12th January 2020

Lab 1 Report

K. Sai Anuroop (170030035) and Mehul Bose (170030010)

Pseudo-code for BFS

```
final_BFS ( )
{
    // check if source and goal cells are valid
    // push source to the queue
    while queue is not empty
    {
        // increment states explored
        // extract head cell from the queue
        // check if current cell is goal by matching coordinates of current cell with that of
           goal cell      —> goaltest( )
            // if yes, find the path traced from source to goal and return distance
        // pop the extracted cell from the queue
        for all neighbours of the current cell in the given order        —> movegen( )
        {
            // if the neighbour is valid and is not yet visited
                // push the cell into queue and note its distance from the source
        }
    }
}
```

Pseudo-code for DFS

```
final_DFS ( )
{
    // check if source and goal cells are valid
    // push source to the stack
    while stack is not empty
    {
        // increment states explored
        // extract head cell from the stack
        // check if current cell is goal by matching coordinates of current cell with that of
           goal cell      —> goaltest( )
            // if yes, find the path traced from source to goal and return distance
        // pop the extracted cell from the stack
        for all neighbours of the current cell in the given order        —> movegen( )
        {
            // if the neighbour is valid and is not yet visited
                // push the cell into queue and note its distance from the source
        }
    }
}
```

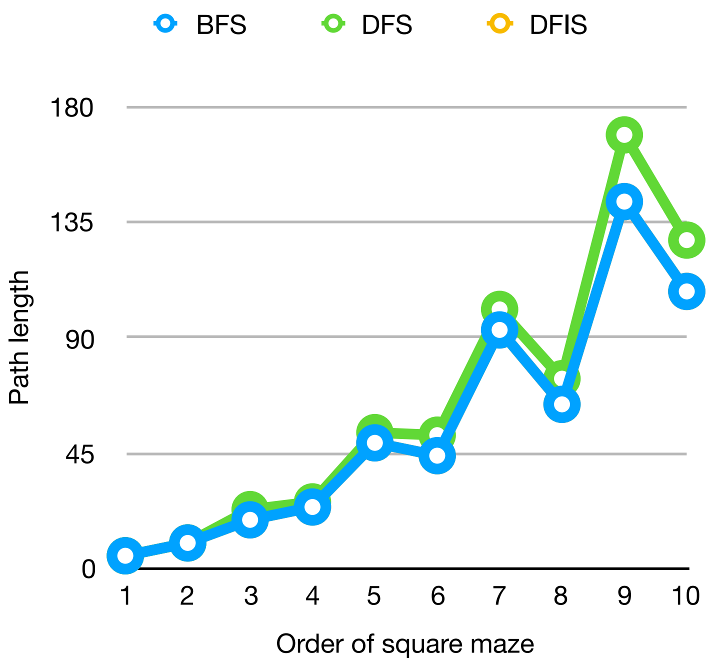
Pseudo-code for DFIS

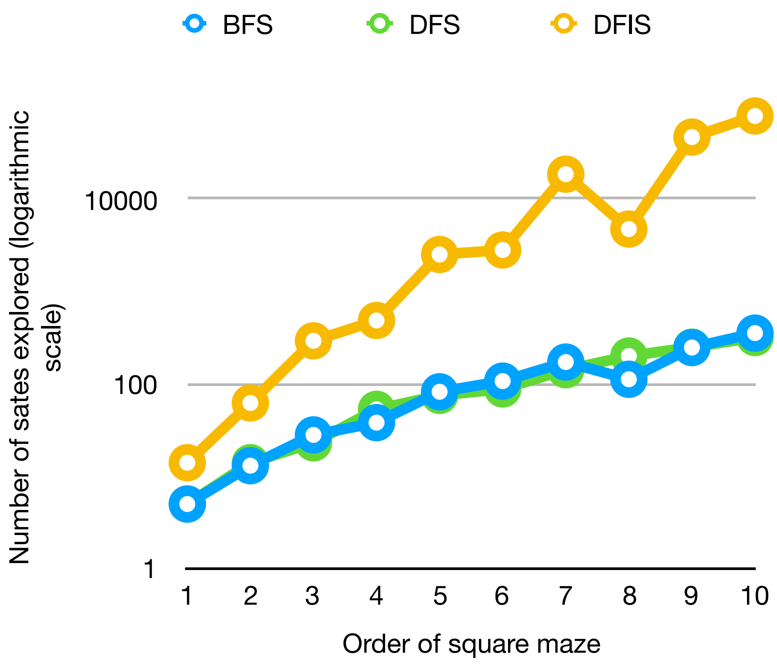
```
final_DFIS ( )
{
    // check if source and goal cells are valid
    // if depth limit is reached, i.e., current depth is 0, return -1
    // note the distance of the current cell from the source
    // check if current cell is goal by matching coordinates of current cell with that of
       goal cell      —> goaltest( )
        // if yes, find the path traced from source to goal and return distance
    // increment states explored
    for all neighbours of the current cell in the given order      —> movegen( )
    {
        // if the neighbour is valid and is not yet visited, or is visited but its current distance
           from source is less than that of its distance from the source in the previous visits
            // call final_DFIS ( ) with depth as current depth-1
    }
}

// depth = 1
while true
{
    // call final_DFIS ( ) with depth
    // if goal is reached
        // break
    // increment depth
}
```

We tested our code with the give test cases and confirmed it to be working correctly. We then tested the code with square mazes of orders 1 through 10 generated using the link given to us in the assignment statement, noted the values of path length and the number of states explored for each of them at three different orders of visiting adjacent cells viz Down>Up>Right>Left, Down>Up>Left>Right, and Right>Down>Up>Left. An analysis of the results obtained is furnished below.

Analysis (Down>Up>Right>Left)

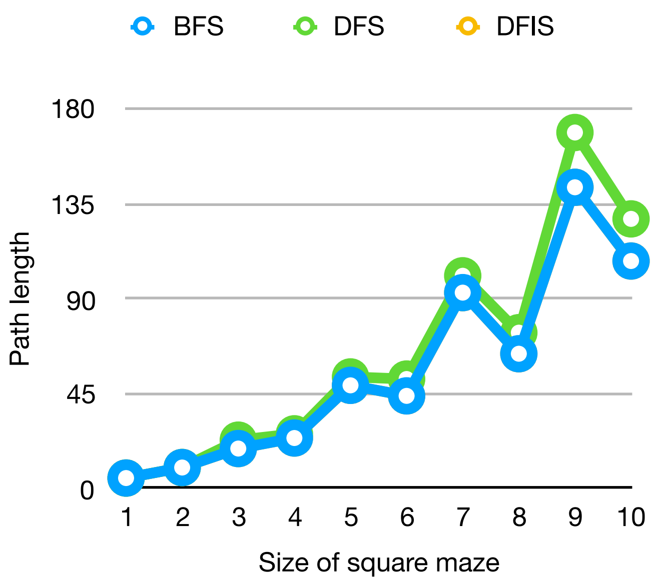
	Path length			**Number of states explored**		
Order	**BFS**	**DFS**	**DFIS**	**BFS**	**DFS**	**DFIS**
1	5	5	5	5	5	14
2	10	10	10	13	14	62
3	19	23	19	28	23	292
4	24	26	24	38	53	485
5	49	53	49	82	75	2511
6	44	52	44	107	88	2810
7	93	101	93	172	142	18585
8	64	74	64	112	200	4733
9	143	169	143	247	249	46820
10	108	128	108	354	317	79273

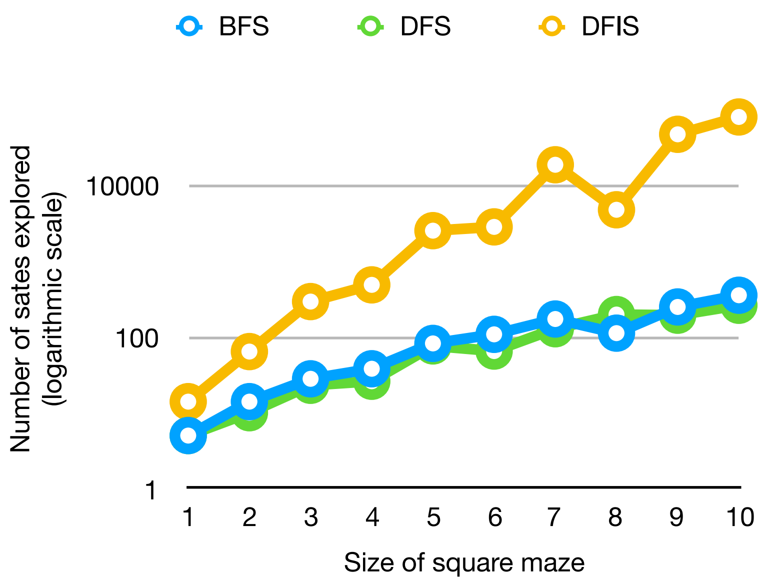




Analysis (Down>Up>Left>Right)

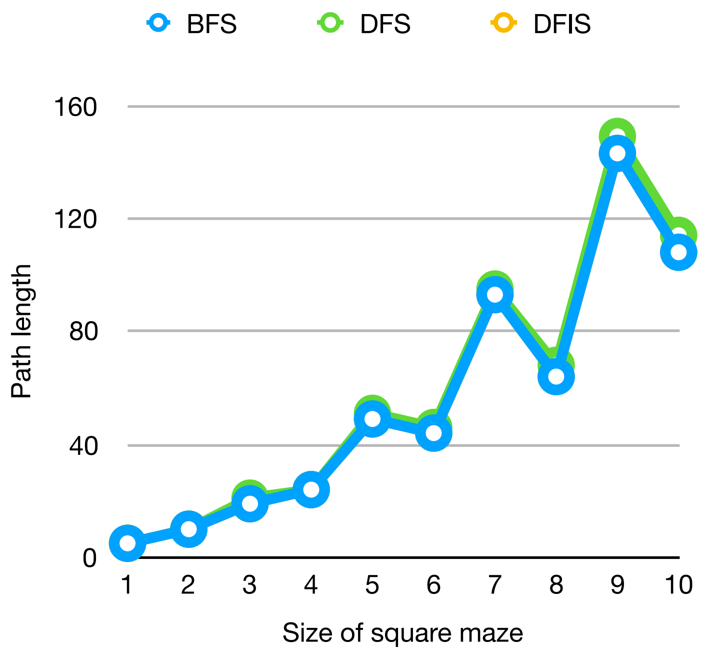
	Path length			Number of states explored		
Order	**BFS**	**DFS**	**DFIS**	**BFS**	**DFS**	**DFIS**
1	5	5	5	5	5	14
2	10	10	10	14	10	64
3	19	23	19	28	23	292
4	24	26	24	38	26	485
5	49	53	49	82	76	2513
6	44	52	44	108	65	2819
7	93	101	93	172	128	18579
8	64	74	64	112	197	4739
9	143	169	143	248	193	46834
10	108	128	108	354	261	79276

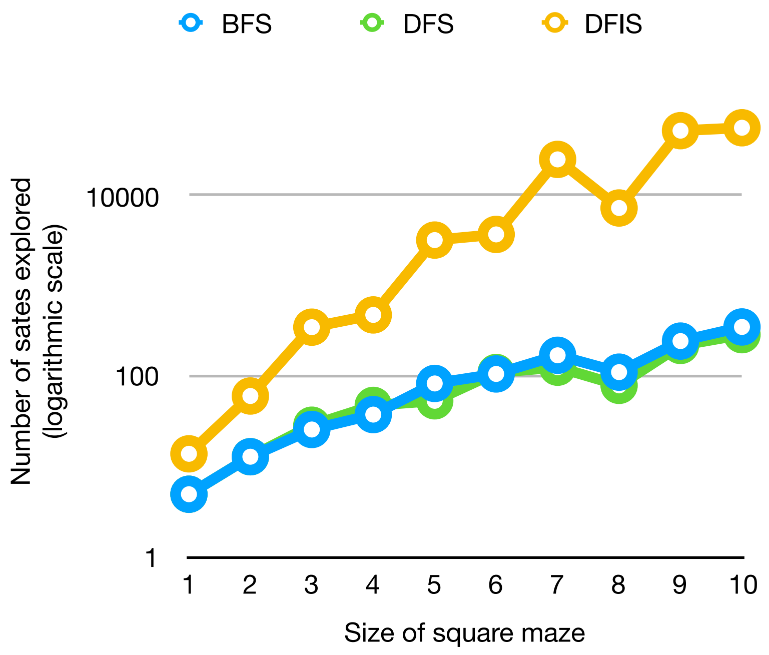




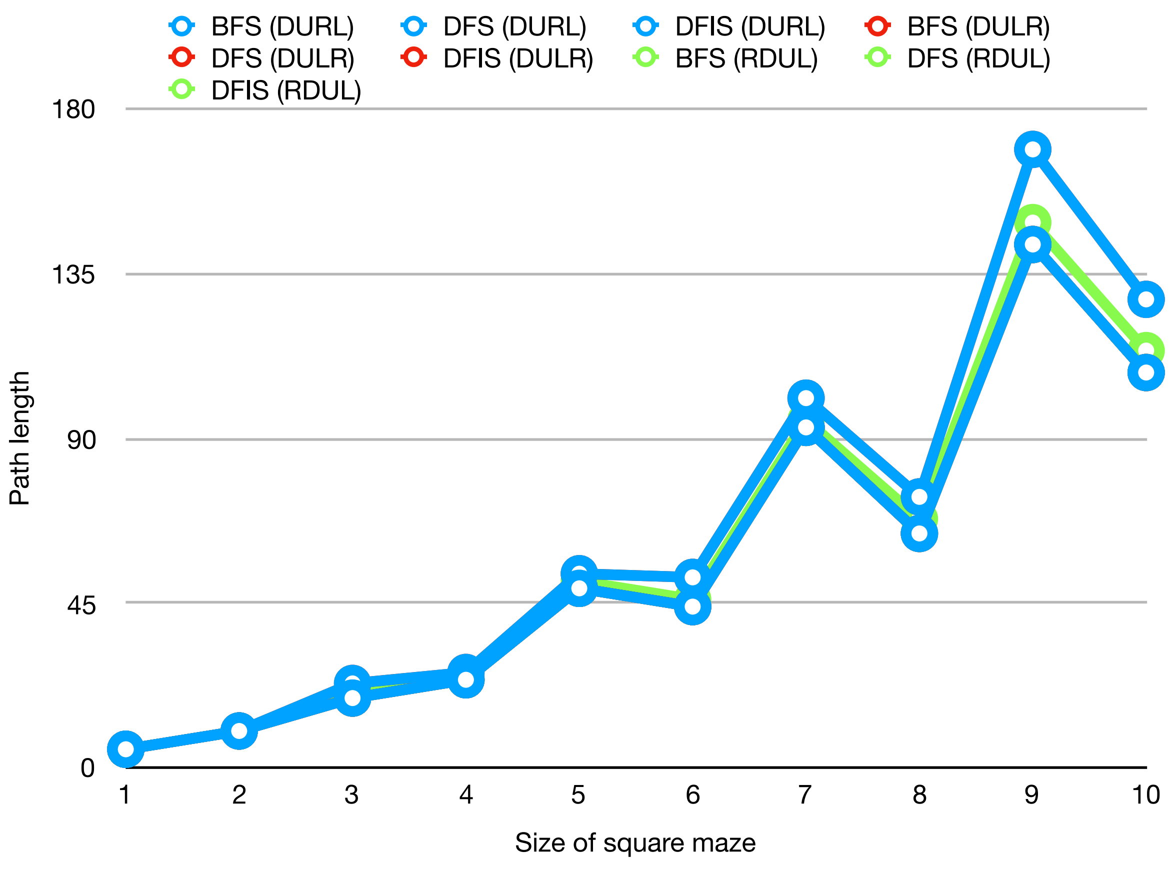
Analysis (Right>Down>Up>Left)

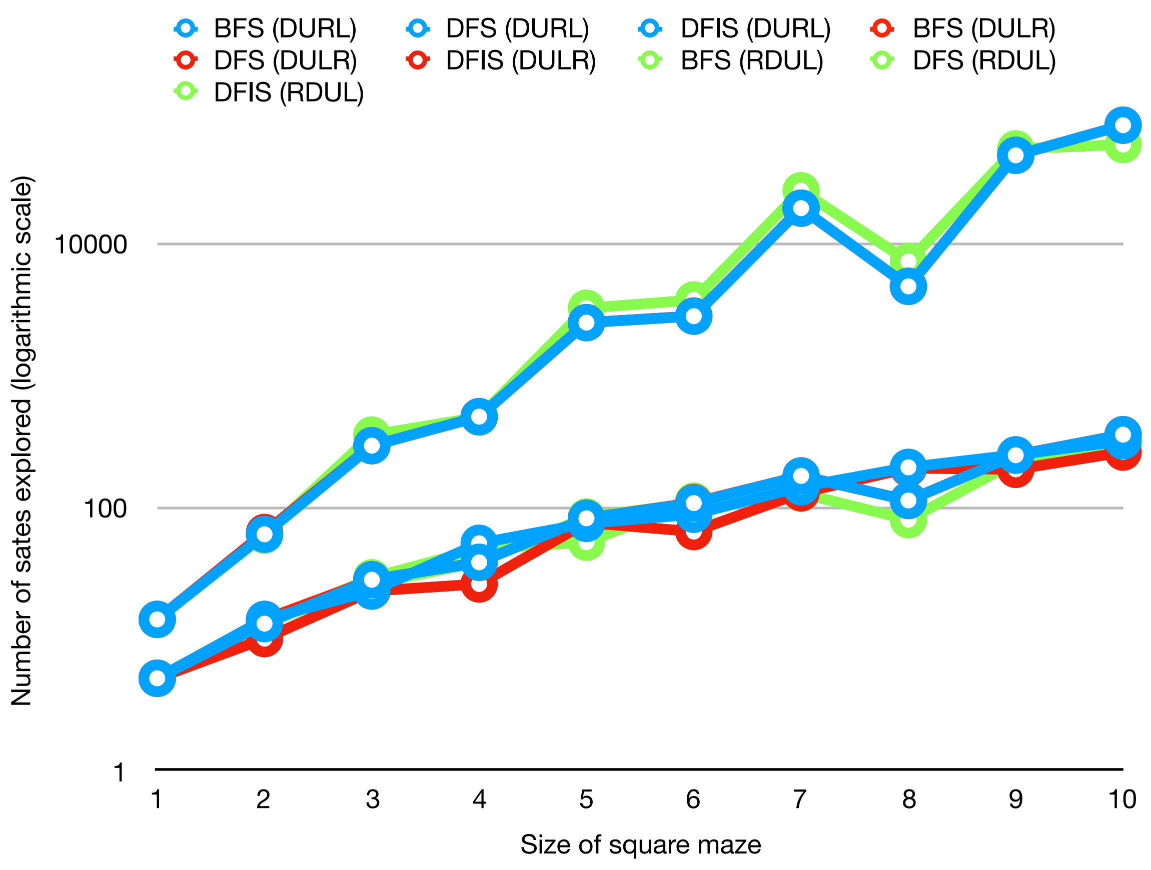
	Path length			**Number of states explored**		
Order	**BFS**	**DFS**	**DFIS**	**BFS**	**DFS**	**DFIS**
1	5	5	5	5	5	14
2	10	10	10	13	13	61
3	19	21	19	26	29	352
4	24	24	24	38	48	481
5	49	51	49	84	54	3234
6	44	46	44	107	111	3736
7	93	95	93	172	127	25242
8	64	68	64	112	80	7322
9	143	149	143	247	221	52453
10	108	114	108	354	291	56486





Cumulative Analysis





Inference

1. BFS always yields paths of shorter (or equal to) lengths than those yielded by DFS
2. BFS and DFIS yield paths of same length
3. Number of states explored by BFS and DFS are generally different but do not vary appreciably among themselves
4. Number of states explored by DFIS is quite large when compared to those explored by BFS and DFS due to the fact that finding optimal path in DFIS with cycles in the graph may require visiting a node multiple times
5. Order of visiting adjacent cells does influence the result obtained, as is evident from the plots of cumulative analysis seen above, though the difference in the path lengths obtained and number of states explored in each case is not appreciably large

---

END OF REPORT