The Knight's Tour Problem

Jien Wei, Lim

registration: 100276935

1 Aim

Create a program that solves the Knight's Tour problem starting on any square of a chessboard as well as displaying the tour on a graphical interface.

2 Introduction

In chess, the knight piece has a unique movement on the board compared to the rest of the pieces. With its L-shaped movement (2 squares horizontally/vertically and 1 square vertically/horizontally), the knight is able to jump over pieces in order to reach its destination or remove an opponent's piece. Something conveniently told "not possible" by a relative to 8-year-old me who was playing against them for the first time.

This unique way of moving led to the Knight's Tour which is a classic mathematical problem in graph theory, with the earliest known reference dating back to 9th century AD when Vikings were raiding Europe (Need to cite). It has been pondered by many mathematicians such as Leonhard Euler until it was finally solved in 1823. Even then, many people continue to research and find ways to solve the problem using different methods. Our aim is to create a program that solves the Knight's Tour problem starting on any square of a chessboard as well as displaying the tour on a graphical interface.

But what *is* the Knight's Tour? It is a sequence of moves made by a knight chess piece such that it visits every square of a chessboard exactly once. There are 2 kinds of tours that can be created from the solution: a closed tour, and an open tour. Closed if the knight lands on the final square that is one move away from the starting square to be able to make a another tour again, and open if otherwise.

3 Research & Methodologies

3.1 Research

There are different approaches when it comes to finding a Knight's Tour, some are algorithms, some involving Neural Networks, whereas others are heuristics such as Warnsdorff's Rule, which states that the knight moves such that it always lands on the square which it will have the fewest onward moves.

By looking at the Knight's Tour as a graph, it can be seen that it is an example of the more common Hamiltonian Path problem in graph theory, with the closed variation of the tour being similar to the Hamiltonian Cycle. This is because the Hamiltonian Path/Cycle problem involves a path/loop on a graph that visits each vertex exactly once. Seeing the similarities? This is proven to be correct by Conrad et al. (1994).

According to Parberry (2020), there are 3 methods that can be used to construct the tour; the random walk, neural network, and divide-and-conquer. While random walk

and neural network can create different tours each time, the run time is exponential, $O(C^N)$. This is the opposite for divide-and-conquer as it has a linear run time $O(N)$, but it can only generate the same tours each time.

3.2 Methodology

As the base of this program, the knight's tour will be using the random walk technique in Python. A brute force method to construct the tour will be implemented by using a recursive Depth-first Search algorithm that uses backtracking. The algorithm works checking whether the square the knight is on contains a non-traversed square that the knight can reach, and if there is, the knight will move to that square and will continue until it reaches the final square or one that does not have a free square to land on. If the knight is on a square that does not have a reachable non-traversed square, the knight will go back to the previous square and choose another. The time complexity of this solution is $O(n^2)$ as there are $n \times n$ number of cells. The general operation of the algorithm will go something like this:

```
While tour is not found :-
        If all squares are visited :-
            Show solution and stop searching.
        Else :-
            If square leads to a dead end :-
                Go back to previous square and try another.
            Else :-
                Go to the next square recursively.
```

Once the algorithm is considered working properly, I will move on to creating the graphics to display the tour. The display will show the knight traversing around the chessboard at each move which will also display the move number on each square that it has landed on.

4 Risks and Contingencies

In terms of technical risks, there are 2 issues that I believe may run into when working on the project; memory and graphics.

As the tour will be constructed recursively, it is possible that stack overflow may occur. At best, the program may simply fail and be halted by the operating system. At worst, the computer may shut down and something breaks physically. Python is used to prevent this, as the interpreter limits the number of times a function can call itself recursively unless the limit is changed.

Regarding the graphics, although my GPU should be able to handle it, there is still a chance that the program will be using a high amount of GPU memory. In turn, this could lead to the computer operating at higher temperatures and damaging hardware. With my laptop slowly having the operational capabilities of a toaster by the day, this is something I have to consider. The best I can do to mitigate this would be to minimise the assets used and not include unnecessary objects in the GUI.

5 Work Plan

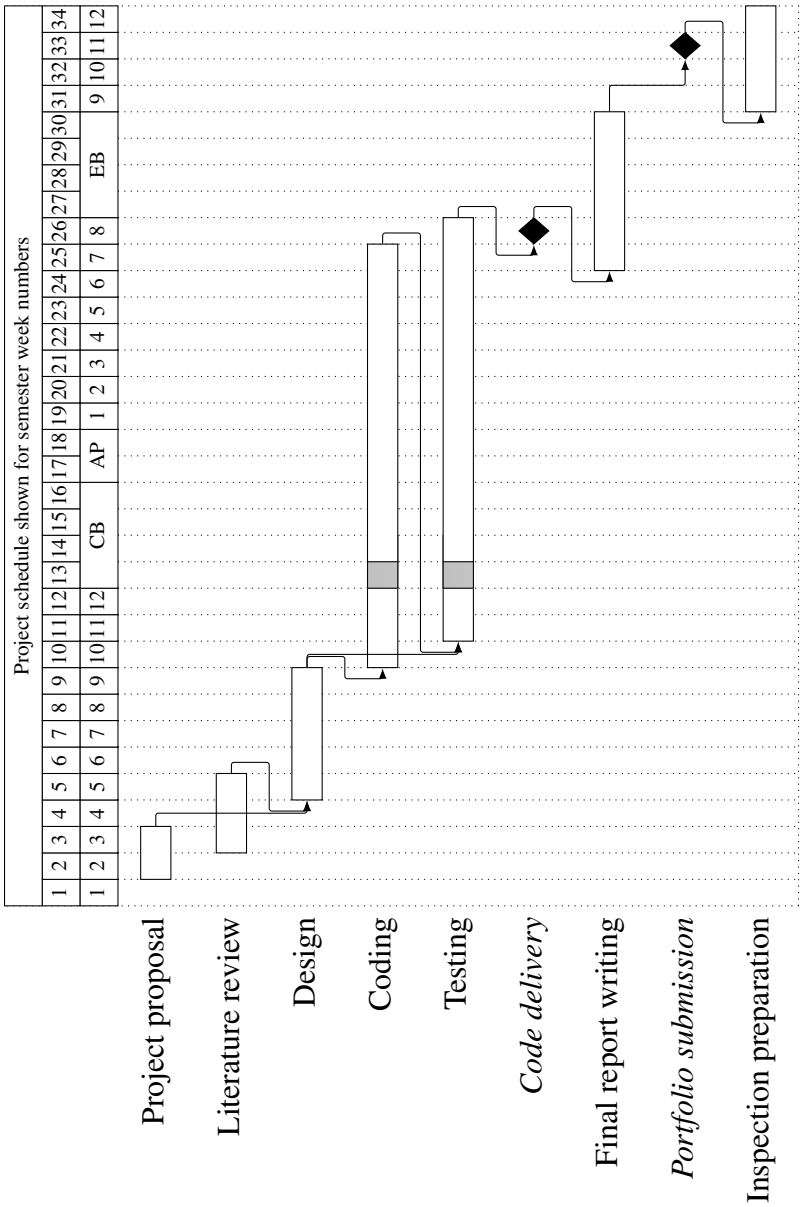

Figure 1: Project Gantt chart

References

Conrad, A., Hindrichs, T., Morsy, H., and Wegener, I. (1994). Solution of the knight's hamiltonian path problem on chessboards. *Discret. Appl. Math.*, 50(2):125–134.

Parberry, I. (2020). Tourneys and the fast generation and obfuscation of closed knight's tours. *CoRR*, abs/2001.06044.

Project proposal

Description of project: aims, motivation, understanding of issues, problems	First	2.1	2.2	3	Fail
Resources, references: evidence of preliminary work to identify key resources, initial reading	First	2.1	2.2	3	Fail
Proposed approaches: relevance, suitability, appropriateness	First	2.1	2.2	3	Fail
Risks: identification, suitable contingency planning	First	2.1	2.2	3	Fail

Quality of writing

Clarity, structure correctness of writing	First	2.1	2.2	3	Fail
Presentation conforms to style	First	2.1	2.2	3	Fail

Workplan

Measurable objectives : appropriate, realistic, timely	First	2.1	2.2	3	Fail
Gantt chart: legibility, clarity, feasibility of schedule	First	2.1	2.2	3	Fail

Comments

Supervisor: Dr. Katharina Huber

Markers should circle the appropriate level of performance in each section. Report and evaluation sheet should be collected by the student from the supervisor.