LAB # 9:

MINIMAX WITH ALPHA-BETA PRUNING

Objectives:

- To implement minimax searching and minimax with alpha-beta pruning in python

Hardware/Software Required:

Hardware: Desktop/ Notebook Computer
Software Tool: Python 2.7/ 3.6.2

Introduction:

Minimax:

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. When dealing with gains, it is referred to as "maximin"—to maximize the minimum gain. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision-making in the presence of uncertainty.

The pseudocode for minimax searching is:

```
01 function minimax(node, depth, maximizingPlayer)
02     if depth = 0 or node is a terminal node
03         return the heuristic value of node

04     if maximizingPlayer
05         bestValue := −∞
06         for each child of node
07             v := minimax(child, depth − 1, FALSE)
08             bestValue := max(bestValue, v)
09         return bestValue

10     else    (* minimizing player *)
11         bestValue := +∞
12         for each child of node
13             v := minimax(child, depth − 1, TRUE)
14             bestValue := min(bestValue, v)
15         return bestValue

minimax(origin, depth, TRUE)
```

Minimax with Alpha-Beta Pruning:

Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc.). It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

The pseudocode for the minimax with alpha-beta pruning is:

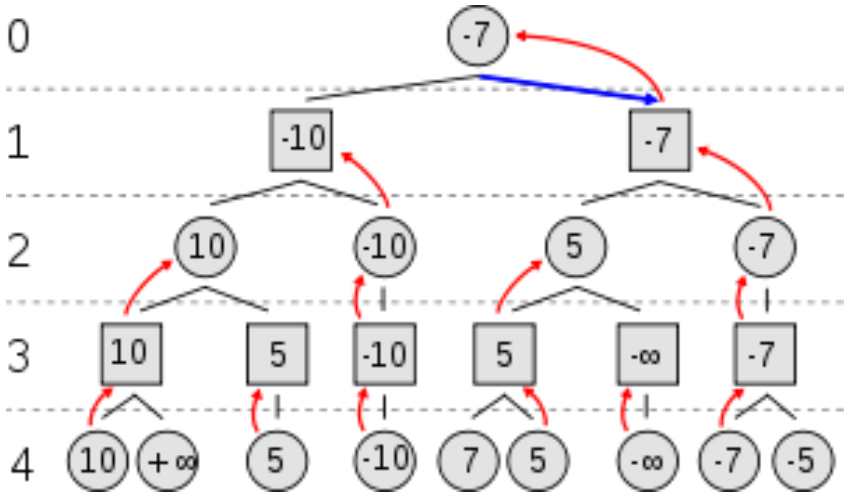
```
01 function alphabeta(node, depth, α, β, maximizingPlayer)
02     if depth = 0 or node is a terminal node
03         return the heuristic value of node
04     if maximizingPlayer
05         v := -∞
06         for each child of node
07             v := max(v, alphabeta(child, depth – 1, α, β, FALSE))
08             α := max(α, v)
09             if β ≤ α
10                 break (* β cut-off *)
11         return v
12     else
13         v := +∞
14         for each child of node
15             v := min(v, alphabeta(child, depth – 1, α, β, TRUE))
16             β := min(β, v)
17             if β ≤ α
18                 break (* α cut-off *)
19         return v

alphabeta(origin, depth, -∞, +∞, TRUE)
```

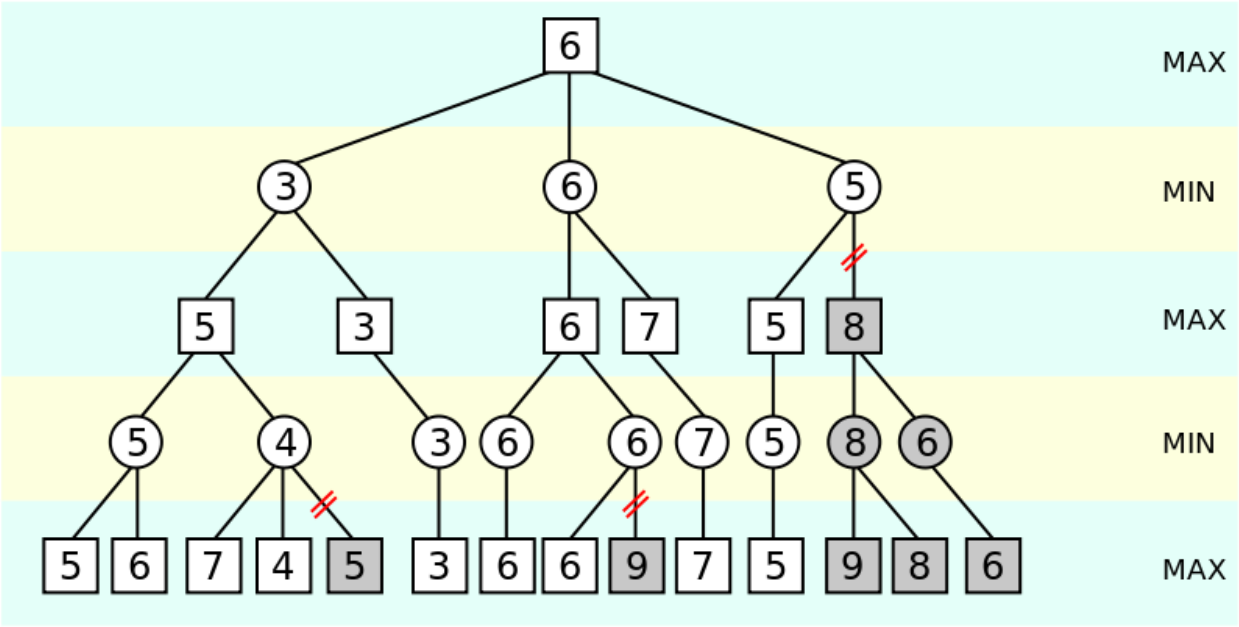
Lab Task:

1) Implement minimax algorithm for Tree 1 and Tree 2. Import the time module and calculate the total time taken by your minimax algorithm.

2) Implement minimax with alpha-beta pruning for both Tree 1 and Tree 2. Compare the execution time of Question # 2 with Question # 1.



Tree 1



Tree 2

Conclusion:
Write the conclusion about this lab

NOTE: A lab journal is expected to be submitted for this lab.