Alpha Beta Pruning:

# Question:

From figure 1 determine alpha beta pruning

|  |  |  |
| --- | --- | --- |
| 3 | 6 | 9 |
| 1 | -4 | -5 |
| 8 | 4 | 2 |
| -3 | 7 | 3 |
| 5 | 1 | 2 |
| 3 | 4 | 5 |

Fig 1.

# Python Code:

tree = [[[3, 6, 9], [1, -4, -5]], [[8, 4, 2], [-3, 7, 3], [5, 1, 2], [3, 4, 5]]]

root = 0

pruned = 0

def children(branch, depth, alpha, beta):

global tree

global root

global pruned

i = 0

for child in branch:

if type(child) is list:

(nalpha, nbeta) = children(child, depth + 1, alpha, beta)

if depth % 2 == 1:

beta = nalpha if nalpha < beta else beta

else:

alpha = nbeta if nbeta > alpha else alpha

branch[i] = alpha if depth % 2 == 0 else beta

i += 1

else:

if depth % 2 == 0 and alpha < child:

alpha = child

if depth % 2 == 1 and beta > child:

beta = child

if alpha >= beta:

pruned += 1

break

if depth == root:

tree = alpha if root == 0 else beta

return (alpha, beta)

def alphabeta(in\_tree=tree, start=root, upper=-15, lower=15):

global tree

global pruned

global root

(alpha, beta) = children(tree, start, upper, lower)

if \_\_name\_\_ == "\_\_main\_\_":

print ("(alpha, beta): ", alpha, beta)

print ("Result: ", tree)

print ("Times pruned: ", pruned)

return (alpha, beta, tree, pruned)

if \_\_name\_\_ == "\_\_main\_\_":

alphabeta(None)

# OUTPUT:

(alpha, beta): 5 15

Result: 5

Times pruned: 1

## Screen Shot:

