Report AI

Question 1 – Backtracking.

The brute force algorithm varies a lot in the amount of calls. For 10 queens, the minimum amount of possible calls is 10 by putting a queen on the board 10 times without breaking the constraints. If you get lucky, some tries get around 10-15 calls. Although, There are also tries when there are over 300 calls, This is because the variables and values are selected randomly without taking the constraints into account. A table with some tries are shown below to demonstrate the randomness.

Run	**Calls**	**Speed**
1	178	6948 calls/s
2	388	7289 calls/s
3	91	6394 calls/s
4	117	6513 calls/s
5	15	4978 calls/s
6	23	6255 calls/s
7	14	5304 calls/s
8	42	6082 calls/s
9	11	5295 calls/s
10	257	7839 calls/s
Mean	**113.5**	**6283 calls/s**
Standard deviation	**120**	**872.8 calls/s**

Question 2 – Forward Checking.

Forward checking removes impossible situations before getting there, which reduces the amount of calls greatly. Implementing the Minimum remaining values (MRV) and Least constraining value (LCV) helps the algorithm a lot in terms of the number of calls, although they require some computation so the amount of calls/s will drop. LCV calculates the value that breaks the least constraints so that forward checking takes the right path right away, instead of taking a path that is more likely to fail. MRV chooses the next variable to assign a value to by looking how many possible values are left. It picks the variable with the fewest possible legal values. It does this to (again) get the algorithm on the right path. As you can see in the table below, the number of calls are reduced greatly, but the number of calls per second has gone up.

MRV and LCV = OFF

Run	Calls	Speed
1	13979	2109.4 calls/s
2	170	1795.7 calls/s
3	32260	1804.5 calls/s
4	4976	2063.5 calls/s
5	1630	2074.6 calls/s
6	2983	1948.3 calls/s
7	134	1703.9 calls/s
8	2865	2015.3 calls/s
9	11504	1887.4 calls/s
10	1595	2184.7 calls/s
MEAN	**7209,6**	**1958,7 calls/s**
STDEV	**9976,6**	**157,0 calls/s**

MRV and LCV = ON

Run	Calls	Speed
1	53	75.5 calls/s
2	53	77.2 calls/s
3	53	85.5 calls/s
4	53	76.0 calls/s
5	53	85.3 calls/s
6	53	76.9 calls/s
7	53	86.9 calls/s
8	53	75.4 calls/s
9	53	79.3 calls/s
10	53	81.6 calls/s
MEAN	**53**	**79.9 calls/s**
STDEV	**53**	**4.5 calls/s**

Question 3 – AC3.

AC3 gives an even better improvement for the algorithm. It makes more checks to remove configurations that lead to constraints, which reduces the amount of calls greatly. Instead of only checking the neighbors as in forward checking, it checks the neighbors first, and then all the neighbors of the variable that removed a value. This allows for checking if a domain is empty, and thus no possible value can be assigned, way sooner. Again, MRV and LCV makes the algorithm even better which leads to only 51 calls for 50 queens. This seems not much better as forward checking, as it had 53 calls, but this is because there is a bit of luck present. If you for example take 60 queens, fc makes 126 calls, while ac3 makes only 61 calls. A difference you can notice immediately is when MRV and LCV is turned off, it went from 7209 calls on average with forward checking, to 118 calls on average with ac3.

MRV and LCV = OFF

Run	Calls	Speed
1	143	73.9 calls/s
2	87	76.4 calls/s
3	53	60.1 calls/s
4	68	105.7 calls/s
5	185	60.7 calls/s
6	341	66.3 calls/s

MRV and LCV = ON

Run	Calls	Speed
1	51	36.5 calls/s
2	51	37.1 calls/s
3	51	36.5 calls/s
4	51	38.0 calls/s
5	51	36.3 calls/s
6	51	37.4 calls/s

7	58	99.8 calls/s
8	95	51.8 calls/s
9	65	100.2 calls/s
10	85	95.4 calls/s
MEAN	**118**	**81,1 calls/s**
STDEV	**88,5**	**19,6 calls/s**

7	51	37.1 calls/s
8	51	36.8 calls/s
9	51	36.1 calls/s
10	51	35.9 calls/s
MEAN	**51**	**36.8 calls/s**
STDEV	**51**	**0.6 calls/s**

Question 4 – Sudoku.

Backtracking: Too long, aborted at:	388580 calls [18:51, 348.15 calls/s]
Forward checking:	12977 calls [00:04, 2772.17 calls/s]
Ac3:	5240 calls [00:12, 437.03 calls/s]

Although forward checking makes more than double the amount of calls, it is 3 times faster than ac3. This is because ac3 does more calculations to remove impossible configurations early. The sudoku is too hard to solve with the backtracking algorithm.

Question 5 – Feedback.

The assignment was well put together, although there were a few points that needed more clarification for me. For the forward checking algorithm, I was confused to how to implement the “forwardChecking” function. I (and other students I asked) thought we had to implement the neighbors and isValidPairwise functions ourselves, which seemed impossible to do for general problems. Later a found out that it is an abstract method, already implemented in Queens.py. I was also struggling with implementing the ac3 algorithm, the given code in the slides were a bit confusing. There was also an extra parameter in the function “ac3” called variable which was not present in the code from the slides, so I did not know what to do with it exactly.