COMP 4106 - Artificial Intelligence
Winter 2016

Assignment #1

Due date: February 22, 2016

The Knight Game

Introduction

In this assignment you will be implementing the *knight game*, a chess-based variation of the well known *snake game*. In the original snake game, a snake moves around a board attempting to eat food without running into itself or the wall. In this variation of the game, the snake is replaced by a chess knight which must capture moving pawns instead of food.

The game is played on a grid board of varying size. The player controls a chess knight which moves in the standard way – two spaces in one direction and one space perpendicular, forming an L (as shown by the black arrow in the diagram). A number of pawns pieces are also present on the board, which are randomly pre-determined (on creation of the game) to move alternatively between two adjacent spaces, as shown in the diagram. Each pawn thus moves from a square A to its adjacent square, B, and then on the next move it moves back to A and so on. Thus, each time the knight makes a move, each of the pawns moves to its other position. The goal of the game is for the knight to capture all of the pawn pieces on the board.

Game Definition

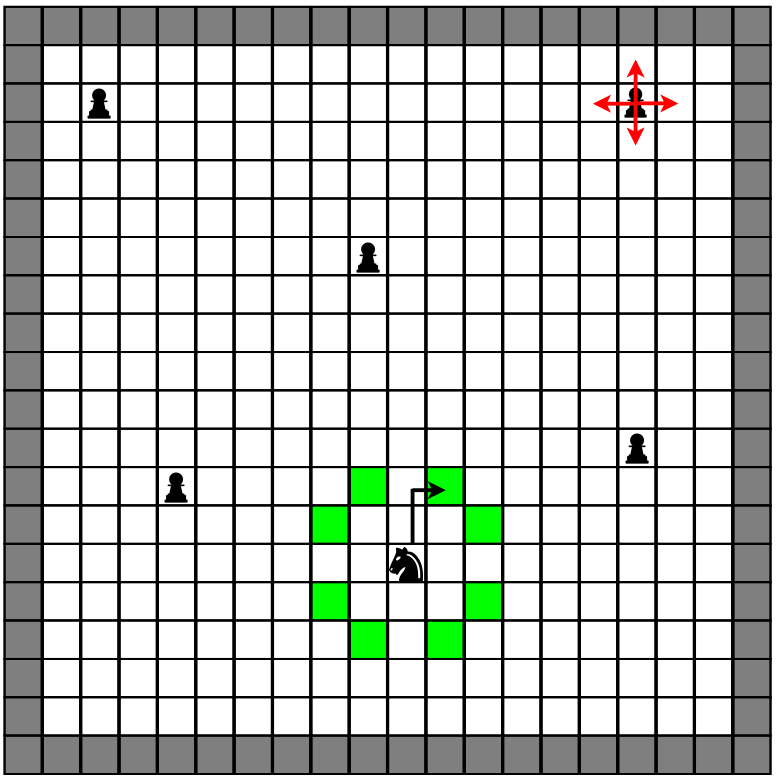

Figure 1: Knight game illustration

The above figure illustrates the game. The green spaces indicate valid places for the knight to move on its next turn. After the knight moves, the pawns move to their other position as indicated by the red arrows. Pieces are not allowed to move diagonally. The grey spaces represent the border of the game. The knight cannot enter the border, and pawns should be configured in a way that they do not enter the border. To capture a pawn, the knight must

move into the space currently occupied by that pawn or have a pawn move into the space currently occupied by the knight. When a pawn is captured, it is removed from the game. The game ends when no pawns remain on the board.

Assignment Objectives

- Implement the *knight game*.
- Implement *Breadth First Search* for the knight to play the game.
- Implement *Depth First Search* for the knight to play the game.
- Implement *A* Search*.
  - Implement two (2) different heuristics for the knight to play the game.
  - Implement a third heuristic which takes the average of the first two heuristics.
- Write a short report (no more than one (1) page) about the state space of the game, and about the choice of your heuristics. Please bring a hard copy of this report with your name and student number to your demo.

Questions

During the demo you should be prepared to discuss the following questions:

- Which search worked best?
- Which heuristics did you use?
- Why did you choose these heuristics?
- Does the combination of the two heuristics work better or worse than they do individually?
- How well do the searches work if you increase the size of the board to 30x30 or 50x50?
- What effect, if any, do scenarios with more/less pawns have on the search algorithms?

Bonus

The following items are considered as bonus. You should work on these if you have completed the required objectives.

- Multiple knights work together to capture the pawns.
- The pawns move in a direction that takes them away from the knight's starting space, instead of moving between just two adjacent spaces.

Tips

Don't spend too much time on the graphics. A command line interface is perfectly acceptable, so long as it is readable. If the search is slow, think about how you can optimize it, but your first priority should be making sure that the search actually works.