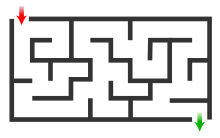
**Assignment Problem 4**

**Shortest Path Finding in Maze using Best First Search:**

|  |  |
| --- | --- |
| **Best First Search Algorithm:**  The name “best-first” refers to the method of exploring the node with the best “score” first. An evaluation function is used to assign a score to each candidate node. The algorithm maintains two lists, one containing a list of candidates yet to explore (OPEN), and one containing a list of visited nodes (CLOSED). Since all unvisited successor nodes of every visited node are included in the OPEN list, the algorithm is not restricted to only exploring successor nodes of the most recently visited node. In other words, the algorithm always chooses the best of all unvisited nodes that have been graphed, rather than being restricted to only a small subset, such as immediate neighbours. Other search strategies, such as depth-first and breadth-first, have this restriction. The advantage of this strategy is that if the algorithm reaches a dead-end node, it will continue to try other nodes  **Algorithm**  The first step is to define the OPEN list with a single node, the starting node. The second step is to check whether or not OPEN is empty. If it is empty, then the algorithm returns failure and exits. The third step is to remove the node with the best score, *n*, from OPEN and place it in CLOSED. The fourth step “expands” the node *n*, where expansion is the identification of successor nodes of *n*. The fifth step then checks each of the successor nodes to see whether or not one of them is the goal node. If any successor is the goal node, the algorithm returns success and the solution, which consists of a path traced backwards from the goal to the start node. Otherwise, the algorithm proceeds to the sixth step. For every successor node, the algorithm applies the evaluation function, *f*, to it, then checks to see if the node has been in either OPEN or CLOSED. If the node has not been in either, it gets added to OPEN. Finally, the seventh step establishes a looping structure by sending the algorithm back to the second step. This loop will only be broken if the algorithm returns success in step five or failure in step two.  The algorithm is represented here in pseudo-code:  1. Define a list, OPEN, consisting solely of a single node, the start node, *s*.  2. IF the list is empty, return failure.  3. Remove from the list the node *n* with the best score (the node where *f* is the minimum), and move it to a list, CLOSED.  4. Expand node *n*.  5. IF any successor to *n* is the goal node, return success and the solution (by tracing the path from the goal node to *s*).  6. FOR each successor node:  a) apply the evaluation function, *f*, to the node.  b) IF the node has not been in either list, add it to OPEN.  7.looping structure by sending the algorithm back to the second step.  **Best-first search** algorithm visits next state based on heuristics function *f(n) = h* with lowest heuristic value (often called greedy). It doesn't consider cost of the path to that particular state. All it cares about is that which next state from the current state has lowest heuristics. |  |
|  |  |

A **maze** is a path or collection of paths, typically from an entrance to a goal. For example, given below there is a maze .



***.*A small maze with one entrance and one exit**

**Imagine that an artificially intelligent robot approaches a valley. Looking down the cliff into the valley, the robots sees a large, man-made, labyrinth structure. The structure has an entrance, an exit, and is surrounded by water. The robot can, of course, simply enter the structure and wander around, until it finally reaches an exit point. However, since the robot is programmed with AI, it can do a lot better.**

**The robot can scan the maze into its memory and perform image processing against it, converting the pixels in the image into a data representation of the maze. With the maze analyzed, an algorithm can be ran against it to determine a solution path through the maze. Once complete, the robot can simply walk through the maze in one try.**

**Develop a program to guide the robot to find shortest path through the given below maze:**

****

**In the above maze # represent wall through the maze. The integer value in the cell represent the heuristic value that helps the robot to decide which path to be followed applying Best First Search. (S) symbol shows start of the entrance through the Maze and (G) represents Goal. Robot have to trace shortest path from start to goal.**

**To implement the above maze you can use an integer matrix[10][10] as follows:**

**Int maze[10][10], start, goal;**

**The value stored in the maze may be taken as heuristic value to guide your search for shortest path using BFS.**

**INPUT: Start state value and Goal state value.**

**OUTPUT- (1). Sequence of cells generated in covering shortest path.**

**(2). Shortest path distance in terms of number of cells traversed in covering that path.**