации полей, реализовали методы.