N-Queens Problem

Approaches used:

1) A-Star
2) Hill Climbing with ‘5 x number of queens’ sideways moves and random restarts: This returns the best result found after running over a period of 10 secs

Extra Credit: Heuristic H3 better than H1 and also admissible

Heuristic 3 is defined in the following way. To remove the attack on a selected queen, we need to either move the selected queen by one step or move all other queens attacking the selected queen by one step. The move cost is calculated for both of these scenarios and the least among these is assigned to the selected queen. So, this value would be the minimum move cost required to remove all the attacks on the selected queen. This is done for all the queens on the board and assigned to each respectively. The maximum value among the values assigned to the queens is set as the heuristic for the board.

This is admissible because the heuristic value is the minimum cost required to remove all the attacks from only one queen on the board. Hence, this cannot be an overestimate of the actual move cost to reach the solution.

This is better than H1, as this provides a better estimate of move cost, which is just the weight^2 of least attacking queen on the board.

The following example shows the Heuristic calculation for the same board used in Q5.
(Case 1: Moving the selected queen, Case 2: Moving other queens)

			8	
5				
		2		
	1			3

Queen Col. No	Under Attack?	Case 1 cost	Case 2 cost	Min Cost
1	No	-	-	-
2	Yes	$1^2 = 1$	$2^2 + 3^2 = 13$	1
3	Yes	$2^2 = 4$	$1^2 = 1$	1
4	No	-	-	-
5	Yes	$3^2 = 9$	$1^2 = 1$	1
Heuristic Value = Maximum(Min Cost) = 1				

Another example

9			2	
		9		1
	5			

Queen Col. No	Under Attack?	Case 1 cost	Case 2 cost	Min Cost
1	Yes	81	81 + 4 = 85	81
2	Yes	25	81	25
3	Yes	81	25 + 1 = 26	81
4	Yes	4	81	4
5	Yes	1	81	1
Heuristic Value = Maximum(Min Cost) = 81				

Question 1: How large of a puzzle can your program typically solve within 10 seconds using A and hill-climbing?*

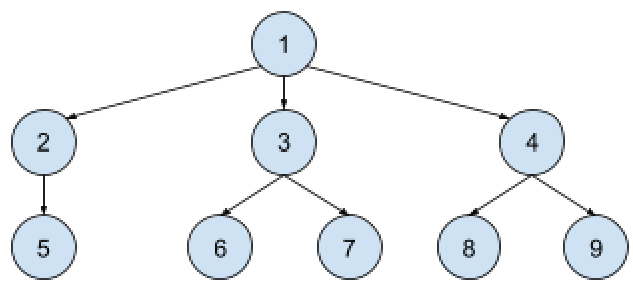
For each board size, 10 random start positions (with random weights) were generated and A*, Hill Climbing with Heuristic 1, 2 & 3 were tested. The following table summarizes the probability of solving within 10 secs based on the 10 trials.

Probability of solving within 10 secs						
Board Size	**A***			**Hill Climbing**		
	H1	**H2**	**H3**	**H1**	**H2**	**H3**
5x5	1	1	1	1	1	1
6x6	0.7	1	1	0.5	0.8	1
7x7	0.3	0.8	0.4	1	1	1
8x8	-	0.2	0.1	0.6	0.8	1
10x10	-	-	-	-	1	0.3
15x15	-	-	-	-	1	1

Question 2: What is the effective branching factor for A and hill-climbing?*

For calculating the branching factor we have used the average number of children per parent node. The following example shows an illustration.

Parent	Children	#Children
1	2,3,4	3
2	5	1
3	6,7	2
4	8,9	2
Eff. Branching Factor = 2.0		



Following table shows the effective branching factor for A* and Hill Climbing for two different board sizes

Effective Branching Factor				
Board Size	**A***			**Hill-Climbing**
	H1	**H2**	**H3**	**H1, H2 & H3**
5x5	9.75	13.85	12.08	20
7x7	19.10	22.72	21.56	42

Question 3: Which approach comes up with cheaper solution paths? Why?

A* comes up with cheaper solution paths.

A* explores based on expanding the node with minimum the ‘heuristic cost + move cost’ from all the open nodes. This does an exhaustive search and because it has move cost in its minimization function it leads to the cheapest solution.

Hill-climbing algorithms are less deliberative; rather than considering all open nodes, they expand the most promising child of the most recently expanded node, based on the heuristic cost, until they encounter a solution. As this algorithm doesn't take into account the move cost, this doesn't optimize the move cost and leads to higher than optimal cost solutions.

The following table shows the average cost for the 10 trials performed in the 5x5 board for the dataset used in Q1. Other boards couldn't be compared as at least 1 trial in each of those boards didn't give a solution.

Average Cost across 10 trials for a 5x5 board					
A*			**Hill Climbing**		
H1	**H2**	**H3**	**H1**	**H2**	**H3**
109	109	109	109	186.3	146.9

Question 4: Which approach typically takes less time? Why?

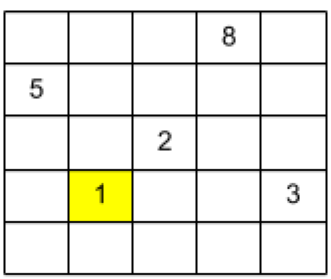
Hill climbing typically takes less time.

As explained in Question 3, A* does an exhaustive search and hence takes more time to find a solution. Whereas hill-climbing only does a local search which makes it faster.

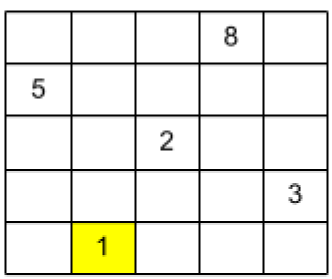
As shown in Question 1 results, we can see that given a time limit of 10 secs, hill climbing is able to solve 10x10 and 15x15 boards whereas A* fails to find a solution in the given time.

Question 5: Either prove the heuristic H2 for A is admissible or not*

Heuristic 2 is not admissible. The counterexample below shows the same. The board on the left has two attacking pairs i.e. columns 2,3 & 2,5. The values on the board are the weight of the queens. Summing up the weight[2] of the least weighted queen among each pair we get a heuristic value of 2. But with just one move of the queen in column 2 as shown in the board on the right, we arrive at the solution. As the heuristic has overestimated the move cost, this heuristic is not admissible.



Heuristic = 2



Move Cost = 1

Reason for choosing '5 x Number of Queens' sideways moves for Hill Climbing

5 random starting boards (6x6) were chosen and 3 different sideways limits were selected i.e. $5 \times n$, $20 \times n$, $50 \times n$ and run for H1, H2, H3. In the next set of runs $1 \times n$, $5 \times n$ and $20 \times n$ were tested.

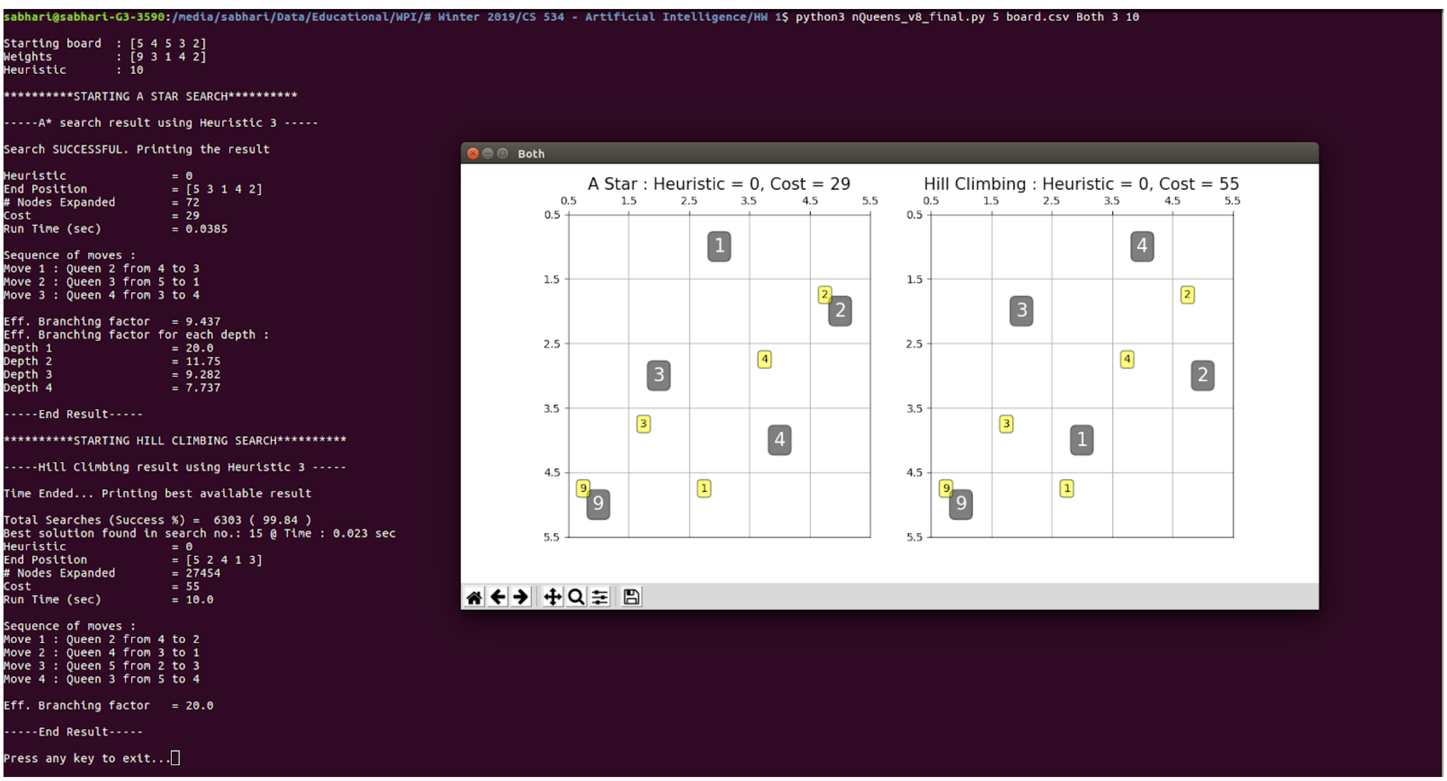
Following table summarizes the following parameters:

1) Average Restarts: Number of restarts in the given 10-sec duration.
2) Average Success %: Percentage of restarts that resulted in a successful solution.
3) Average Cost: Average cost for the solution found in the 5 trials.
4) n: Number of Queens

		S1 = 5 × n, S2 = 20 × n, S3 = 50 × n				S1 = 1 × n, S2 = 5 × n, S3 = 20 × n		
Heuristic Number	**Sideways Mode**	**Avg Restarts**	**Avg Suc %**	**Avg Cost**		**Avg Restarts**	**Avg Suc %**	**Avg Cost**
H1	**S1**	783	5.08	264.6		3433	0.67	259.8
	S2	216	14.84	373.2		793	4.11	286.8
	S3	93	36.13	681.8		210	16.49	382.0
H2	**S1**	2715	86.33	232.4		3185	10.22	339.0
	S2	2249	96.10	232.4		1907	15.07	339.0
	S3	2157	98.64	232.4		1346	18.71	339.8
H3	**S1**	970	73.61	231.0		1728	8.09	309.2
	S2	562	90.47	231.0		505	23.91	306.8
	S3	418	96.94	231.0		191	56.56	311.2

From this table we can see that '5 × n' gives us a good balance between average success % and the average cost. Allowing higher sideways movements leads to higher success %, but the solutions tend to have a higher cost due to sideways movement. Comparing '1 × n' and '5 × n' they have almost the same average cost but '5 × n' has higher success % and hence it was selected.

Snapshot of the output from the code:



The yellow squares show the starting position & gray squares the final position for A* and Hill Climbing Algorithm. The values in the squares are the weight of the queens.
