****

**Lab 1 report: Knight tour problem**

**Course: Introduction to artificial intelligence**

**Theory lecturer: Nguyễn Ngọc Thảo**

**Practice lecturer: Nguyễn Hải Đăng**

**Hồ Thị Thanh Tuyền**

**Lê Ngọc Thanh**

**Student name: Lê Nguyễn Nguyên Anh - 20127438**

**Class: 20CLC09**

# **I/Project Requirement:**

We write program to solve the knight tour problem by applying Backtracking and Warnsdorff’s Heuristic, with sizes include 8, 15, 25 and with each of the sizes we have to get 5 different initial points to get data for experiment.

+For Backtracking:

Size 8: operate all 5 initial points

Sizes 15 and 25: we only run the algorithm for 1-hour-long-run because of the slowness of backtracking

+For Warnsdorff’s Heuristic: run 5 inital points for all board size

# **II/Files:**

Subfolders include:

SOURCE:

+ step folder: include step.h and step.cpp -> represent for knight moves

+ chessBoard folder: include chessBoard.h and chessBoard.cpp -> represent for board where the knight move

+ strategy folder: include backtrack.cpp and heuristic.cpp -> invoke KnightTour with 2 algorithm with backtracking and warnsdorff’s heuristic

+ 20127438.cpp: The program run both backtracking and warnsdorff’s heuristic algorithm in strategy folder at the same time when we call through command line with argument (20128438.exe -px x -py y -s m).

OUTPUT:

+Testing case in txt files include **20127438\_backtrack\_<x>\_<y>\_<m>.txt** and **20127438\_heuristic\_<x>\_<y>\_<m>.txt** where x,y is the inital position and m is the size of chess board. These file will carry information following the format that include values: x, y, m, number of moves (ignore the first value because it is initial value and not cound move backtrack in the backtracking algorithm), time measured (in milliseconds) and board with knight moves.

+output requirement: these file will be generate after we call file named 20127438.exe with parameters (-px x -py y -s m) with format **20127438\_<strategy>.txt** while strategy include backtrack and heuristic

DOCUMENT:

+A pdf file brief about the problem, statistics base on data from running operation of backtracking and heuristics (include both data board and figure), the completeness of the project. This file are named report.pdf

# 

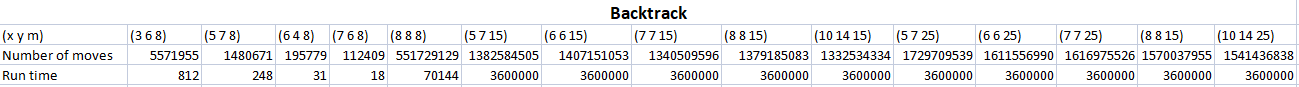
# **III/Criteria:**

|  |  |  |
| --- | --- | --- |
| **#** | **Criteria** | **Status** |
| **1** | Manipulate the input and output | Done |
| **2** | Implement the backtracking | Done |
| **3** | Implement the Warnsdorff’s Heuristic | Done |
| **4** | Provide valid results for backtracking strategy | Done |
| **5** | Provide valid results for Warnsdorff’s Heuristic | Done |
| **6** | Provide all the evidential | Done |
| **7** | Report sufficient information in the document | Done |

# **IV/ S**tatistics and comments**:**

**Backtracking:**

+Database:

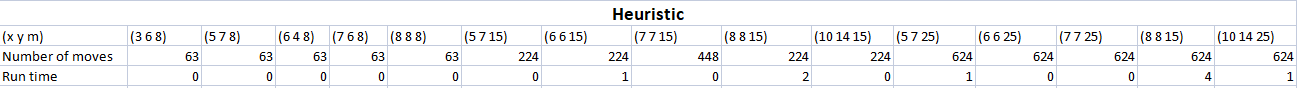


+Chart:

+Comment: an algorithm try exhaustedly for entire possible knight moves since it take giant space and almost infinitely long time. This method is a simpliest but it also the slowest.

**Warnsdorff’s Heuristics:**

+Database:



+Chart:

+Comment: Thanks to heuristic moves, this algorithm solve the problem faster than backtracking because the knight will move rationally (move to pieces that have minimum unvisited adjacent). After the CSP section in class, I notice that this method like Degree Heuristic in the CSP when apply to search problem.Warnsdorff’s Heuristic just need a few milliseconds to find out the solution. There are the wonderful results compared to Backtracking

# **V/References:**

**<https://bradfieldcs.com/algos/graphs/knights-tour/>**

**<https://github.com/wisn/knights-tour/tree/master/outputs>**

**<https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-1/>**

**<https://www.geeksforgeeks.org/warnsdorffs-algorithm-knights-tour-problem/>**