Maze Report

Data Structure Used in Maze :

1) Arrays .

2) Single LinkedList .

3) Stack .

4) Queue .

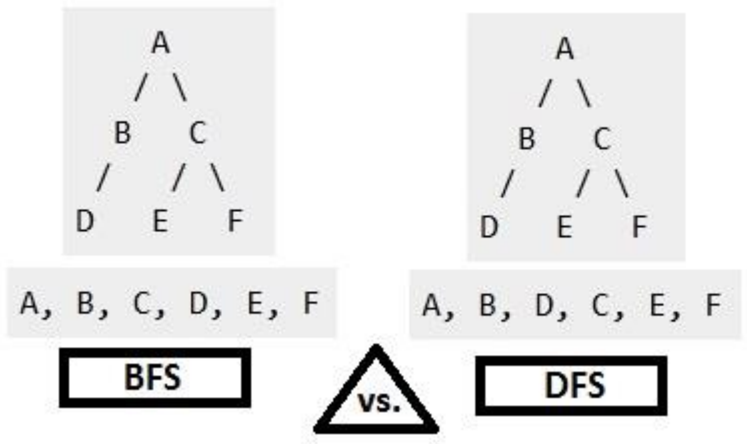
Algorithms used in Maze :

1) Depth First Search .

2) Bredth First Search .

Comparison between the two algorithms :

Example showing the difference between BFS & DFS algorithms :



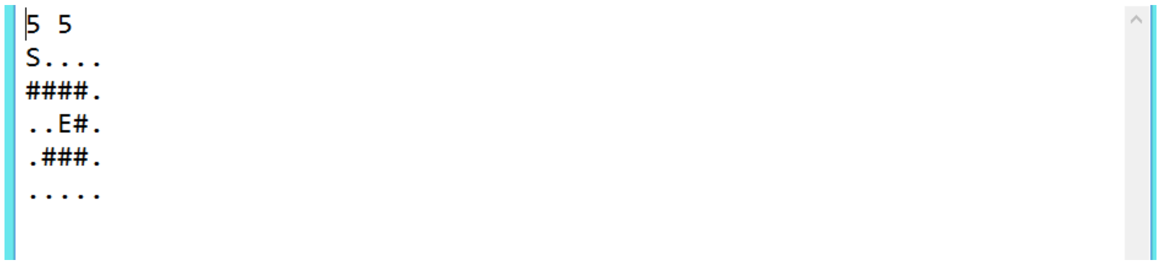
DFS	BFS
• DFS starts the traversal from the root node and explore the search as far as possible from the root node i.e. depth wise. • In the example it starts with any node in second level B or C • assume it chooses B • it is still moving until the end point if found return • Else return to the second root which is C • And do the same work until found the end point .	• BFS starts traversal from the root node and then explore the search in the level by level manner , as close as possible from the root node. • In the example it starts with the first level which contains B,C • If found return • Else visit the second level which contains D,E,F • If found return back • End
• Using stack to implement	• Using queue to implement
• I think it 's faster	• slower
• Using less memory	• Using more memory

How to work :

- Input the file which contains the dimensions and the map .
- Choosing how to get the solution whether DFS or BFS .
- Reading the file and detect the dimensions and the map .
- Run the DFS algorithm on the map By Visting the Right , Left , Up then Down .
- Store the solution in Point form in Single LinkedList .
- Put the data from the LinkedList to the 2D array .
- Get the Solution and Printing it for the user .

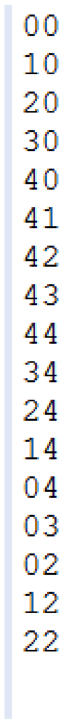
Sample Runs :

(1) Test snake case :



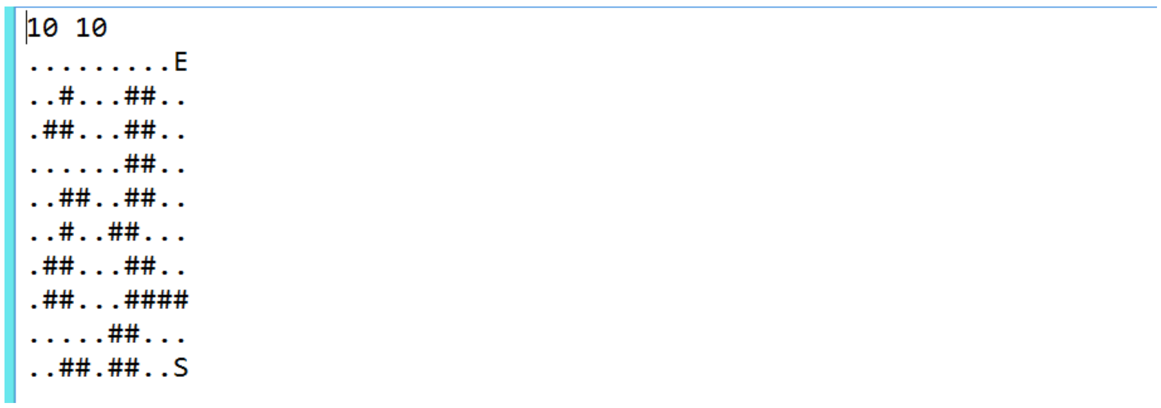
```
5 5
S....
####.
..E#.
.###.
.....
```

Solution :



```
00
10
20
30
40
41
42
43
44
34
24
14
04
03
02
12
22
```

(2) Test No path :

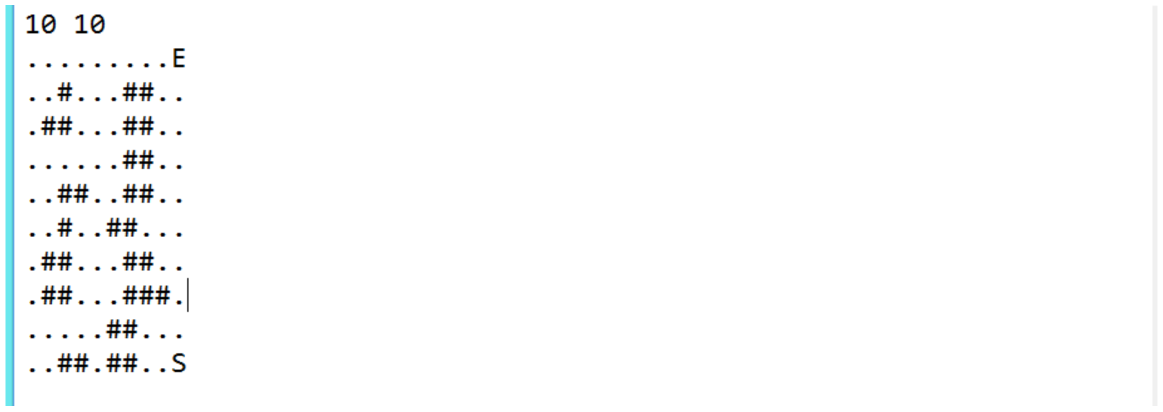


```
10 10
.........E
..#...##..
.##...##..
......##..
..##..##..
..#..##...
.##...##..
.##...####
.....##...
..##.##..S
```

Solution :

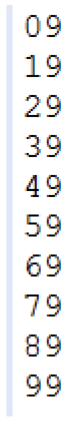
```
null
```

(3) Test Big map with line path :



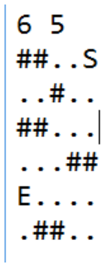
```
10 10
.........E
..#...##..
.##...##..
......##..
..##..##..
..#..##...
.##...##..
.##...###.
.....##...
..##.##..S
```

Solution :



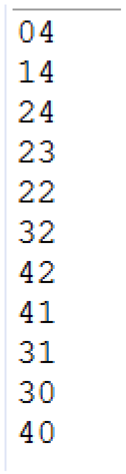
```
09
19
29
39
49
59
69
79
89
99
```

(4) Test different Dimensions :



```
6 5
##..S
..#..
##...
...##
E....
.##..
```

Solution :



```
04
14
24
23
22
32
42
41
31
30
40
```