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

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

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | 14 | -1 | -1 | 9 | -1 | -1 | -1 |
| **1** | -1 | **-1** | **8** | **15** | **12** | **-1** | **-1** | **-1** |
| **2** | 7 | **-1** | **13** | **-1** | **-1** | **10** | **-1** | **-1** |
| **3** | -1 | **-1** | **6** | **11** | **16** | **-1** | **0** | **-1** |
| **4** | 5 | **-1** | **17** | **20** | **-1** | **-1** | **-1** | **-1** |
| **5** | -1 | **19** | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| **6** | -1 | **4** | **21** | **18** | **-1** | **2** | **-1** | **-1** |
| **7** | 22 | **-1** | **-1** | **3** | **-1** | **-1** | **-1** | **-1** |

## 1.1.3) Success - p=0.8

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

## *1.1.4) Unsuccessful - p=0.8*

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

## 1.1.5) Success - p=0.85

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

## *1.1.6) Unsuccessful - p=0.85*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 |
| **1** | -1 | **-1** | **-1** | **-1** | **-1** | **0** | **-1** | **-1** |
| **2** | -1 | **-1** | **-1** | **-1** | **2** | **-1** | **-1** | **9** |
| **3** | -1 | **22** | **-1** | **-1** | **7** | **10** | **-1** | **-1** |
| **4** | -1 | **-1** | **-1** | **21** | **14** | **3** | **8** | **-1** |
| **5** | 23 | **20** | **-1** | **-1** | **11** | **6** | **13** | **16** |
| **6** | -1 | **-1** | **24** | **-1** | **18** | **15** | **4** | **-1** |
| **7** | 25 | **-1** | **19** | **-1** | **5** | **12** | **17** | **-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** | 16721 | 100000 | %16.721 | 10.30 (seconds) |
| **0.8** | 2437 | 100000 | %2.437 | 10.51 (seconds) |
| **0.85** | 587 | 100000 | %0.587 | 10.21 (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?

2- Define trade-offs of the algorithm.

(1) As p increases, knight is expected to traverse more squares without falling into a dead-end state. Our policy is random in this part so the number of successes and probability decreased but it didn't affect the total time of execution by a significant amount compared to the notable change in the number of successes.

(2) In this part we examined the completely random algorithm while forwarding on the chess board. Algorithm gave us a good and stable running time thanks to the random decision mechanism. But the probability decreased exponentially as we increased the p. This algorithm might be the optimal choice if we will choose a low p rate. But there is a tradeoff, as we increase the p rate -even if the algorithm runs fast- it will reach a less number of successes.

# 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 | %100 | 35.56(seconds) |

| | | | | |
|---|---|---|---|---|
| **2** | 100000 | 100000 | %100 | 33.84(seconds) |
| **3** | 99944 | 100000 | %99.944 | 32.88(seconds) |

### 2.1.2) p = 0.8

| k | Number of Success | Number of Trials | Probability | Total Time |
|---|---|---|---|---|
| **0** | 100000 | 100000 | %100 | 40.33(seconds) |
| **2** | 100000 | 100000 | %100 | 38.41(seconds) |
| **3** | 99938 | 100000 | %99.938 | 37.47(seconds) |

### 2.1.1) p = 0.85

| k | Number of Success | Number of Trials | Probability | Total Time |
|---|---|---|---|---|
| **0** | 100000 | 100000 | %100 | 42.10(seconds) |
| **2** | 100000 | 100000 | %100 | 40.19(seconds) |
| **3** | 99935 | 100000 | %99.935 | 39.12(seconds) |

### 2.1.2) Comments

Also answer these questions while commenting
1- How does total time change with k?

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

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

(1) As k increases, total time decreases because the backtracking algorithm starts from where the random walk stops and it cannot backtrack the path which random walk forwarded. It decreases the average recursion count in the backtracking algorithm so the total time.
(2) For a fixed k as p increases total time also increases.When we make a comparison with the first part, there is no significant increase or decrease in the first part. In contrast to the first part, backtracking is getting computationally expensive as p increases and algorithms are trying to find a way to exceed p proportion of the board by trying all possible paths.

(3) Lastly, we tried the 2nd part algorithm with a k value higher than 10. In this part we did our experiment with k = 12 and we observe that even if there is no problem with 0.7 and 0.8 p. There are some twitches in the flow of execution p = 0.85 part and we tried to increase k more at 16 algorithm takes too long to execute. We used Warnsdorff's rule to optimize backtracking and probably complexity is too high after taking random k steps -even if we use optimization- and the problem is hard to reach a feasible solution in a particular amount of time (because we make too many random steps before starting deterministic approach).
Also when we set the k =12. Total time is reduced dramatically, as we mentioned above due to the algorithm having less backtracking leaf after 13th steps. Running time average is about 0.30(seconds).

## 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.

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

Clearly state your findings and comment on them. 4)When would you choose Part1 algorithm and when would you choose the other?

1)When we ran part1 algorithm with p = 1, the number of successful tours became 0 as in 100000 tours our knight wasn't able to visit all of the squares without backtracking.(maximum tour length is 61) The execution lasted in 10.22 seconds, still not much different than the experiment p = (0.7, 0.8, 0.85). The probability of tracing all board with random decision is not impossible but has small probability. Even if the maximum the algorithm can do is 61 in a try with 100000, when we observe the board it seems reaching a successful run can take millions of try.

2)When we run part2 algorithm with p = 1 and k = 0, the number of successful tours turns out to be 100000 when the number of trials is 100000 so the probability of a successful tour is 1.0. This means that without taking a step, whenever we start to trace a path our optimized backtracking algorithm finds a way.

The total time taken is 47.62 seconds which is higher than

35.56 seconds when p = 0.7 and k = 0

40.33 seconds when p = 0.8 and k = 0

42.10 seconds when p = 0.85 and k = 0.

So we may end up with the comment that when p increases until 1, total time of execution increases with it also.

3) We tried k > 0 values, for (100000 try) algorithm finds values for some step but at a point it takes too long time for particular initial steps.

**For small k values [1-5]:**

After starting an execution for 100000 try. When we take one random step, algorithm averagely finds more way before stucking. Stucking means, algorithm takes too much time and doesn't return a result before finding a way. K values bigger than 1 execute less successful runs before stucking.

So we can say that the initial step is important if we want to solve the complete knight's tour problem in a limited amount of time.

**For medium k values[30-35]:**

We tried some medium sizes k. Even if it still takes too much time it turns into a false result after a long time (3 minutes for 35, just one try among 100000).

**For medium-high k values[40-45]:**

When we increase the k values more, naturally more random walks fall into the dead-end state. But, random walk which are not in a dead-end state could find a successful path on board.
7/100000 for k = 40 in 60s.
0/100000 for k = 45 in 5s. (probably most of them entered into the dead-end state)

For high values it couldn’t find a successful path again.

4) If we are dealing with high p values such as 0.8, 0.85; we should encounter backtracking so we should use part2 algorithm, but if we are dealing with low p values such as 0.3, 0.35; we may use part1 algorithm.

If we have to make random steps and p value is too high (close to 1.0) part2 algorithm can take too much time but whenever it returns we can say that output is correct so there is a feasible way.

If we want to find any solution for high p values (close to 1.0) and we have to make random steps before it, we can set k values about (38,42).

If we want to find any solution for high p values as fast as possible and if we don’t have to complete all tries we can use small (0,5) k values.

If we don’t have to make random steps before deterministic approach, part 2 algorithms is best for this situation.
Final comment: We didn’t find a chance to try but, our algorithm in part2 uses a deterministic approach after taking k steps. For the real problem (p=1.0), it takes too much time when we take a small number of steps. Maybe we can add some randomness to backtracking the algorithm in a way so it might find a way without using an optimization. If there is no path it will take too much time again.