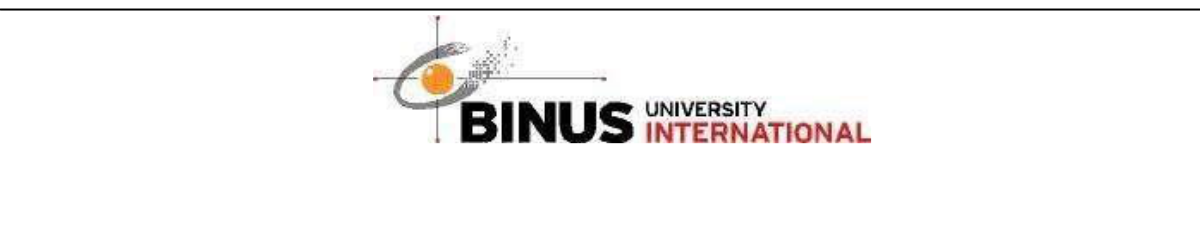
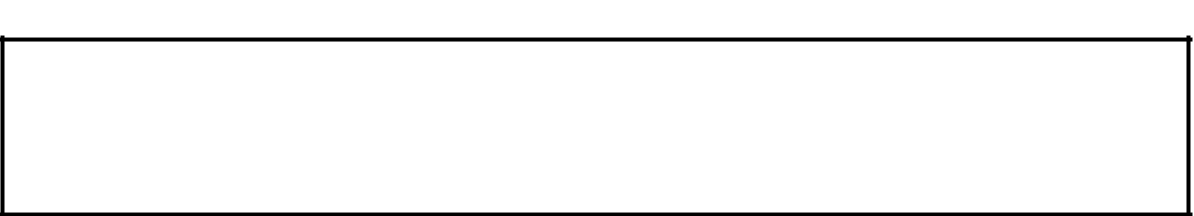
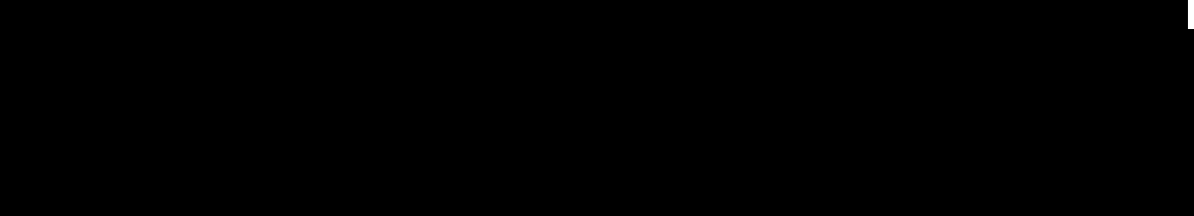
Even Semester (2024)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Individual Work)**

**Student Information**:

**Surname**

**Given Names**

**Student ID Number**

1.

Alvin

Nathaniel

2440042430

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Code** | **:** COMP6571 | **Course Name** | **:** Data Structures and Algorithms |
| **Class** | **:** L2AC | **Name of Lecturer(s) :** MARIA SERAPHINA ASTRIANI, Andreas Kurniawan | |
| **Major** | **:** Computer Science |  |  |

**Title of Assignment** : Knight Travails

**Type of Assignment :** Final Project

**Submission Pattern**

**Due Date** **:**  **Submission Date** **:**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Nathaniel Alvin

**“Knight Travails”**

**Name : Nathaniel Alvin**

**ID** **: 2440042430**

1. **Problem Description**

The problem that I have chosen for this final exam is called the Knight Travails. The problem is as follows. Given is a chess board of size n x n, the initial position of the knight and the final position of the knight. The task that is needed to be solved is to find the minimum number of steps required to reach the final position. The Knight’s move are according to the official chess moves and the knight is not allowed to leave the chess board.

1. **Solution**

My solution to this problem is to use the breadth first search algorithm. First of all, the knight is allowed to move in 8 directions. The situation can be visualized as a graph where the edges will represent the possible moves and the vertices will represent the possible position of the knight. This result to just a common problem of shortest path in an unweighted graph. There are two ways that I can think of to solve this problem. First is using the DFS or the depth first search algorithm.

Since the graph is undirected and connected, there is at least one path between any two vertices of the graph. Therefore it is possible to find the shortest path between any two vertices using the DFS traversal algorithm. The idea is to successively seek for a smaller path from source to destination vertex using the DFS algorithm. We explore all possible path and compare then based on their length. The one with the shortest length is the shortest path between the given vertices.

However, for my solution I think BFS is more intuitive. We try all 8 possible positions where a knight can reach from its position. If reachable position is not already visited and is inside the board, we push this state into queue and increase the number of move. Finally it will return the number of moves to the target position. This approach will always produce the shortest path in an unweighted graph. In BFS, the path is explored level by level. So if the destination is reached, it must be reached through the shortest path.

1. **Theoretical analysis**

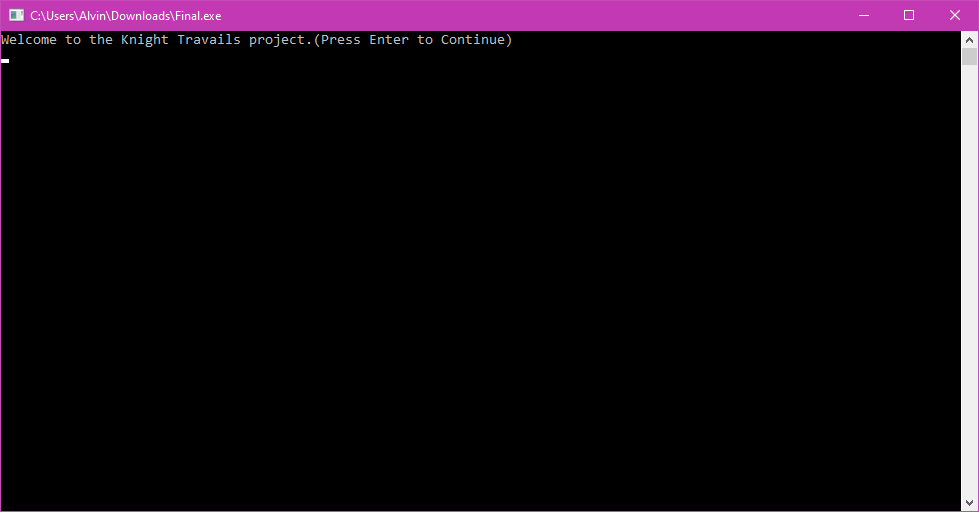
The chessboard is represented as a 2D list in this solution. Therefore, the worst case for this algorithm is to visit all the cells of the board, making the worst-case time complexity as O(N\*\*2).

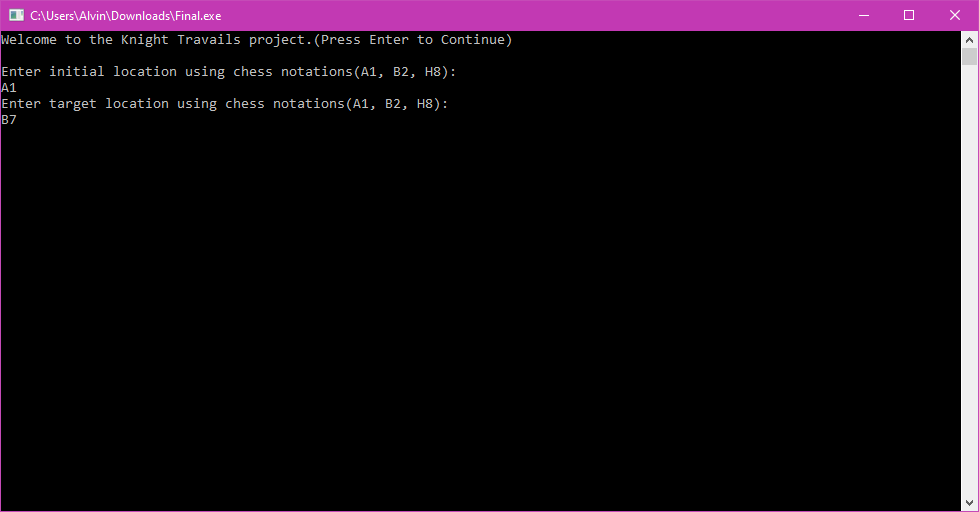
The table below shows the elapsed time for different number of N (size of board). For consistency, the knight target position will always be at N, N and the starting point for the knight will be 1,1.

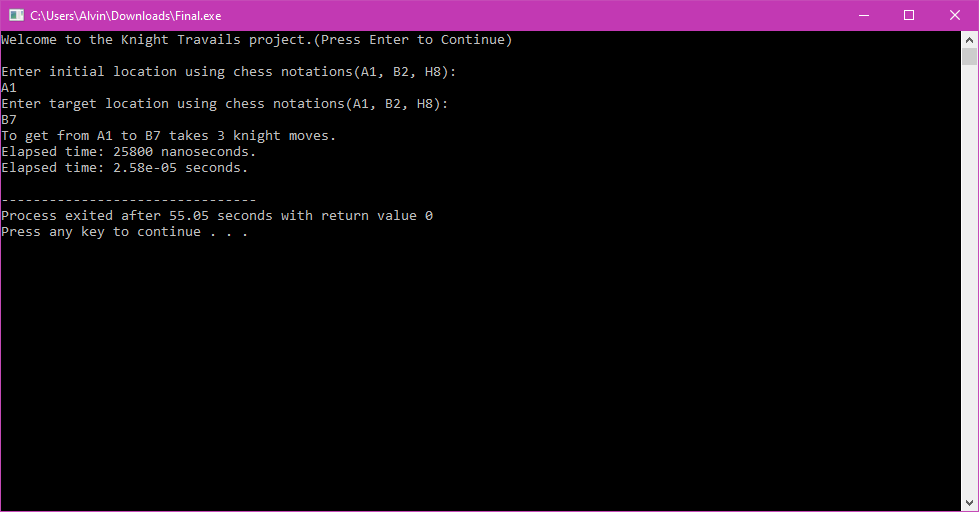
|  |  |
| --- | --- |
| N | Elapsed Time (s) |
| 10 | 0.000020178 |
| 50 | 0.0001795 |
| 100 | 0.0006619 |
| 500 | 0.01616 |
| 1000 | 0.06403 |

1. **Program Manual**

This program is very simple to execute. So for this program, the size of the board is coded to be the size of a normal chess board, 8x8. We can first input the first position of the knight and the last position of the knight in chess notation. For example A1, B7. The program will calculate the number of moves and the time it takes to calculate that in nanoseconds and seconds. This is as shown below in the screenshots.







In this example it took 25800 nanoseconds to calculate the result.

1. **References**

<https://www.geeksforgeeks.org/minimum-steps-reach-target-knight/>

1. **GitHub Link**

https://github.com/nathaniel-alvin/FinalDSA