PROBLEM TYPE 1 - Depth First Search

The implemented DFS gives priority to search depth, right, left,up in decreasing order.

Complete:- Yes, DFS is complete in the given case since it is bounded. For some other problem, where there should be an infinite path, then DFS can not be complete.

Optimality:- No, DFS is not optimal. It will somehow find the path to reach the destination, but it's not optimal.

Data Structure Used: Stack

Time Complexity
For n i.e., (n * n grid)
Displaying maze=$O(n^2)$
Make Screen=$O(n^2)$
Redraw screen=$O(n^2)$
Check position=O(1)
Search Algorithm: $n^2(O(1)+O(n^2)=O(n^4)$

Maximum no of iteration possible=$O(n^2)$
Pop/Push operation on stack=O(1)
Internally calling Redraw Screen in every iteration=$O(n^2)$

So, total complexity= No of iteration *(pop operation+ redraw image internally)
$n^2*(O(1)+O(n^2)=O(n^4)$

PROBLEM TYPE 2 - Breadth First Search

Complete:- Yes, BFS is always complete.

Optimality:-Yes, for constant path cost, BFS is optimal. On traversing back from destination to their respective parent node, we will get the optimal path. But the total searching cost is more as it grows level by level compared to other searches.

Data Structure Used: Queue

Time Complexity
For n i.e.,(n * n grid)
Displaying maze=$O(n^2)$
Make Screen=$O(n^2)$
Redraw screen=$O(n^2)$

Check position=O(1)
Search Algorithm: n^2*(O(1)+O(n^2))=**O(n^4)**

Maximum no of iteration possible=O(n^2)
Dequeue/ enqueue operation=O(1)
Internally calling Redraw Screen in every iteration=**O(n^2)**
So, total complexity= No of iteration * (queue operation+ redraw image internally
n^2*(O(1)+O(n^2))=O(n^4)

PROBLEM TYPE 3 - Varying Cost Search (VCS)

Complete:- Yes, VCS is always complete.

Optimality:-Yes, the final cost is optimal for the optimal path. However, the combined cost for expanding all nodes to find the optimal solution is very expensive.

Data Structure Used: Priority Queue

Time Complexity
For n i.e., (n * n grid)
Displaying maze=O(n^2)
Make Screen=O(n^2)
Redraw screen=O(n^2)
Check position=O(1)
Search Algorithm: n^2*(O(log(n))+O(n^2)=**O(n^4)**

Maximum no of iteration=O(n^2)
Priority Queue enqueue & dequeue= O(log(n))
Internally calling Redraw Screen in every iteration=O(n^2)

So, total complexity= No of iteration *(priority queue operations+redraw image internally)
n^2*(O(log(n)+O(n^2))=O(n^4)

PROBLEM TYPE 4 - A* Search (VCS)

For implementation of A* algorithm for the given problem

Complete:- Yes, A* is always complete.

Optimality:-Yes,the cost is optimal. Its combined search cost for expanding all nodes is also less compared to other searches (in general), but for the given setting of the game, it is algo

expensive as moving toward goal from heuristic is the best option, but at the same time, cost of moving toward goal is high.

Data Structure Used: Priority Queue

Time Complexity
For n i.e.,(n * n grid)
Displaying maze=$O(n^2)$
Make Screen=$O(n^2)$
Redraw screen=$O(n^2)$
Check position=O(1)
Search Algorithm: $n^2*(O(\log(n))+O(n^2)=\mathbf{O(n^4)}$

Maximum no of iteration possible =$O(n^2)$
Priority Queue enqueue & dequeue= O(log(n))
Internally calling Redraw Screen in every iteration=$O(n^2)$

So, total complexity= No of iteration *(priority queue operations+redraw image internally)
$n^2*(O(\log(n)+O(n^2))=O(n^4)$