# CMPE 300 ANALYSIS OF ALGORITHMS
# PROJECT 3 - ANSWERS

# PART 1

## 1.1) Fill the steps of one successful and one unsuccessful execution for each p value

### *1.1.1) Success - p=0.7*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | 1 | 44 | 17 | -1 | 19 | -1 | -1 |
| **1** | 15 | **46** | **-1** | **0** | **-1** | **22** | **-1** | **20** |
| **2** | 2 | **43** | **16** | **45** | **18** | **25** | **-1** | **-1** |
| **3** | 47 | **14** | **3** | **26** | **23** | **12** | **21** | **-1** |
| **4** | 42 | **37** | **40** | **13** | **32** | **5** | **24** | **7** |
| **5** | 39 | **34** | **27** | **4** | **-1** | **8** | **11** | **30** |
| **6** | 36 | **41** | **38** | **33** | **28** | **31** | **6** | **9** |
| **7** | -1 | **-1** | **35** | **-1** | **-1** | **10** | **29** | **-1** |

### *1.1.2) Unsuccessful - p=0.7*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | 28 | 25 | -1 | 31 | 20 | 23 | -1 |
| **1** | -1 | **7** | **30** | **27** | **24** | **-1** | **32** | **21** |
| **2** | 29 | **26** | **11** | **6** | **19** | **22** | **-1** | **0** |
| **3** | 12 | **-1** | **8** | **-1** | **-1** | **5** | **18** | **33** |
| **4** | -1 | **-1** | **13** | **10** | **-1** | **34** | **1** | **-1** |
| **5** | -1 | **9** | **-1** | **-1** | **-1** | **17** | **4** | **37** |
| **6** | -1 | **-1** | **-1** | **14** | **35** | **2** | **-1** | **16** |
| **7** | -1 | **-1** | **-1** | **-1** | **-1** | **15** | **36** | **3** |

### 1.1.3) Success - p=0.8

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | -1 | 42 | 9 | 14 | 17 | -1 | -1 |
| **1** | -1 | **10** | **13** | **18** | **41** | **4** | **1** | **16** |
| **2** | -1 | **43** | **30** | **11** | **8** | **15** | **20** | **3** |
| **3** | 29 | **12** | **7** | **40** | **19** | **2** | **5** | **0** |
| **4** | 44 | **-1** | **48** | **31** | **6** | **39** | **24** | **21** |
| **5** | 47 | **28** | **45** | **38** | **23** | **34** | **53** | **-1** |
| **6** | -1 | **49** | **32** | **27** | **36** | **51** | **22** | **55** |
| **7** | -1 | **46** | **37** | **50** | **33** | **26** | **35** | **52** |

### *1.1.4) Unsuccessful - p=0.8*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | 13 | -1 | 1 | 4 | 7 | -1 | -1 | -1 |
| **1** | 2 | **-1** | **10** | **-1** | **0** | **5** | **8** | **-1** |
| **2** | -1 | **12** | **3** | **6** | **9** | **-1** | **-1** | **-1** |
| **3** | -1 | **-1** | **-1** | **11** | **-1** | **-1** | **-1** | **-1** |
| **4** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **5** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **6** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **7** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |

### 1.1.5) Success - p=0.85

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | -1 | 29 | 32 | -1 | 4 | 51 | 54 |
| **1** | 30 | **33** | **44** | **39** | **50** | **53** | **48** | **3** |
| **2** | 43 | **38** | **31** | **28** | **5** | **14** | **55** | **52** |
| **3** | 34 | **27** | **40** | **45** | **-1** | **49** | **2** | **47** |
| **4** | 37 | **42** | **35** | **6** | **15** | **46** | **13** | **56** |
| **5** | 26 | **9** | **24** | **41** | **58** | **1** | **16** | **19** |
| **6** | 23 | **36** | **7** | **10** | **21** | **18** | **57** | **12** |
| **7** | 8 | **25** | **22** | **59** | **0** | **11** | **20** | **17** |

### *1.1.6) Unsuccessful - p=0.85*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | -1 | -1 | 3 | -1 | -1 | 0 | 11 |
| **1** | -1 | **-1** | **-1** | **-1** | **7** | **2** | **-1** | **-1** |
| **2** | -1 | **-1** | **6** | **-1** | **4** | **-1** | **10** | **1** |
| **3** | -1 | **-1** | **-1** | **-1** | **-1** | **8** | **-1** | **-1** |
| **4** | -1 | **-1** | **-1** | **5** | **-1** | **-1** | **-1** | **9** |
| **5** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **6** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **7** | -1 | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |

## 1.2) Fill the table and comment on it

### 1.2.1)

| p | Number of Success | Number of Trials | Probability | Total Time of Execution |
|---|---|---|---|---|
| **0.7** | 16323 | 100000 | 0.16323 | 33.2091 seconds |
| **0.8** | 2514 | 100000 | 0.02514 | 33.9310 seconds |
| **0.85** | 615 | 100000 | 0.00615 | 31.1630 seconds |

### 1.2.1) Comments

Also answer these questions while commenting

1- How do changes on p affect total success probability and total execution time?

***The parameter p in the code determines the minimum successful squares required for a tour to be considered successful. As p increases, the minimum successful squares also increase. This means that the probability of a successful tour decreases because it becomes more difficult to meet the higher threshold.***

***On the other hand, increasing p may also increase the total execution time. This is because as the minimum successful squares increase, it may take longer to find a successful tour that meets the higher threshold. However, the exact impact on execution time depends on various factors such as the size of the chessboard and the number of trials performed.***

2- Define trade-offs of the algorithm.

***Time Complexity: The algorithm uses a Las Vegas approach to find successful knight's tours. It performs a large number of trials (total_trials) to determine the probability of a successful tour. As a result, the time complexity of the algorithm is directly proportional to the total number of trials***

***Accuracy vs Efficiency: The algorithm aims to find successful knight's tours with a given probability (p_value). By increasing the number of trials (total_trials), the algorithm can achieve a higher accuracy in estimating the probability. However, this comes at the cost of increased execution time,***

***The algorithm relies on random moves to explore the chessboard. While this randomness allows for a more diverse exploration, it also introduces the possibility of getting stuck in a loop or not finding a successful tour. The algorithm tries to mitigate this by backtracking***

***and exploring different moves, but it is not guaranteed to find a successful tour in every trial.***

# PART 2

## 2.1) Fill the tables and comment on them

### 2.1.1) p = 0.7

| k | Number of Success | Number of Trials | Probability | Total Time |
|---|---|---|---|---|
| **0** | 100000 | 100000 | 1 | 43.187 seconds |
| **2** | 100000 | 100000 | 1 | 43.7396 seconds |
| **3** | 99945 | 100000 | 0.99945 | 40.2887 seconds |

### 2.1.2) p = 0.8

| k | Number of Success | Number of Trials | Probability | Total Time |
|---|---|---|---|---|
| **0** | 100000 | 100000 | 1 | 40.725 seconds |
| **2** | 100000 | 100000 | 1 | 42.713 seconds |
| **3** | 99941 | 100000 | 0.99941 | 57.4618 seconds |

### 2.1.1) p = 0.85

| k | Number of Success | Number of Trials | Probability | Total Time |
|---|---|---|---|---|
| **0** | 100000 | 100000 | 1 | 54.789 seconds |
| **2** | 100000 | 100000 | 1 | 53.062 seconds |
| **3** | 99934 | 100000 | 0.99934 | 58.0547 seconds |

### 2.1.2) Comments

Also answer these questions while commenting

1- How does total time change with k?

***As k increases, the total time taken to complete the tour generally increases. This is because a higher value of k means that the knight has to make more fixed moves before starting the backtracking process. Each fixed move involves checking the validity of the move and updating the visited squares, which takes additional time.***

***Therefore, as k increases, the total time taken to complete the tour also increases.***

2- How do total time change with p for a specific k value? How does this change different from the first part?

***Increasing p_value may also increase the total time taken to find a successful tour. This is because as p_value increases, the algorithm explores more possible moves(need to satisfy more successful move ), which can lead to longer search times.***

3- Run this algorithm for each p value for k a value larger than 10 multiple times. What are your thoughts?

***As we set k = 15 and test for each p value, as p value increases probabilities decrease as expected. Also, its probability of success is lower than k < 15 as expected as well. Because we put backtracking mechanism into restriction. So possibility of finding success path gets lower as we increase the value of 'k'.***

# PART 3

In this part, you will compare Part1 and Part2 algorithms according to their ability to solve the actual Knight's Problem where p=1.

- Run Part1 algorithm with p=1.
- Run Part2 algorithm with p=1 and k=0.
- Run Part2 algorithm with p=1 and a k value you think will work well.

Clearly state your findings and comment on them. When would you choose Part1 algorithm and when would you choose the other?

***As we set the p to1 in Part1 we clearly see that in 100.000 trial finding a solution is nearly impossible. We tried 20 times and never got an optimal result. However, in part 2 setting p to 1 and k to 0, does not change to probability of finding a perfect solution. If it can backtrack every single time, algorithm find a desired result every time. In part 2, setting k to higher values such as 15-20-25-30 we got lower probability of finding optimal result, but also lower total time consumed.***

***Part 2 is like tradeoff between time and probability. If we choose higher k values, we get lower execution times due to restrictions in backtracking steps. We may choose Part 1 algorithm if we prioritize speed over finding the most optimal solution and backtracking is not feasible or not necessary for our specific problem. On the other and, we may choose Part2 when we want to find more optimal solutions. As we set k higher, we lower the probability of finding an optimal result because we put backtracking mechanism into restriction. So, possibility of finding success path gets lower as we increase the value of 'k'. Moreover, we lower the total time consumed. If we set p to 1 and k to 18, we get approximately 91.38% getting optimal solution and total time consumed is less than part1.Therefore, this is a more optimal solution than part 1.***