Artificial Intelligence Lab
Assignment 1

Group Number: 27

Pavan Kumar V Patil 200030041

Karthik J Ponarkar 200010022

December 26, 2021

Contents

1 Problem Statement (Uninformed Search)

Teach Pacman how to intelligently find his food! The objective of this task is to simulate breadth-first search, depth-first search, and DFID in the state space. The state-space consists of an m x n grid. The start state is (0,0). The goal state is the position of (*) in the grid. The Pacman is allowed to move UP, DOWN, LEFT and RIGHT (except for boundary). Compare the above search methods on two accounts:

1. Length of the path (from the initial state to the goal state) that each algorithm finds
2. Number of states explored (visited) during the search.

2 Functions used to implement the solution

1. readFile(): This function helps us to read the input file and store the variables such as run[0(BFS), 1(DFS), 2(DFID)], m(rows) and n(columns).
2. createGraph(): This function creates the graph by joining the necessary edges between the vertices having '(empty space)' as a character.
3. BFS(): This function performs Breadth first search(BFS) on the created graph to reach the goal state.
4. DFS(pair<int, int> start): This function performs Depth first search(DFS) on the created graph to reach the goal state.
5. DFID(): This function performs Depth first iterative deepening(DFID) on the created graph to reach the goal state.
6. DFSrestricted(pair<int,int> start, int depth): This function performs DFS on the created graph till some depth, which is the sub-graph of created graph to reach the goal state.
7. BFSpath(): This function helps to back-track the path, after reaching the goal state, after calling the BFS function.
8. DFSpath(): This function helps to back-track the path, after reaching the goal state, after calling the DFS function.
9. DFIDpath(): This function helps to back-track the path, after reaching the goal state, after calling the DFID function.
10. moveGen(): This function generates all possible states that can be generated from the current position with the help of createGraph() function.
11. goalTest(): This function checks whether the current position is a goal state or not and returns true if it is a final state else returns false.

3 Pseudocode

3.1 MoveGen function

```
function moveGen(pair< int, int > node)
        vector<pair<int, int>> answer ← empty
        foreach (coordinates in node.adjacencyList )
          ans.append ← coordinates
        end foreach
        return ans
    end function
```

Note: The adjacency list used in the moveGen() function for a vertex is created by the createGraph() function.

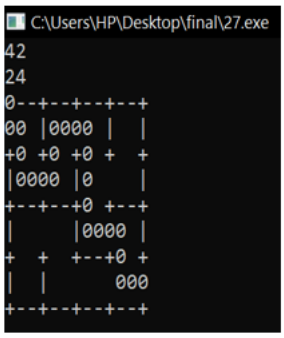
3.2 GoalTest function

```
function goalTest(pair<int, int> node)
    if node.first == goalX and node.second == goalY  then
        return true
    end if
    return false
end function
```

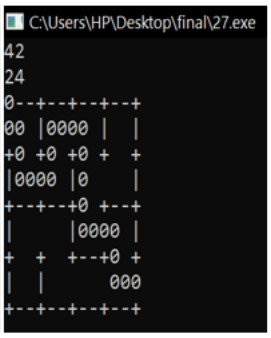
4 Result

Results obtained for different test cases:

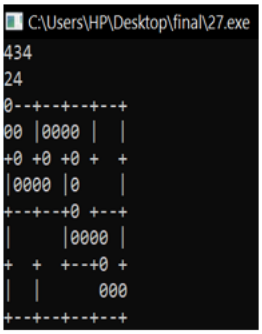
BFS



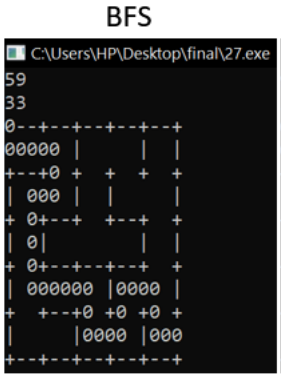
DFS



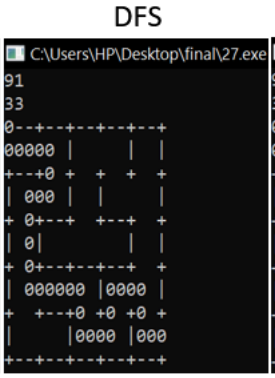
DFID



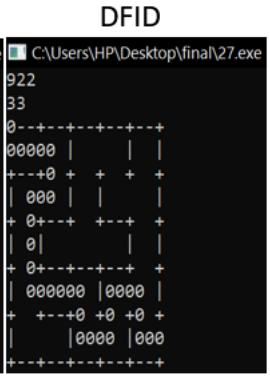
BFS



DFS



DFID



- For all the uninformed searches the path direction and its length is same, but the number of explored states will vary.
- For the above test cases, time taken by DFID is greater than, time taken by BFS and DFS.

5 Conclusion

- From the below table we can understand the dependence of path travelled to reach the goal state on the priority of neighbours added.

		U>D>R>L	U>D>L>R	D>U>L>R
Sample test 1	BFS	42, 24	43, 24	43, 24
Sample test 1	DFS	42, 24	59, 24	59, 24
Sample test 1	DFID	434, 24	436, 24	436, 24
Sample test 2	BFS	61, 33	61, 33	59, 33
Sample test 2	DFS	65, 45	64, 45	88, 33
Sample test 2	DFID	929, 33	926, 33	919, 33

- DFID is the combination of DFS and BFS, to optimise the time and space complexities, to reach the goal state.