1 The Sticks Problem

Consider the following puzzle\(^1\). There are 8 sticks lined up as shown in Fig. 1 (a). By moving exactly 4 sticks, you are to achieve the configuration shown in Fig. 1 (b). Each move consists of picking up a stick, jumping over exactly two other sticks, and then putting it down onto a fourth stick. For example, in Fig.1 (c), stick d could be moved onto stick a or g passing over sticks c and b or e and f, respectively. It could not be moved anywhere else. In Fig. 1 (d) stick d cannot be moved at all. Note that when there is a stick \(x\) on top of another stick \(y\), it is illegal to put another stick \(z\) on top of \(y\).

The search space for this problem is shown on page 4. States which are essentially identical due to symmetry are not duplicated.

Here are your tasks for this tutorial:

1. Decide on a notation for representing states, which you could use in state space search.
2. Using this notation, show the initial state and the goal state.
3. Give the details of all the operators. For each, describe its preconditions, and give an example (i.e. apply the operator to a state and give the new state).
4. Apply the uninformed search algorithms, depth-first and breadth-first search, to the search tree. Is any one obviously the best in reaching the goal? How many nodes do they expand? How many nodes do they keep on the frontier at one time?

\(^1\)Originally set by John Hallam
2 Adversarial Search

This exercise was taken from R&N Chapter 5.

Consider the two-player game shown in Figure 2

1. Draw the complete game tree, using the following conventions:
   - Write each state as \((S_A, S_B)\) where \(S_A\) and \(S_B\) denote token locations
   - Put each terminal state in square boxes and write its game value in a circle
   - Put loop states (states that already appear on the path to the root) in double square boxes. Since it is not clear how to assign values to loop states, annotate each with a “?” in a circle.

2. Now mark each node with its backed-up minimax value (also in a circle). You will have to think of a way to assign values to the loop states.

3. Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to item (2) above. Does your modified algorithm give optimal decisions for all games with loops?
Figure 2: The starting position of a simple game. Player A moves first. The two players take turns moving, and each player must move their token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then the player may jump over the opponent to the next available space. For example, if $A$ is on 3 and $B$ is on 2, then $A$ may move back to 1. The game ends when one player reaches the opposite end of the board. If player $A$ reaches space 4 first, then the value of the game to $A$ is $+1$; if player $B$ reaches space 1 first the value of the game to $A$ is $-1$. 
Figure 3: Search tree for the Sticks Problem