Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

CMPE 300 ANALYSIS OF ALGORITHMS

PROJECT 3 - ANSWERS

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

PART 1

d) (You can adjust the length of the tables)

Success, n=6

Step	Columns	Available
1	[3]	[0, 1, 5]
2	[3, 0]	[2, 4]
3	[3, 0, 4]	[1]
4	[3, 0, 4, 1]	[5]
5	[3, 0, 4, 1, 5]	[2]
6	[3, 0, 4, 1, 5, 2]	[]

Visualization of the table

	0	1	2	3	4	5
0				Q		
1	Q					
2					**Q**	
3		**Q**				
4						**Q**
5			**Q**			

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

Success, n=6

Step	Columns	Available
1	[2]	[0, 4, 5]
2	[2, 5]	[1, 3]
3	[2, 5, 1]	[4]
4	[2, 5, 1, 4]	[0]
5	[2, 5, 1, 4, 0]	[3]
6	[2, 5, 1, 4, 0, 3]	[]

Visualization of the table

	0	1	2	3	4	5
0			Q			
1						**Q**
2		**Q**				
3					**Q**	
4	Q					
5				**Q**		

Failure, n=6

Step	Columns	Available
1	[4]	[0, 1, 2]
2	[4, 1]	[3, 5]
3	[4, 1, 3]	[0, 5]
4	[4, 1, 3, 5]	[2]
5	[4, 1, 3, 5, 2]	[]

Failure, n=6

Step	Columns	Available
1	[4]	[0, 1, 2]
2	[4, 1]	[3, 5]
3	[4, 1, 5]	[0, 2]
4	[4, 1, 5, 2]	[]

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

Success, n=8

Step	Columns	Available
1	[5]	[0, 1, 2, 3, 7]
2	[5, 3]	[0, 1, 6]
3	[5, 3, 6]	[0, 4]
4	[5, 3, 6, 0]	[2, 7]
5	[5, 3, 6, 0, 2]	[4]
6	[5, 3, 6, 0, 2, 4]	[1, 7]
7	[5, 3, 6, 0, 2, 4, 1]	[7]
8	[5, 3, 6, 0, 2, 4, 1, 7]	

Visualization of the table

	0	1	2	3	4	5	6	7
0						Q		
1				**Q**				
2							**Q**	
3	Q							
4			**Q**					
5					**Q**			
6		**Q**						
7								**Q**

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

Success, n=8

Step	Columns	Available
1	[3]	[0, 1, 5, 6, 7]
2	[3, 5]	[0, 2, 7]
3	[3, 5, 7]	[1, 2, 4]
4	[3, 5, 7, 2]	[0, 4, 6]
5	[3, 5, 7, 2, 0]	[6]
6	[3, 5, 7, 2, 0, 6]	[1, 4]
7	[3, 5, 7, 2, 0, 6, 4]	[1]
8	[3, 5, 7, 2, 0, 6, 4, 1]	[]

Visualization of the table

	0	1	2	3	4	5	6	7
0				Q				
1						**Q**		
2								**Q**
3			**Q**					
4	Q							
5							**Q**	
6					**Q**			
7		**Q**						

Failure, n=8

Step	Columns	Available
1	[5]	[0, 1, 2, 3, 7]
2	[5, 3]	[0, 1, 6]
3	[5, 3, 1]	[4, 6, 7]
4	[5, 3, 1, 6]	[2, 4]
5	[5, 3, 1, 6, 2]	[]

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

Failure, n=8

Step	Columns	Available
1	[2]	[0, 4, 5, 6, 7]
2	[2, 0]	[3, 5, 6, 7]
3	[2, 0, 3]	[1, 6, 7]
4	[2, 0, 3, 6]	[4]
5	[2, 0, 3, 6, 4]	[1]
6	[2, 0, 3, 6, 4, 1]	[]

Success, n=10

Step	Columns	Available
1	[5]	[0, 1, 2, 3, 7, 8, 9]
2	[5, 7]	[0, 1, 2, 4, 9]
3	[5, 7, 1]	[3, 4, 6]
4	[5, 7, 1, 4]	[0, 2, 6, 8]
5	[5, 7, 1, 4, 0]	[8, 9]
6	[5, 7, 1, 4, 0, 8]	[3, 6]
7	[5, 7, 1, 4, 0, 8, 3]	[9]
8	[5, 7, 1, 4, 0, 8, 3, 9]	[2, 6]
9	[5, 7, 1, 4, 0, 8, 3, 9, 6]	[2]
10	[5, 7, 1, 4, 0, 8, 3, 9, 6, 2]	[]

Visualization of the table

	0	1	2	3	4	5	6	7	8	9
0						Q				
1								**Q**		
2		**Q**								
3					**Q**					
4	Q									
5									**Q**	
6				**Q**						
7										**Q**
8							**Q**			
9			**Q**							

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

Success, n=10

Step	Columns	Available
1	[0]	[2, 3, 4, 5, 6, 7, 8, 9]
2	[0, 7]	[1, 3, 4, 5, 9]
3	[0, 7, 1]	[4, 6, 8]
4	[0, 7, 1, 8]	[2, 5, 6]
5	[0, 7, 1, 8, 5]	[2, 9]
6	[0, 7, 1, 8, 5, 2]	[4, 9]
7	[0, 7, 1, 8, 5, 2, 9]	[3]
8	[0, 7, 1, 8, 5, 2, 9, 3]	[6]
9	[0, 7, 1, 8, 5, 2, 9, 3, 6]	[4]
10	[0, 7, 1, 8, 5, 2, 9, 3, 6, 4]	[]

Visualization of the table

	0	1	2	3	4	5	6	7	8	9
0	Q									
1								Q		
2		Q								
3									Q	
4						Q				
5			Q							
6										Q
7				Q						
8							Q			
9					Q					

Failure, n=10

Step	Columns	Available
1	[9]	[0, 1, 2, 3, 4, 5, 6, 7]
2	[9, 7]	[0, 1, 2, 3, 4, 5]
3	[9, 7, 5]	[0, 1, 2, 3, 8]
4	[9, 7, 5, 3]	[0, 1, 6, 8]
5	[9, 7, 5, 3, 1]	[6]
6	[9, 7, 5, 3, 1, 6]	[4, 8]
7	[9, 7, 5, 3, 1, 6, 8]	[]

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

Failure, n=10

Step	Columns	Available
1	[9]	[0, 1, 2, 3, 4, 5, 6, 7]
2	[9, 5]	[0, 1, 2, 3, 8]
3	[9, 5, 2]	[0, 4, 8]
4	[9, 5, 2, 4]	[1, 6, 7]
5	[9, 5, 2, 4, 6]	[0, 3, 8]
6	[9, 5, 2, 4, 6, 0]	[]

d)

n	Number of Success	Number of Trials	Probability
6	718	10000	0.0718
8	1301	10000	0.1301
10	607	10000	0.0607

PART 2

c)

n = 6

k	Number of Success	Number of Trials	Probability
0	10000	10000	1.0
1	6623	10000	0.6623
2	2217	10000	0.2217
3	1106	10000	0.1106
4	781	10000	0.0781
5	660	10000	0.066

n = 8

k	Number of Success	Number of Trials	Probability
0	10000	10000	1.0
1	10000	10000	1.0
2	8783	10000	0.8783
3	4928	10000	0.4928
4	2629	10000	0.2629
5	1630	10000	0.1630
6	1424	10000	0.1424
7	1315	10000	0.1315

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

n = 10

k	Number of Success	Number of Trials	Probability
0	10000	10000	1.0
1	10000	10000	1.0
2	10000	10000	1.0
3	8017	10000	0.8017
4	4234	10000	0.4234
5	1967	10000	0.1967
6	1090	10000	0.109
7	774	10000	0.0774
8	703	10000	0.0703
9	648	10000	0.0648

d) Comments

In the first part, our solution to the problem was using Las Vegas Algorithm, which is a randomized algorithm that employs the randomness as part of its procedure. We have seen how the algorithm starts with placing a queen to a random cell in the first row of the board and then searches for the place in the next row by calculating the available cells and chooses among them randomly again. The computation yielded both successful and unsuccessful results in each run. After 10000 number of trials for each case of n=6, n=8, and n=10 we have the conclusion:

Probability that it will come to a solution is 0.0718 for 6*6 board

Probability that it will come to a solution is 0.1301 for 8*8 board

Probability that it will come to a solution is 0.0607 for 10*10 board

It is clear to see that there is not any proportionality between the table size n and the probability of a successful computation. We have obtained the minimum probability at 10*10 board with a probability of 0.0607 successful computation, while 8*8 board yielded the maximum probability with a probability of 0.1301.

In the second part of the project, we have approached the placement of n queens to a n*n chess board so that none of them is threatening each other problem in a deterministic way. First, we have placed k number of queens in the first k row of the n*n board with k < n using the Las Vegas Algorithm from the first part of the project. Then, we used the recursive backtracking algorithm to fill out the rest of the board. Using the backtracking algorithm added a deterministic computation for filling the chess board compared to the probabilistic approach of Las Vegas Algorithm. We tested our program from k = 0 to k = (n – 1) for each of 6*6, 8*8, and 10*10 board. For each case, the tendency of the results was in the same direction although the results vary.

For a 6*6 board using the algorithm, placing 0 queens beforehand had the best probability of success with 1.0 while placing 5 (n – 1) queens beforehand had the worst probability of success with 0.066. The probability of successful computation decreased as the k parameter increased for each value of k.

Altay Acar – 2018400084
Engin Oğuzhan Şenol - 2020400324

For an 8*8 board using the algorithm, placing 0 and 1 queens beforehand had the best probability of success with 1.0 while placing 7 (n – 1) queens beforehand had the worst probability of success with 0.1315. The probability of successful computation decreased as the k parameter increased for each value of k.

For a 10*10 board using the algorithm, placing 0, 1, and 2 queens beforehand had the best probability of success with 1.0 while placing 5 (n – 1) queens beforehand had the worst probability of success with 0.0648. The probability of successful computation decreased as the k parameter increased for each value of k.

It is clear to see that the deterministic algorithm performs with a 100% success rate when k = 0. And as n increases the total number of k's that yield a 100% successful result increases. On the other hand, k and the probability of successful computation has an inverse proportionality: as k increases, the probability of a successful computation decreases.

Compared to the results of part 1, we can say that as k increases, the probability of a successful computation approaches to the results of the probabilistic Las Vegas Algorithm. Also, the performance of the deterministic algorithm when the performance of 6*6 board, 8*8 board, and 10*10 board is compared, yielded a parallel result to the part 1: computation for 8*8 board yielded the best result with a probability of 0.1315 for k = (n – 1), while the computation for 10*10 board yielded the worst result with a probability of 0.0648 for k = (n – 1). Comparing these results to the results of part 1:

For a 6*6 board:

Probability that probabilistic algorithm will come to a solution is 0.0718 and the worst-case probability of deterministic algorithm will come to a solution is 0.066. When $k \leq (n - 2) = 4$, the deterministic algorithm yields a better result compared to the probabilistic algorithm. In the contrast of both 8*8 and 10*10 board computations, in the case of 6*6 board, the deterministic algorithm performed better than the probabilistic algorithm not for every value of k.

For an 8*8 board:

Probability that probabilistic algorithm will come to a solution is 0.1301 and the worst-case probability of deterministic algorithm will come to a solution is 0.1315. When $k \leq (n - 1) = 7$, the deterministic algorithm yields a better result compared to the probabilistic algorithm. In the case of 8*8 board, deterministic algorithm performed better than the probabilistic algorithm for every value of k.

For a 10*10 board:

Probability that probabilistic algorithm will come to a solution is 0.0607 and the worst-case probability of deterministic algorithm will come to a solution is 0.0648. When $k \leq (n - 1) = 9$, the deterministic algorithm yields a better result compared to the probabilistic algorithm. In the case of 10*10 board, deterministic algorithm performed better than the probabilistic algorithm for every value of k.