CS312, Assignment-1 Lab Report

R.Chidaksh(200010046), R.Sai Krupakar(200010047)

December 26, 2021

1 Objective

The objective of this task is to simulate BFS, DFS and DFID in the state space. The state-space consists of an (m $\times$ 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). A comparison of the path length and the number of states explored between the different search methods and, also between the orders in which neighbours are added, are performed.

2 Results observed

Here are some of the results we observed while implementing the code for three different algorithmic cases(BFS, DFS and DFID).

- Generally the states explored by DFS and BFS are different.
- BFS and DFID yield paths of almost same length.
- While comparing BFS, DFS and DFID the number of nodes explored by DFID is larger compared to other two. Because in DFID the nodes are visited multiple times in a cyclic manner.
- BFS is useful for finding shortest paths while DFS is helpful for finding longest paths.
- BFS always yields paths of shorter (or equal to) lengths than those yielded by DFS.

- It matters how we visit the adjacent nodes in finding the shortest path. We can see the difference in BFS, DFS and DFID.
- In all the three cases we can guarentee that we can find the solution if it exists.
- DFS and DFID consume same memory which is less than memory consumed by BFS, DFS takes memory of the order $O(bd)$ whereas BFS takes memory of $O(b^d)$.

3 MoveGen( ) Pseudo code

Function takes a set states as input and returns a set of states that are reachable from the input state in one step.

Algorithm 1 MoveGen( )

```
procedure MoveGen(row, col, maze,rl,cl )

    if row+1<total rows and maze[row+1][col] is equals to ' ' or '*' then
        add the element in the adjacent matrix
    end if
    if row-1>0 and maze[row-1][col] is equals to ' ' or '*' then
        add the element in the adjacent matrix
    end if
    if column<total columns and maze[row][col + 1] is equals to ' ' or '*'
then
        add the element in the adjacent matrix
    end if
    if column-1>0 and maze[row][col-1] is equals to ' ' or '*' then
        add the element in the adjacent matrix
    end if
```

4 GoalTest( ) Pseudo code

The function checks whether the input element is equal to '*' and returns True if it is equal to '*' otherwise it will return False.

Algorithm 2 GoalTest( )

```
procedure GoalTest(r, c, matrix)

    if matrix[r][c] equals to '*': then
        return True
    return False
```

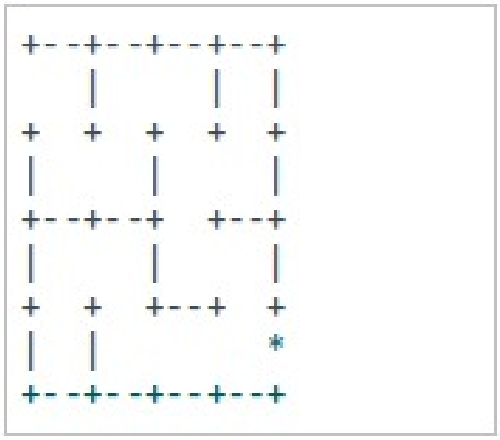
5 Conclusion:

Clearly, these three algorithms generate an accepted output but with different efficiency and optimizations. In DFID, there is an increase in the number of explored paths because of the branching factor , BFS and DFS follow their normal conventional search patterns.

6 SNAPSHOTS OF OUTPUT:

Sample Input 1 (BFS):

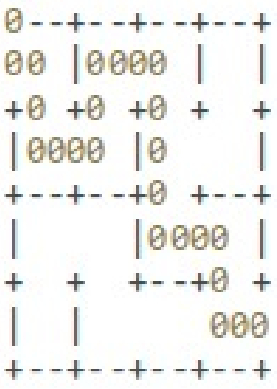
0



Sample Output 1:

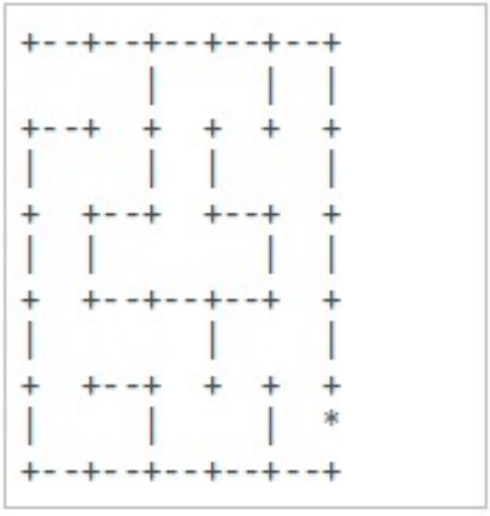
42

24



Sample Input 4 (BFS):

0



Sample Output 4:

59

33