N-Queens Problem

Assignment 2

خالد عبدالعزيز الدويسان	
علي بن دهيم	
عبدالعزيز العريفي	
عبدالرحمن الميمان	

Description :

The aim of this problem is to place N queens on a chessboard of size NxN in an order where no queen may attack another. A queen can attack other queens either diagonally, or in same row or column.

State Representation :

1. ***State:*** NxN grid with N-queens with cost and heuristic function if exist.
2. ***Initial state:*** queens are randomly distributed across the board.
3. ***Actions:*** The user can use any search algorithm to reach Goal state.
4. ***Goal State:*** N-queens are distributed where no queen can attack another.

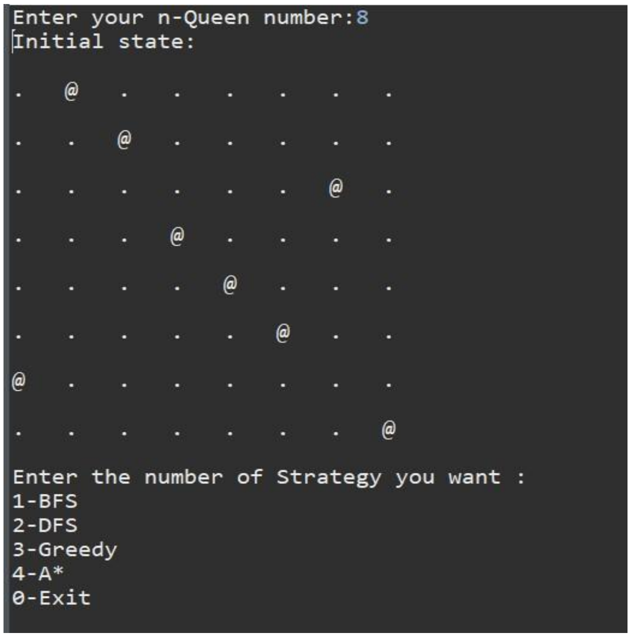
Description of the Solution implementation :

In this implementation we used 2-Dimensional array With N-queens are randomly generated, and solved with 4 Search algorithms BFS,DFS,Greedy,A* -3 data structures are used:

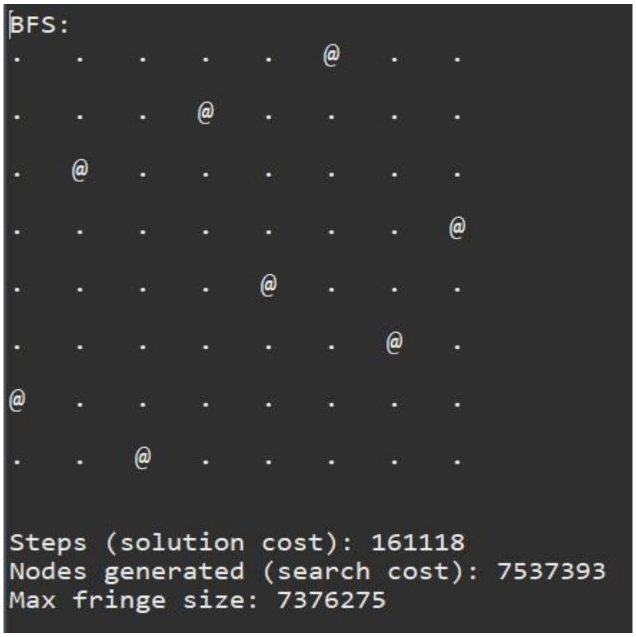
	BFS	*DFS*	*Greedy*	*A**
Data Structure	Queue	Stack	Priority queue	Priority queue

Sample Run :

Initial State BFS

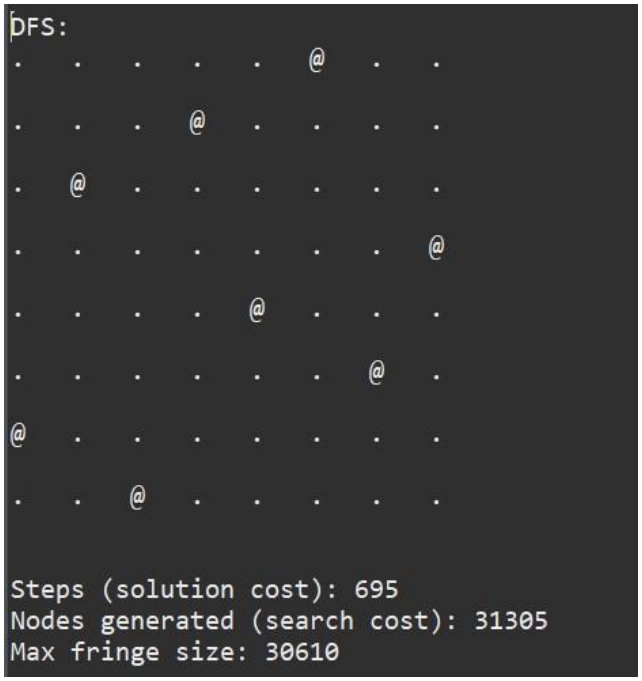


```
Enter your n-Queen number:8
Initial state:

.   @   .   .   .   .   .   .

.   .   @   .   .   .   .   .

.   .   .   .   .   .   @   .

.   .   .   @   .   .   .   .

.   .   .   .   @   .   .   .

.   .   .   .   .   @   .   .

@   .   .   .   .   .   .   .

.   .   .   .   .   .   .   @

Enter the number of Strategy you want :
1-BFS
2-DFS
3-Greedy
4-A*
0-Exit
```



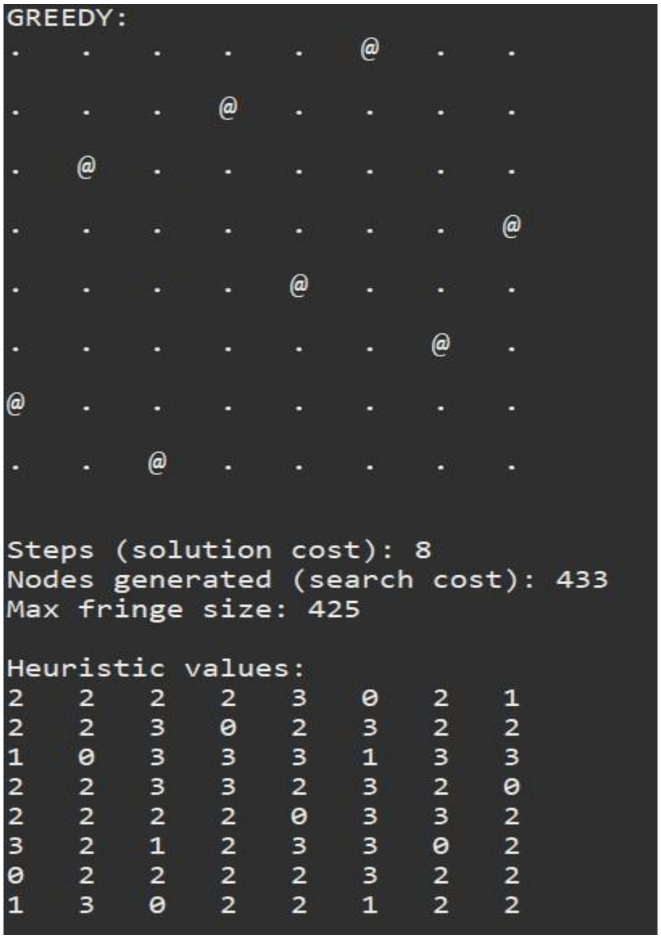
```
BFS:
.   .   .   .   .   @   .   .

.   .   .   @   .   .   .   .

.   @   .   .   .   .   .   .

.   .   .   .   .   .   .   @

.   .   .   .   @   .   .   .

.   .   .   .   .   .   @   .

@   .   .   .   .   .   .   .

.   .   @   .   .   .   .   .


Steps (solution cost): 161118
Nodes generated (search cost): 7537393
Max fringe size: 7376275
```

DFS

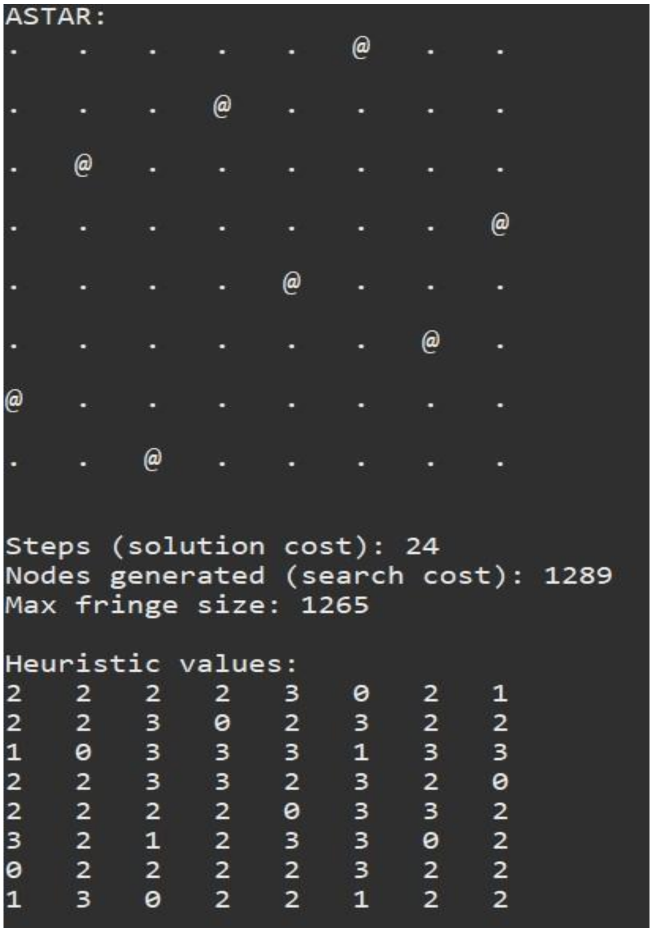


```
DFS:
.   .   .   .   .   @   .   .

.   .   .   @   .   .   .   .

.   @   .   .   .   .   .   .

.   .   .   .   .   .   .   @

.   .   .   .   @   .   .   .

.   .   .   .   .   .   @   .

@   .   .   .   .   .   .   .

.   .   @   .   .   .   .   .


Steps (solution cost): 695
Nodes generated (search cost): 31305
Max fringe size: 30610
```

Greedy   A*



```
GREEDY:
.   .   .   .   .   @   .   .

.   .   .   @   .   .   .   .

.   @   .   .   .   .   .   .

.   .   .   .   .   .   .   @

.   .   .   .   @   .   .   .

.   .   .   .   .   .   @   .

@   .   .   .   .   .   .   .

.   .   @   .   .   .   .   .


Steps (solution cost): 8
Nodes generated (search cost): 433
Max fringe size: 425

Heuristic values:
2   2   2   2   3   0   2   1
2   2   3   0   2   3   2   2
1   0   3   3   3   1   3   3
2   2   3   3   2   3   2   0
2   2   2   2   0   3   3   2
3   2   1   2   3   3   0   2
0   2   2   2   2   3   2   2
1   3   0   2   2   1   2   2
```



```
ASTAR:
.   .   .   .   .   @   .   .

.   .   .   @   .   .   .   .

.   @   .   .   .   .   .   .

.   .   .   .   .   .   .   @

.   .   .   .   @   .   .   .

.   .   .   .   .   .   @   .

@   .   .   .   .   .   .   .

.   .   @   .   .   .   .   .


Steps (solution cost): 24
Nodes generated (search cost): 1289
Max fringe size: 1265

Heuristic values:
2   2   2   2   3   0   2   1
2   2   3   0   2   3   2   2
1   0   3   3   3   1   3   3
2   2   3   3   2   3   2   0
2   2   2   2   0   3   3   2
3   2   1   2   3   3   0   2
0   2   2   2   2   3   2   2
1   3   0   2   2   1   2   2
```

Discussion of the result :

The results are clearly better when using GREEDY search path costs often don’t exceed 10, and worst algorithm is BFS solution cost is the highest also it takes a lot of memory.