CAP 4621 INTRODUCTION TO AI SPRING 2021

**A\* Search Algorithm**

**DUE: 2/15/2021, 11:59 PM**

**DESCRIPTION**

Write a program which implements the A\* algorithm on an arbitrary map described through “maze.txt”. **To form a group of two please go to Canvas -> click “People” tab -> click “Groups” tab on the top->enroll in one of the “**[**Assignment 1 – A\***](https://usflearn.instructure.com/courses/1523514/groups#tab-114790) **search \_#\_" group. No matter you are doing this assignment in a group of two or individually, you must enroll in a group BEFORE you or your partner make any submission for this assignment.**

1. **Rule**

The objective is to find the “best” path (shortest distance or fewest nodes) from a selected starting node to the goal node “O”. Each node has a name, an x and y position on a 2-dimensional space, and a colored (red or blue) directional arrow (N, S, E, W, NE, NW, SE, SW). From any node you must travel in the direction of the arrow associated with that node and can only stop at a node that is associated with an arrow that has a different color. You don’t need to stop at the first node that meets this condition. For example, below, when you get to “B16” you can travel to “R25” or to R61. Below are two path examples if the starting node is “R0”

Path 1: R0🡪B16🡪R61🡪B5🡪R3🡪B2🡪R20🡪B18🡪O

A picture containing diagram

Description automatically generated

Path 2: R0🡪B16🡪R61🡪B37🡪R23🡪B15🡪R6🡪B62🡪R55🡪B10🡪R8🡪B35🡪R21🡪B39🡪O

A picture containing diagram

Description automatically generated

There may be many other paths. Your goal is to find the “best” path and your A\* search should be guided by the “heuristic” of the algorithm.

1. **Implementation**

Two A\* algorithm heuristics, “Fewest nodes” and “Straight-line distance" should be implemented, and the user should select which one should be used for a given run. Note: The “Fewest nodes” is the path in which fewest number of nodes are included in the solution path.

The program should also allow the user to select whether to show just the optimal path found or a “step-by-step” path to the solution ending with the optimal path. This “step-by-step” option should show how the A\* algorithm is progressing through the search. At every step it should show:

* **The name of the node** selected from which to search next.
* **From the node selected**, name all the permissible nodes to where to travel next.
* **After all possible paths are evaluated**, list all the nodes at the end of paths that are still being considered in the search and the estimated total path distance.

The user input at runtime consists of the following:

* The start node
* “step-by-step” option or not.
* The heuristic to use
* Fewest nodes, or
* Straight-line distance

**INPUT FILE FORMAT**

The name of the file will always be “maze.txt”. Its contents may vary but the file will be formatted as follows. The input begins with two positive characters on a line indicating the number of rows r and columns c of the maze, respectively. The next r lines contain the color and directional information for each node in the maze. Each line has c values, where each value represents the <colorID>-<direction>-<x-coordinate>-<y-coordinate>. The <colorID> codes R and B represent red and blue followed by the node ID; the <direction> codes represent north N, east E, south S, west W, northeast NE, southeast SE, southwest SW, or northwest NW; the <x-coordinate>-<y-coordinate> represents X and Y coordinate for the node. You may assume that the node ‘O’ will always be in the bottom-right corner of the maze.

A picture containing diagram

Description automatically generated

**OUTPUT FILE**

The solution path must be written to an output file called “maze-sol.txt”. This output file should contain the information of the heuristic method selected; if step-by-step option was selected; and the start node. It should then include the results of the program execution. See “mazeSol-SBS.txt” for simple step by step output file example. See “mazeSol-notSBS.txt” for simple non step-by-step output file example.

**If the “step-by-step” option is NOT selected**, the program output should be the solution path from start city to end city as a series of segments. Each segment should include its length (in distance or number of nodes depending on the heuristic selected). The total length of the path should also be included at the end. – for example, if the start city is R0 the output might be,

|  |  |
| --- | --- |
| R0 to B16 distance: 35.0,  B16 to R61 distance: 149.777,  R61 to B5 distance: 183.0,  B5 to R3 distance: 5.0,  R3 to B2 distance: 1.0,  B2 to R20 distance: 35.057,  R20 to B18 distance: 2.0,  B18 to O distance: 154.69,  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Total path length: 565.524 | R0 to B16 distance: 1,  B16 to R61 distance: 1,  R61 to B5 distance: 1,  B5 to R3 distance: 1  R3 to B2 distance: 1  B2 to R20 distance: 1  R20 to B18 distance: 1  B18 to O distance: 1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Total path length: 8 |

Note that the final path must contain blue and red alternating nodes.

**If the “step-by-step” option IS selected**, the program output, prior to showing the solution path as above, should provide the following information at every step. Note that this is not the solution for the maze.txt, but just an example on the print format for the “step-by-step” option:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

City selected: R0

Possible node to travel: B16

This should be distance traveled + estimated distance to the goal node. I.e. (R0 to B16)+(R16 to O) = 35+160.461=195.461

Node at the end of possible path: B16(195.462)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

City selected: B16

Possible node to travel: R25 R61

Node at the end of possible path: R25(202.0) R61(223.777)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

City selected: R25

Possible node to travel: B16

Node at the end of possible path: R61(223.777)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

City selected: R61

Possible node to travel: B53 B45 B37 B29 B13 B5

Node at the end of possible path: B53(376.702) B45(417.318) B37(450.501) B29(477.965) B13(552.909) B5(554.887)

No duplicates allowed, and should always be the shortest estimated distance

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

City selected: B53

Possible node to travel: R61

Node at the end of possible path: B45(417.318) B37(450.501) B29(477.965) B13(552.909) B5(554.887)

ETC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The final solution path is:

R0 to B16 distance: 35.0,

B16 to R61 distance: 149.777,

R61 to B5 distance: 183.0,

B5 to R3 distance: 5.0,

R3 to B2 distance: 1.0,

B2 to R20 distance: 35.057,

R20 to B18 distance: 2.0,

B18 to O distance: 154.69,

Total path length:565.524

**SUBMISSION**

Through a link in Canvas. For those working in pairs, perform a single submission.

**DELIVERABLES**

Assignment should be submitted as a single ZIP file which include source files, and a readme text file with your and your partner’s name and instructions about how to compile and execute the program from a command line.

**PROGRAMMING GUIDELINES**

You are allowed to use the programming language of your preference. For grading purposes, the use of the following languages is recommended: Python, Java, C (gcc), C++ (g++). ***For any other language, you need to clear it with the TA before beginning.***

**PLAGIARISM!**

Your program must be your original work, as stated and described in the syllabus. If you are unsure about whether some open source code can be used, contact the TA.