Min-Max

- In the case where two agents are competing so that a positive reward for one is a negative reward for the other agent, we have a two-agent **zero-sum game**.
- The value of such a game can be characterized by a single number that one agent is trying to maximize and the other agent is trying to minimize.
- Having a single value for a two-agent zero-sum game leads to a **minimax** strategy.
- Each node is either a MAX node, if it is controlled by the agent trying to maximize, or is a MIN node if it is controlled by the agent trying to minimize.
- Backward induction could be used to find the optimal minimax strategy.
- From the bottom up, backward induction maximizes at MAX nodes and minimizes at MIN nodes.
- However, backward induction requires a traversal of the whole game tree.
- It is possible to prune part of the search tree by showing that some part of the tree will never be part of an optimal play.

Alpha-Beta Pruning

Suppose the values of the leaf nodes are given or are computed given the definition of the game. The numbers at the bottom show some of these values. The other values are irrelevant.

Suppose we are doing a left-first depth-first traversal of this tree.

The value of node h is 7, because it is the minimum of 7 and 9.

Just by considering the leftmost child of I with a value of 6, we know that the value of i is less than or equal to 6.

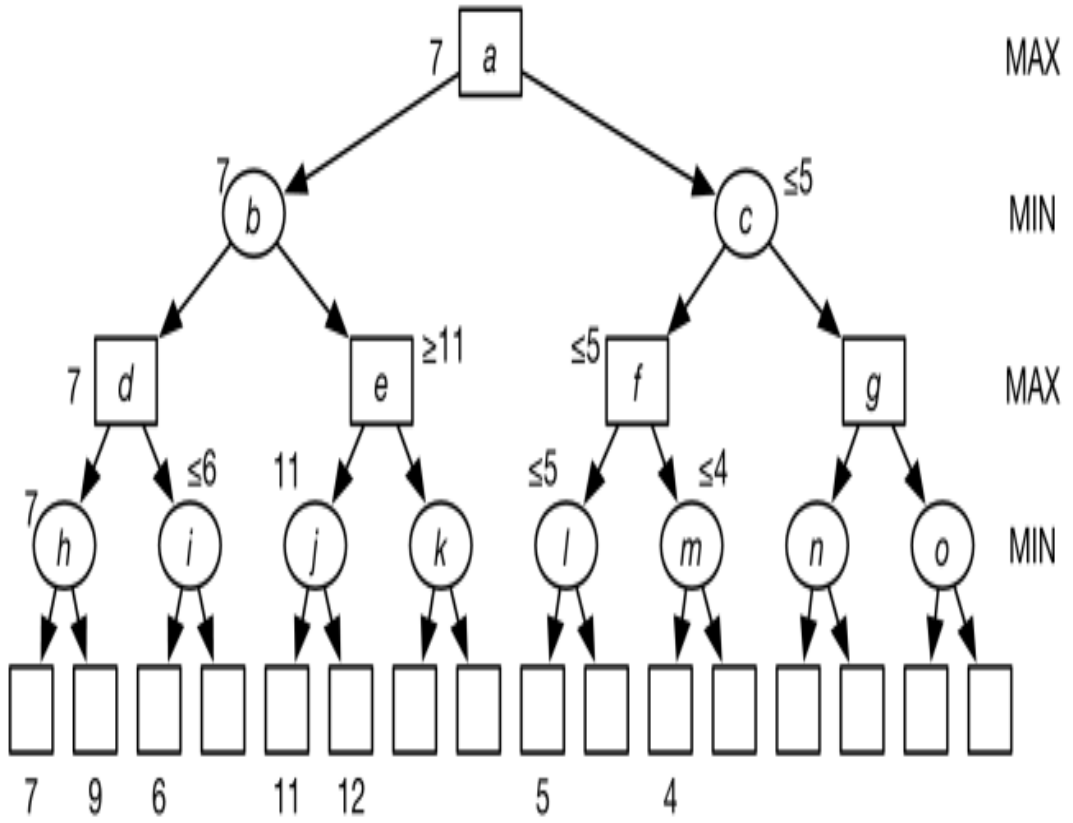
Therefore, at node d, the maximizing agent will go left. We do not have to evaluate the other child of i.

Similarly, the value of j is 11, so the value of e is at least 11, and so the minimizing agent at node b will choose to go left.

The value of l is less than or equal to 5, and the value of m is less than or equal to 4; thus, the value of f is less than or equal to 5, so the value of c will be less than or equal to 5.

So, at a, the maximizing agent will choose to go left.

this argument did not depend on the values of the unnumbered leaves. Moreover, it did not depend on the size of the subtrees that were not explored.



Minimax with alpha-beta (α-β) pruning

- Minimax with **alpha-beta (α-β) pruning** is a depth-first **search** algorithm that prunes by passing pruning information down in terms of parameters α and β. In this depth-first search, a node has a score, which has been obtained from (some of) its descendants.
- The parameter α is used to prune MIN nodes. Initially, it is the highest current value for all MAX ancestors of the current node. Any MIN node whose current value is less than or equal to its α value does not have to be explored further. This cutoff was used to prune the other descendants of nodes l, m, and c in the previous example.
- The dual β parameter is used to prune MAX nodes.

MiniMax Procedure

```
procedure Minimax_alpha_beta( )
    Inputs :  n  a node in a game tree ; α, β real numbers;
   Output :  A pair of a value for node , path that gives this value
    best := None
   if n  is a leaf node then
      return evaluate (n), None
   else if n is a MAX node then
     for each child c  of n do
          score, path := MinimaxAlphaBeta (c, α, β)
           if score ≥ β  then
              return score, None
          else if score > α  then
               α := score
               best := c : path
     return α, best
else
  for each child c  of n do
          score, path := MinimaxAlphaBeta (c, α, β)
           if score ≤ α then
              return score, None
          else if score < β then
               β  := score
               best := c : path
       return β, best
```

---

called initially with

Minimax_alpha_beta(R, - ∞, ∞)

where R is the root node. It returns a pair of the value for the node n and a path of choices that is best for each agent the results in this path

Alpha-Beta