National University of Computer & Emerging Sciences

AL2002 – Artificial Intelligence – Lab (Spring 2024)

BSCS-6B

Lab Work 8 (Minimax, Alpha Beta Pruning)

Lab Instructor	Momna Javaid
Department	Computer Science

Instructions:

1. You also have to submit .ipynb file.
2. Comments in the code explaining chunks of the code are important.
3. Plagiarism is strictly prohibited, 0 marks would be given to students who cheat.

Lab Tasks:

Task 1:

Minimax is a method used to evaluate game trees. Implement an algorithm for calculating the optimal move using Minimax—the move that leads to a terminal state with maximum utility, under the assumption that the opponent plays in a 2 player game.

MAX

- Wants to maximize the result of the utility function
- Winning strategy if, on MIN's turn, a win is obtainable for MAX for all moves that MIN can make

MIN

- Wants to minimize the result of the utility function
- Winning strategy if, on MAX's turn, a win is obtainable for MIN for all moves that MAX can make

Max = -∞

Min = +∞

Calculate the following:

- Complete Path
- Time Complexity
- Space Complexity

The pseudocode of MINIMAX is:

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))
```

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v
```

```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

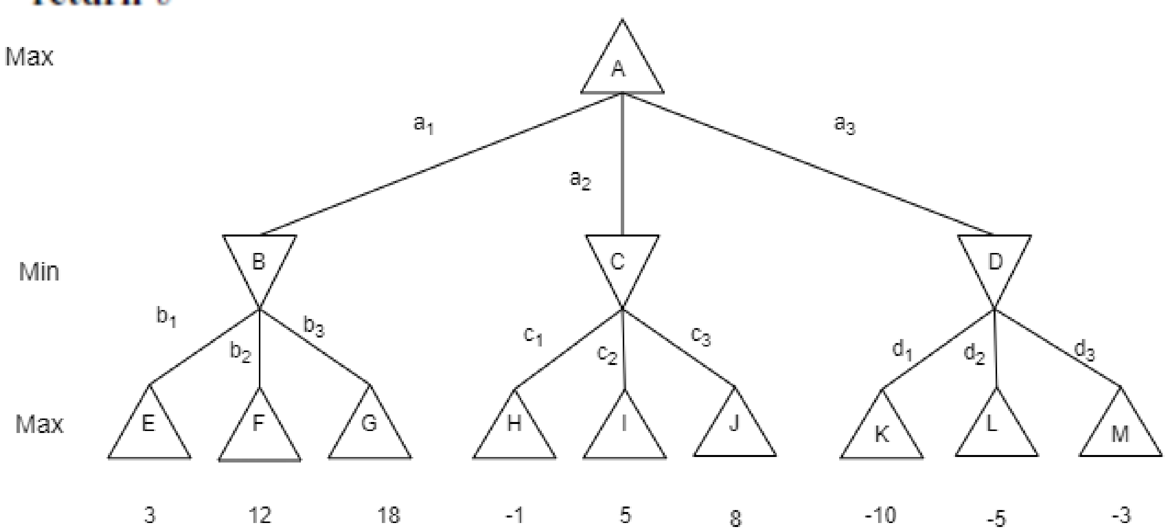

Figure 1. Minimax Tree

Task 2:

Exhaustively searching a game tree is not usually a good idea. We can use a branch-and-bound technique to reduce the no. of states that must be examined to determine the value of a tree.

Branch-and-bound Technique:

- We keep track of a lower bound on the value of a maximizing node, and don't bother evaluating any trees that cannot improve this bound.
- Keep track of an upper bound on the value of a minimizing node. Don't bother with any sub-trees that cannot improve this bound.

To improve the task 1 you should implement MiniMax with Alpha-Beta pruning (prunes away search tree nodes that are not necessary for finding an optimal solution) in a two player game. Use the same graph as in task 1 for this task.

- At minimizing nodes, we stop evaluating children if we get a child whose value is less than the current lower bound **(alpha)**.
- At maximizing nodes, we stop evaluating children as soon as we get a child whose value is greater than the current upper bound **(beta)**.

Max = $\alpha = -\infty$
Min = $\beta = +\infty$

$\alpha \geq \beta$

Calculate the following:

- Complete Path
- Time Complexity
- Space Complexity