CMPE 300: Analysis of Algorithms
Project 3

Aral Dörtoğul
aral.dortogul@boun.edu.tr

Hasan Baki Küçükçakıroğlu
hasan.kucukcakiroglu@boun.edu.tr

Strict Deadline: Sunday 14 January 2024 23:59

1 Introduction

In this project, your task is to develop a program that employs a probabilistic algorithm to solve the Knight's Tour problem on a chessboard. Beyond the implementation, the project encourages a thorough analysis of the algorithm's performance under changing parameters, providing you with valuable insights into its intricacies and effectiveness.

2 The Knight's Tour Problem

The Knight's Tour is a classic chess problem that involves moving a knight on a chessboard in such a way that the knight visits every square exactly once. The challenge is to find a sequence of legal knight moves that accomplishes this task. The knight makes L-shaped moves: two squares in one direction and one square perpendicular to that. Figure 1 shows an example solution to the problem.

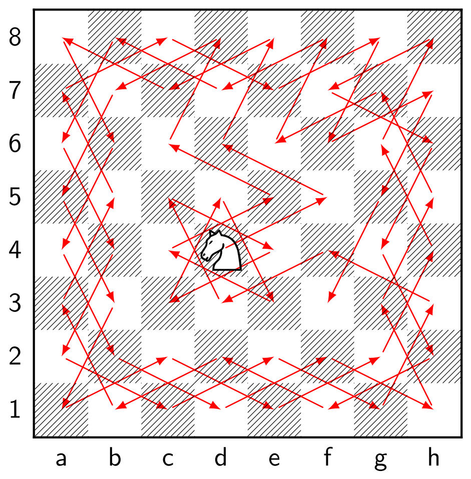


Figure 1: An Example Solution to the Knight's Tour Problem

3 Part 1

In the first part of the project, your task is to address a different version of the Knight's Tour problem for an 8×8 board using a completely random approach. The knight starts its journey from a randomly chosen square and, at each step, selects the next move uniformly at random from the available squares. The tour progresses until the knight visits all the squares or encounters a dead end.

For simplification, success for the knight is defined in this project as **having traversed more than p percent of the squares of the board by the conclusion of the tour.** For example, for p=70%, traveling more than $64 \times 70\% = 44.8$ (45) squares is enough for the knight to be considered successful. You are free to terminate a tour when enough squares to make the tour valid are stepped into.

3.1 Implementation

Implement the algorithm in any programming language you like. (**Python** is recommended.) Run the program 100,000 times for each p = 0.7, 0.8, and 0.85.

3.1.1 Output

For each value of 'n', follow the steps below and record the results in separate files named `results_p.txt`, where 'p' can be 0.7, 0.8, or 0.85. You will need to include these txt files in your submission.

1. Each run begins with the line:

   `Run <count>:  starting from (<row>,<column>)`

   to indicate the count of the run and initial position of the knight on the chessboard. `<count>` ranges from 1 to 100000, and `<row>` and `<column>` range from 0 to n-1.

2. Record each step of the knight's tour:

   `Stepping into (<next_row>,<next_column>)`

3. After the tour concludes, the result of the run is written in the format:

   `<Unsuccessful/Successful> - Tour length:  <length>`

4. Finally, print the sequence of squares stepped on (starting from 0) an n×n chessboard. Place -1 in squares that were not visited.

The following is an example output when p = 0.7:

```
Run 1: starting from (7,2)
Stepping to (5,1)
Stepping to (4,3)
Stepping to (6,2)
Stepping to (7,0)
Unsuccessful - Tour length: 5
-1  -1  -1  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1  -1
-1  -1  -1   2  -1  -1  -1  -1
-1   1  -1  -1  -1  -1  -1  -1
-1  -1   3  -1  -1  -1  -1  -1
 4  -1   0  -1  -1  -1  -1  -1

Run 2: starting from (2,4)
```

```
Stepping to (0,5)
Stepping to (1,3)
Stepping to (3,4)
...
... (Continue printing steps)
...
Stepping to (2,5)
Stepping to (0,4)
Stepping to (1,6)
Successful - Tour length: 58
32  -1  34  -1  56   1  -1  -1
35  16  31   2   7  48  57  -1
30  33   6  17   0  55   8  47
 5  36  15  20   3  18  49  54
14  29   4  23  40  53  46   9
37  26  21  12  19  50  43  52
28  13  24  39  22  41  10  45
25  38  27  -1  11  44  51  42
...
... (Continue printing all the runs)
```

After the execution is finished, your program should print the number of successful tours, number of trials, and the probability of successful tours to console in the format of:

```
LasVegas Algorithm With p = 0.7
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.8
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.85
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y
```

Where x values are empirical data you get and y values are (x/100000). After obtaining the estimated probability of successful tours, you should fill the appropriate tables and answer the questions in the answer sheet under the heading “Part 1”.

4 Part 2

In this part, you are required to implement a similar algorithm that advances the knight's tour by 'k' steps randomly and then subsequently employes a deterministic approach using backtracking to find a tour that covers more than **p** of the squares.

Backtracking, as applied to the Knight's Tour problem, is a systematic algorithmic approach that involves exploring different paths on the chessboard, where, upon encountering an unsuccessful move, the algorithm backtracks to the previous decision point, allowing for the exploration of alternative moves in the quest to find a successful tour.

4.1 Output

You should implement the algorithm for k = 0, 2 and 3; and try to solve the problem 100,000 times for each. The program should give the following summary output to the console. You don't need to include `results_p.txt` for this part.

```
--- p = 0.7 ---
LasVegas Algorithm With p = 0.7, k = 0
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.7, k = 1
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: p

LasVegas Algorithm With p = 0.7, k = 2
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.7, k = 3
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

--- p = 0.8 ---
LasVegas Algorithm With p = 0.8, k = 0
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

```

```
LasVegas Algorithm With p = 0.8, k = 1
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.8, k = 2
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.8, k = 3
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

--- p = 0.85 ---
LasVegas Algorithm With p = 0.85, k = 0
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.85, k = 1
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.85, k = 2
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y

LasVegas Algorithm With p = 0.85, k = 3
Number of successful tours: x
Number of trials: 100000
Probability of a successful tour: y
```

Where x values are empirical data you get and y values are (x/100000). After obtaining the estimated probability of successful tours, you should fill the appropriate tables and answer the questions in the answer sheet under the heading “Part 2”.

5 Part 3

To handle this section, you should adhere to the steps outlined in PART3 of the Answer Sheet. It is not necessary to complete tables for the executions pertaining to this part.

6 Implementation Guidelines

Your program should receive 1 command line argument which will be either `part1` or `part2`. When the argument is `part1`, the program should generate the output described under Section 3: Part 1. If it is `part2`, the program should generate the output described under Section 4: Part 2.

7 Notes

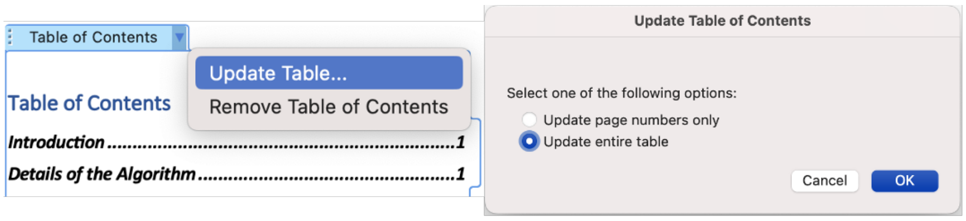
- This project is designed to be completed by a pair of two students.

8 Deadline

- Sunday 14 January 2024 23:59. **Deadline is strict.**
- No late submission will be accepted.

9 Submission

- The answers of all the questions must be collected into the Word answer sheet. Please follow the headings, and type your writings under the appropriate heading.
- Please update the table of contents after your report is ready (Click on the arrow on the top of the contents table, then select Update Entire Table).



Update Table of Contents

Select one of the following options:

Update page numbers only [✓ Update entire table]

- Prepare the answers using a word processor, not in handwritten.
- In addition to the project files, each student in a group must submit a document stating explicitly the parts of the project she/he has worked on. The document must begin with the line "I worked on the following parts in this project:" and all the parts the student has worked on

must be explained very clearly and in not less than 5 lines. Do not write general comments such as "we worked on the project together", and the like. Write what you have done in the project explicitly. If both students write the same or similar explanations, their projects will not be graded.

- You should also submit the program code.
- Each student in the group should upload a single zip file on Moodle, which consists of the pdf file of the answers, `results_07.txt`, `results_07.txt`, and `results_085.txt` for Part 1, a single file that is the program code and the document that explains the parts of the project you worked on. If you want, you can also include a Makefile in the zip.
  Name the files as follows:
  - Answer Sheet:
    `StudentNumber.pdf`
  - Part 1's output files:
    - `StudentNumber_results_07.txt`
    - `StudentNumber_results_08.txt`
    - `StudentNumber_results_085.txt`
  - Program code:
    `StudentNumber.xxx`,
    (xxx: depending on the language)
  - Readme file that explains how to run the code:
    `Readme.txt`
  - Document:
    `StudentNumber_WorkDone.pdf`
  - Zip file that includes all files:
    `StudentNumber.zip`
- **Each group must answer the questions themselves, without any interactions with others. No materials from resources (internet, books, etc.) are allowed to be used.**