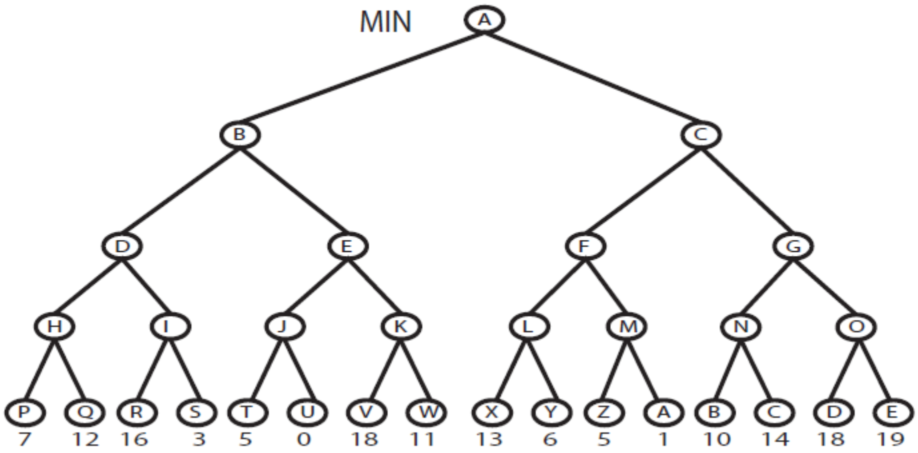
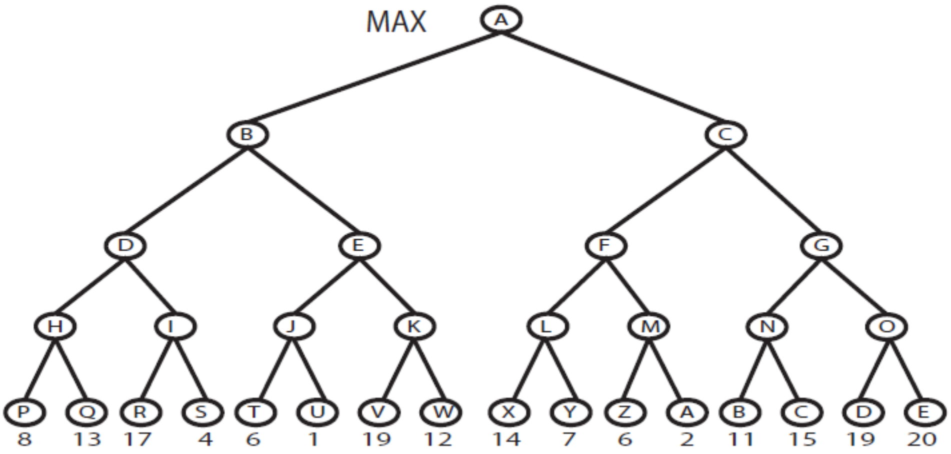
1 GamePlaying

1. Apply the minimax algorithm to the game trees below.

2. Apply the minimax algorithm with alpha-beta pruning to the game trees below.

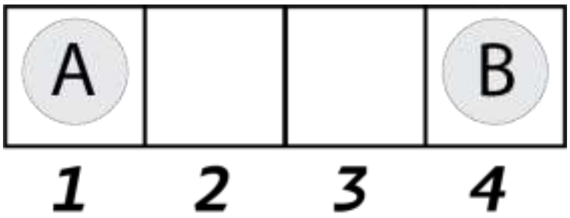




2 LineGame

Consider the two-player game described below:

Figure 1: The starting position of a simple game.



The starting position is given in Figure 1. Player A moves first. The two players take turns moving. Each player must move her 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 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 B reaches space 1 first, then the value of the game to A is −1.

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

3 TicTacToe

This problem exercises the basic concepts of game playing, using **tic-tac-toe** (noughts and crosses) as an example. We define X_n as the number of rows, columns, or diagonals with exactly n X's and no O's. Similarly, O_n is the number of rows, columns, or diagonals with just n O's. The utility function assigns +1 to any position with $X_3 = 1$ and −1 to any position with $O_3 = 1$. All other terminal positions have utility 0. For nonterminal positions, we use a linear evaluation function defined as $\mathrm{Eval}(s) = 3X_2(s)+X_1(s)-(3O_2(s)+O_1(s))$.

1. Approximately how many possible games of tic-tac-toe are there?
2. Show the whole game tree starting from an empty board down to depth 2 (i.e., one X and one O on the board), taking symmetry into account.
3. Mark on your tree the evaluations of all the positions at depth 2.
4. Using the minimax algorithm, mark on your tree the backed-up values for the positions at depths 1 and 0; and use those values to choose the best starting move.
5. Circle the nodes at depth 2 that would not be evaluated if alpha–beta pruning were applied, assuming the nodes are generated in the optimal order for alpha–beta pruning.