HOMEWORK (chapter 5: Adversarial search)

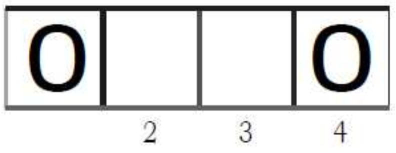


Figure 5.17 The starting position of a simple game. Player *A* moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (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 *.3* reaches space 1 first, then the value of the game to A is -1.

Q-1: Consider the two-player game described in Figure 5.17.

1. Define the game by specifying the initial state, the legal actions in each state, the result of each action, a terminal test, and a utility function.

2. Draw the complete game tree, using the following conventions:

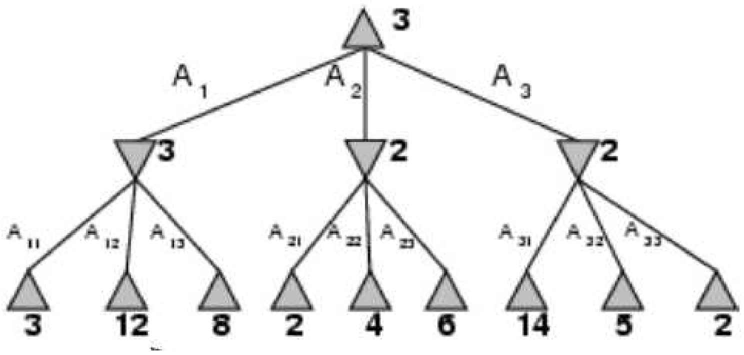
    - Write each state as (sA, sB), where sA and sB denote the token locations.
    - Put each terminal state in a square box and write its game value in a circle.
    - Put loop states (states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate each with a "?" in a circle.

3. Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why.

4. 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 (3). Does your modified algo-rithm give optimal decisions for all games with loops?

5. This 4-square game can be generalized to n. squares for any n > 2. Prove that A wins if n is even and loscs if it is odd.

Q-2: Apply alpha-beta search to the following game tree, assuming that actions are ordered from right-to-left at each node. What parts of the tree can be pruned?


3
A 1
A 2
A 3
3
2
2
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2