Intro to AI - Assignment 1

Haochen "Justin" Ji

September 28, 2022

Problem 1

a

1. To randomly generates 50 grids with size 100x50, cd to the directory "scripts" under the root folder, then using following command line to run the script

   `./grids_generator.sh`

   Notice: Please create target directory before using this script

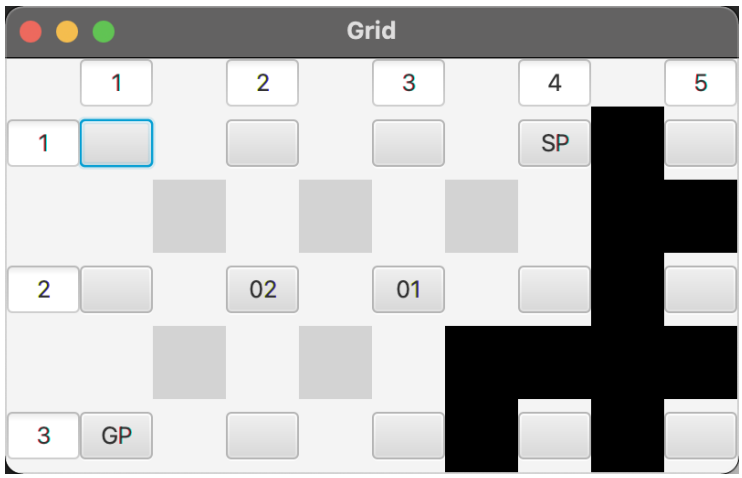
2. To run the basic terminal version of program, cd to the directory "scripts" under the root folder, then using following command line to run the script

   `./terminal_version.sh`

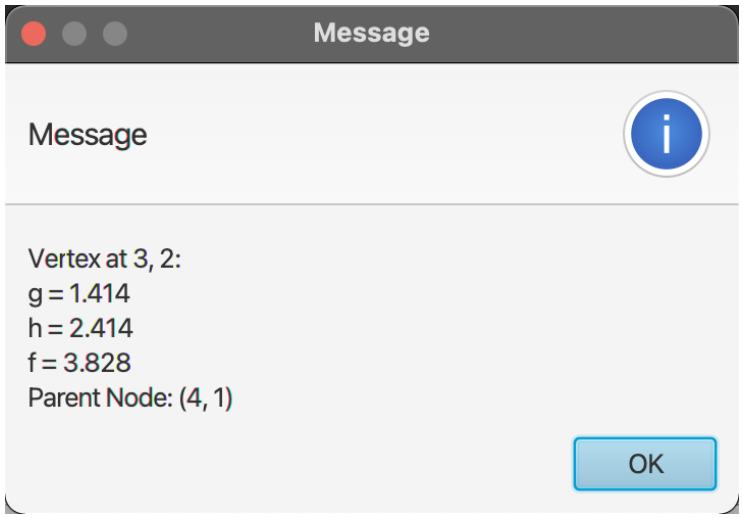
   Then you can type query and after click enter you can query the vertex's values by type in the x and y coordinates split by "," or " "(space)

3. To run the JavaFX version of program, cd to the directory "scripts" under the root folder, then using following command line to run the script

   `./javafx_version.sh`

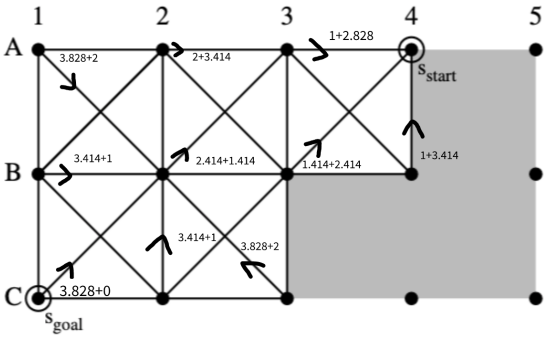


Black squares are blocked edge/grid; GP means Goal Point and SP means Start Point. Then you can click any button on the window to query any node

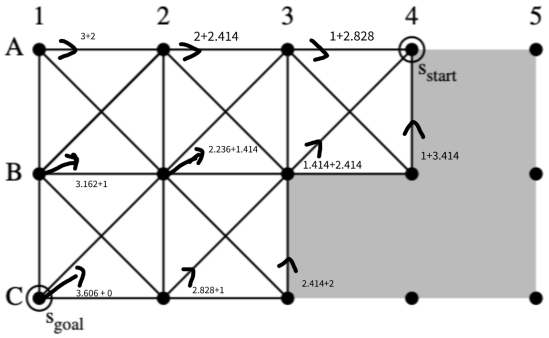


Any script above will compile java files into bin folder and run the program. When it asks the grid file/folder, using absolute path or path from the root folder.

b


1
2
3
4
5
A
B
C
3.828+2
2+3.414
1+2.828
s start
3.414+1
2.414+1.414
1.414+2.414
1+3.414
3.414+1
3.828+2
3.828+0
s goal

a*


1
2
3
4
5
A
B
C
3+2
2+2.414
1+2.828
s start
2.236+1.414
1.414+2.414
1+3.414
3.162+1
3.606 + 0
2.828+1
2.414+2
s goal

theta*

c

In order to implement A* algorithm, I need to implement MinHeap as the data structure that stores all fringe nodes, Nodes that stores the coordinates and g value that the distance from the start node, and functions that calculates the h value that predict the distance from the goal node, function that updates vertex, and funtion that finds accessable neighbor nodes.

d

In order to implement Theta* algorithm, all implemented things in A* are needed since Theta* is based on A*. Other than that, I need to implement line of sight function that can check the accessability between neighbors of current node and the parent of current node, and a updated "update vertex" method.

e

For eight direction movement, in order to find the shortest path, we want to move along diagonal as much as possible because we only need $\sqrt{2}$ to move 1 unit on both x and y direction. So the maximum diagonal path between two points is $\sqrt{2} * min(|s^x - s^x_{goal}|, |s^y - s^y_{goal}|)$. Then for the rest distance, we want to take $max(|s^x - s^x_{goal}|, |s^y - s^y_{goal}|) - min(|s^x - s^x_{goal}|, |s^y - s^y_{goal}|)$ step to move forward to the goal point in straight line. So the total minimum distance between two points is

$$\sqrt{2} * min(|s^x - s^x_{goal}|, |s^y - s^y_{goal}|) + max(|s^x - s^x_{goal}|, |s^y - s^y_{goal}|) - min(|s^x - s^x_{goal}|, |s^y - s^y_{goal}|)$$

And this h function is consistent.

f

The Min Heap I implemented is sorted based on f value, which is the sum of g and h value. When two nodes having same f value, the larger g value one will be considered as a smaller one. The root node is located at index 0, and for any node with index i, its left child is at the index 2i + 1 and its right child is at the index 2i + 2.