Recruitment assigment - Hare Documentation

Dawid Ciemała

March 23, 2023

Contents

1 Program purpose

The goal of the program is to find the shortest path using A* algorithm beginning from node marked as ”z” and ending on node ”n” and returning the number of moves needed to travel to the end.

2 User manual

2.1 Usage

To use this algorithm, first create an object of class Field and pass to its constructor the path to the input data. Then create an object of class AStar and pass to its constructor the previously created Field object. Then you can run the run() method of object AStar.

2.2 Structure of input files

Files used for importing field should have the following structure:

- Two integer numbers separated by space. First number is the number of rows, and the second is the number of columns.
- Structure of fields by characters. There must be one ”z” character and one ”n”.

Input file structure should look like this:

```
4 5
.zx.x
.xx..
..x.x
x..n.
```

It is important to write it properly because the program is not validating this data so improper data will result in errors.

3 Program’s design

The program consists of three classes:

- Node - basic structure of field that we are traveling through
- Field - set of nodes
- AStar - class implementing A* algorithm on field

It operates on a field of nodes which can be marked as:

- ”z” - start node
- ”.” - walkable node
- ”x” - un-walkable node
- ”n” - end node

3.1 Movement pattern

We can only move through the centers of Nodes.

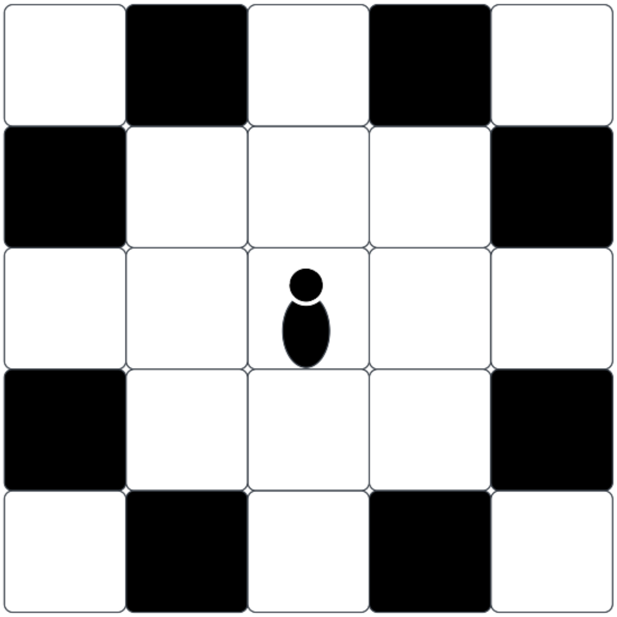

Image above shows how the movement pattern looks like. We can move only by $\sqrt{5}$ and if our nodes have dimensions 1x1 pattern looks like this.

3.2 A* algorithm

Algorithm for finding the shortest paths. It’s based on nodes and their costs. It will choose from the neighboring nodes with the smallest cost which is based on distance from starting point and ending point. When it will reach the ending point it will recreate path based on parents of each node.

3.3 UML

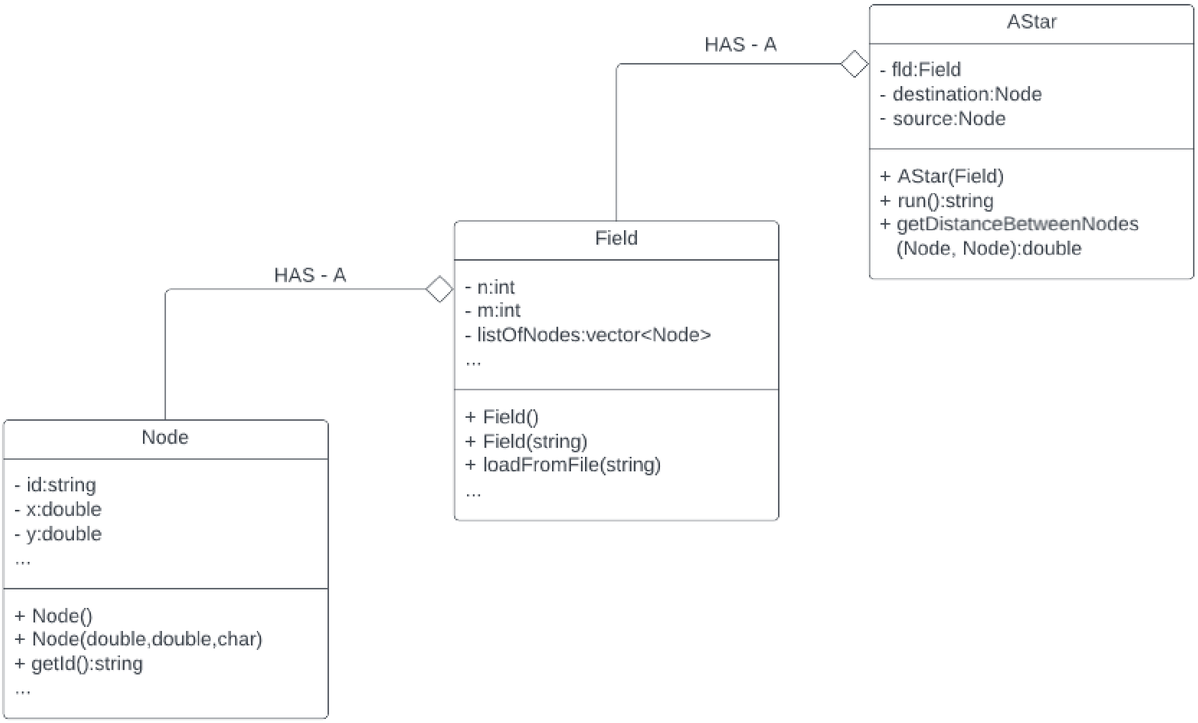

Image above shows UML (Unified Modeling Language) structure of the program.

4 Tests

Program includes 3 basic tests of AStar's run method. They are made by using Google Test framework.

5 Code documentation

5.1 Node class

5.1.1 Attributes

std::string id - id of node created form floor of x and y values
double x - x value of node in euclidean space
double y - y value of node in euclidean space
char type - type od node. It determines if node is star, end, walkable or not
double gCost - distance from start

double hCost - distance to end
std::string parentID - id of parent from which we arrived

5.1.2 Methods

Node() - default constructor. Sets x and y values to 0, type to ”x”, id to ”0” and parent id to empty string
Node(double nx,double ny,char nType) - constructor. sets x, y and type attributes to that given as parameters. It also counts id and sets parentID as empty string
std::string getId() const - returns id
std::string getParentId() const - returns parentId
double getX() const - returns x
double getY() const - returns y
double getGCost() const - returns gCost
double getHCost() const - returns hCost
void setGCost(double newCost) - sets gCost to that given as parameter
void setHCost(double newCost) - sets hCost to that given as parameter
char getType() const - returns type of node
void setParentId(std::string newID) - sets parentID to that given as parameter
bool operator==(const Node & other) const -overloads == operator. It compares two nodes by id
bool operator!=(const Node & other) const -overloads != operator. It compares two nodes by id
void printAllData() const - prints all variables of node
double fCost() - counts f cost which is f cost added to h cost

5.2 Field class

5.2.1 Attributes

int n - number of rows in field
int m - number of columns in field
std::vector<Node>listOfNodes - list of all nodes in field

5.2.2 Methods

Field() - default constructor. Sets n and m to 0
explicit Field(std::string path) - constructor. It runs loadFromFile method based on path given as parameter
void loadFromFile(const std::string & path) - It sets n,m and list of nodes based on file given as parameter

int getM() const - returns m
int getN() const - returns n
std::vector<Node>getListOfNodes() const - returns list of all nodes in field
std::vector<Node>findNeighbours(Node & target) - returns a list of all nodes that are "next to" node given as a parameter. Nodes "next to" target are specified in function (they are matching pattern)
Node findNodeById(std::string id) - returns node that's id matches that given as parameter
Node findNodeByType(char type) - returns node that's type matches that given as parameter

5.3 AStar class

5.3.1 Attributes

Field fld - field on which we will be performing algorithm
Node destination - node that we are going to
Node source - node that we are starting from

5.3.2 Methods

explicit AStar(Field f) - constructor
std::string run() - run algorithm and return number of jumps to reach destination. If it's impossible to reach destination method will return "NIE"
static double getDistanceBetweenNodes(const Node & n1,const Node & n2) - return distance between node n1 and n2