AI Assignment -1

Q1:

Algorithm	Computing Time	Memory Required	Total Cost Taken
Depth First Search	1.0	13	72
Breadth-First Search	1.78125	5	90
Varying Cost Search	2.578125	7	45
A* algorithm	2.015625	7	45

DFS is not optimal and not complete if the depth is infinite
A* is complete and optimal
BFS is complete and optimal
VCS is complete and optimal

Observations
1. From the above table, we can notice that the cost needed to reach the final goal is minimum for Uniform Cost Search and A* Algorithm. It happens because of the optimality property of these searches. We can also notice that the cost needed to reach the goal for BFS and DFS is also the same, it happens because the path to reach the goal state is only one and hence DFS always finds the most optimal path. The difference between the cost of A* algorithm and BFS is observed because the cost moving downwards for BFS in 2 whereas for A* algorithm it's 1.
2. We observe that mostly the number of nodes expanded for the A* algorithm and the DFS
algorithm is the least whereas for the BFS algorithm we need to expand most of the nodes because of the fact BFS works level-wise because of which it has to expand almost all the nodes before reaching the goal. For the bigger graphs, A* Algorithm has to expand almost all nodes because the goal is at the bottom right corner and start position is at top left, so in every direction, the heuristic decreases and does not provide any benefit to the search.
3. For the small graphs, A* algorithm expands less nodes but takes time equivalent to the DFS because of the fact A* algorithm also have to calculate the heuristic function because of which it generally takes a little more time than the UFS, BFS, and DFS.
4. The memory taken for storing the nodes in the frontier is maximum for A* algorithm and
BFS algorithm. This happened because the Heuristic function for A* algorithm was underestimating the goal by a huge margin for every node and hence we explored most

of the nodes. Next, for BFS it is large because it works in a level-wise manner and it has to store the nodes of at least 2 levels

Q2

Please find the statistics of the test cases ran :

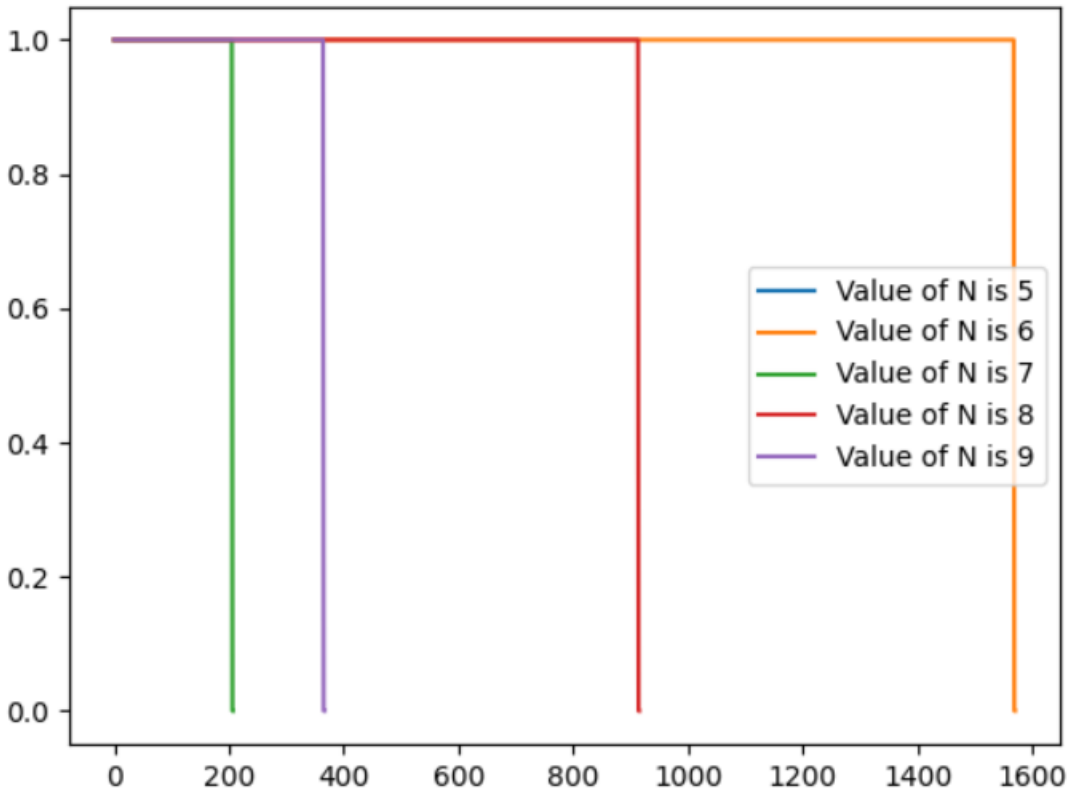
```
G:\F\AI\Assignment-1\GA>python GA.py
Enter the value of N -8
Enter initial population size -100
Solution : [4, 6, 1, 5, 2, 8, 3, 7]
Number of Queens:  8
Iterations :473
Time Taken =  1.234375
CrossOver Probability:  0.6
Mutation Probability:  0.01
```

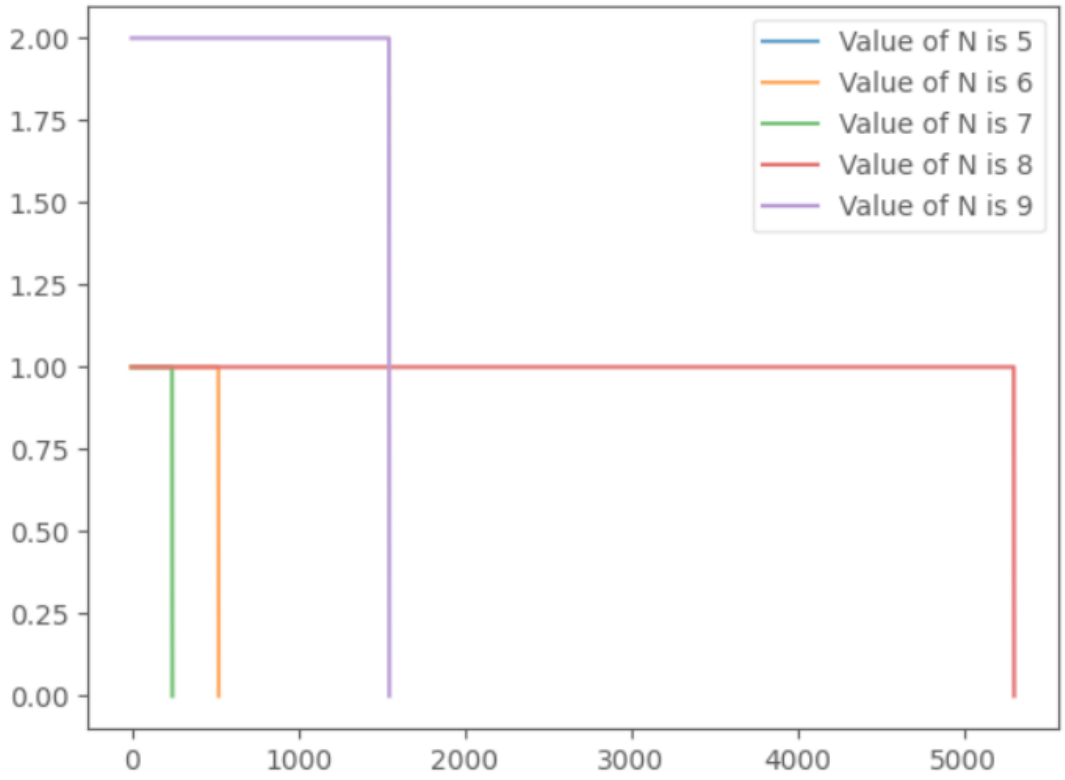
```
G:\F\AI\Assignment-1\GA>python GA.py
Enter the value of N -6
Enter initial population size -100
Solution : [2, 4, 6, 1, 3, 5]
Number of Queens:  6
Iterations :2180
Time Taken =  3.828125
CrossOver Probability:  0.4
Mutation Probability:  0.2
```

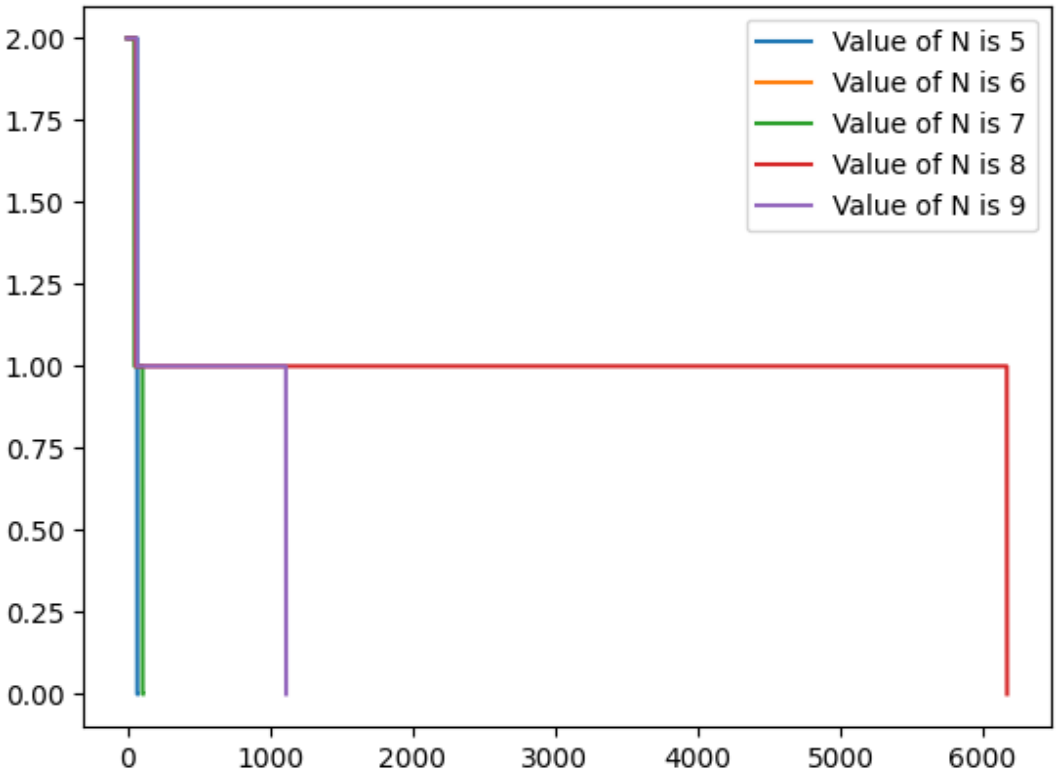
```
G:\F\AI\Assignment-1\GA>python GA.py
Enter the value of N -7
Enter initial population size -50
Solution : [7, 2, 4, 6, 1, 3, 5]
Number of Queens:  7
Iterations :431
Time Taken =  0.546875
CrossOver Probability:  0.7
Mutation Probability:  0.01
```

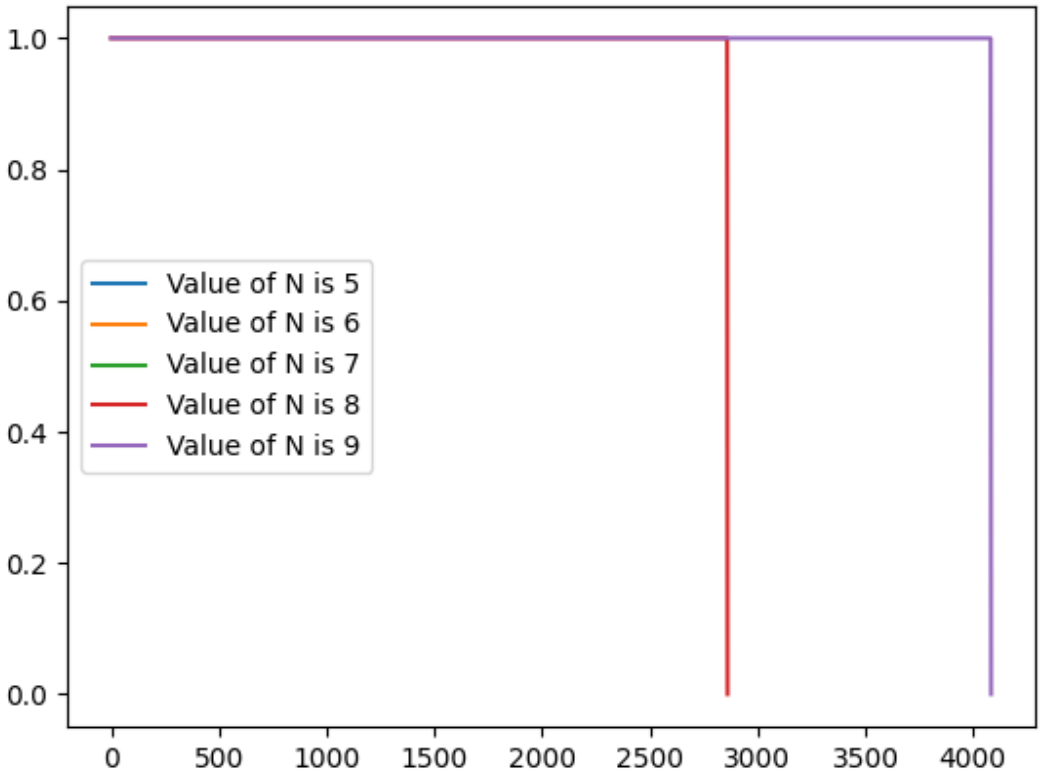
```
G:\F\AI\Assignment-1\GA>python GA.py
Enter the value of N -5
Enter initial population size -10
Solution : [5, 3, 1, 4, 2]
Number of Queens:  5
Iterations :0
Time Taken =  0.0
CrossOver Probability:  0.6
Mutation Probability:  0.02
```

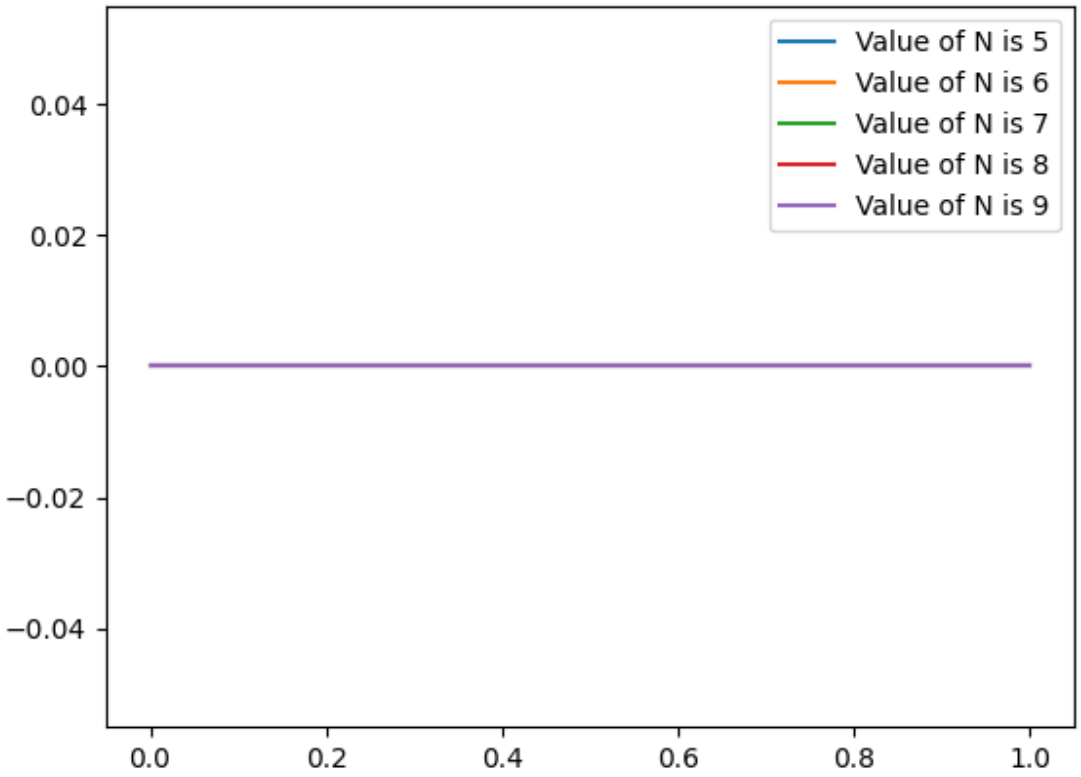
```
G:\F\AI\Assignment-1\GA>python GA.py
Enter the value of N -7
Enter initial population size -1000
Solution : [7, 2, 4, 6, 1, 3, 5]
Number of Queens:  7
Iterations :0
Time Taken =  0.0
CrossOver Probability:  0.6
Mutation Probability:  0.01
```


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
1600
Value of N is 5
Value of N is 6
Value of N is 7
Value of N is 8
Value of N is 9


Value of N is 5
Value of N is 6
Value of N is 7
Value of N is 8
Value of N is 9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1000
2000
3000
4000
5000


Value of N is 5
Value of N is 6
Value of N is 7
Value of N is 8
Value of N is 9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1000
2000
3000
4000
5000
6000


1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000
3500
4000
Value of N is 5
Value of N is 6
Value of N is 7
Value of N is 8
Value of N is 9



Q3 (c)

	Backtracking with CP	**Min Conflict**
Time Complexity	O(9^(n*n))	-
Space Complexity	O((n*n))	-

	Backtracking with CP			Min Conflict		
	Total Clock Time(sec)	**Total Search Time(sec)**	**Number of Nodes Generated**	**Total Clock Time(sec**	**Total Search Time(sec)**	**Number of Nodes Generated**
Test Case 1	0.146	0.138	1148	3156.06	3156.059	98923480
Test Case 2	0.0060	0.0084	71	57.171875	56.817	622686
Test Case 3	0.021	0.021	92	3.78125	3.1985	981908
Test Case 4	0.0169	0.0134	97	22.328125	19.1728	764942
Test Case 5	0.0125	0.0099	80	7.828125	7.268	71208

(d)
Test Case 1: None found
Test Case 2: POWDERING
Test Case 3: None found
Test Case 4: None found
Test Case 5: None found