

Date: 17/12/24

Program Title: Implement Alpha-Beta Pruning.

Algorithm:

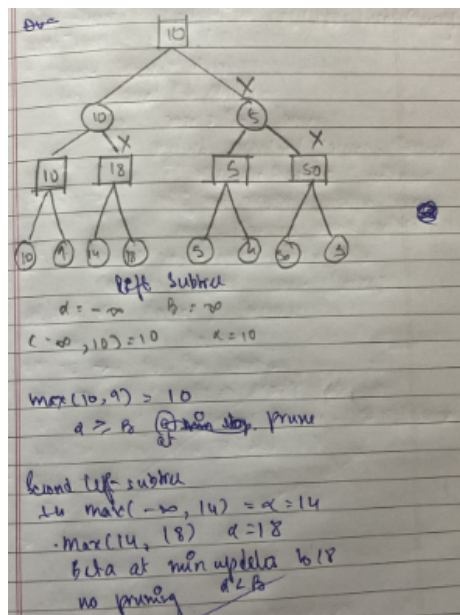
ALGORITHM:
Function ALPHA-BETA-SEARCH(state) returns an opt
val MAX-VALUE(state, $-\infty$, $+\infty$)
return the action in ACTIONS with value v

Function MAX-VALUE(state, α , β) returns utility
value
if TERMINAL-TEST(state) then return
UTILITY(state)
 $v \leftarrow -\infty$
for each a in ACTIONS(state) do
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{state}$
RESULT(s, a), $\alpha, \beta))$
if $v \geq \beta$ then return β
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

Function MIN-VALUE(state, α , β) returns a
utility value
if TERMINAL-TEST(state) then return
UTILITY(state)
 $v \leftarrow +\infty$
for each a in ACTIONS(state) do
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{state}$
RESULT(s, a), $\alpha, \beta))$
if $v \leq \alpha$ then return v
if $\beta \leq \text{MIN}(\beta, v)$

return v

Tree:



Code:

```
import math
def alpha_beta_pruning(depth, nodeIndex, isMax, values, alpha, beta):
    if depth == 3:
        return values[nodeIndex]
    if isMax:
        best = -math.inf
        for i in range(0, 2):
            val = alpha_beta_pruning(depth + 1, nodeIndex * 2 + i, False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break
        return best
    else:
        best = math.inf
        for i in range(0, 2):
            val = alpha_beta_pruning(depth + 1, nodeIndex * 2 + i, True, values, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)
            if beta <= alpha:
                break
        return best
values = [10, 9, 14, 18, 5, 4, 50, 3]
alpha = -math.inf
beta = math.inf
result = alpha_beta_pruning(0, 0, True, values, alpha, beta)
print("Optimal Value:", result)
```

import math

def alpha_beta_pruning(depth, nodeIndex, isMax, values, alpha, beta):

if depth == 3:

```

        return values[nodeIndex]
    if isMax:
        best = -math.inf
        for i in range(0, 2):
            val = alpha_beta_pruning(depth + 1, nodeIndex * 2 + i, False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break
        return best
    else:
        best = math.inf
        for i in range(0, 2):
            val = alpha_beta_pruning(depth + 1, nodeIndex * 2 + i, True, values, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)
            if beta <= alpha:
                break
        return best
values = [10, 9, 14, 18, 5, 4, 50, 3]
alpha = -math.inf
beta = math.inf
result = alpha_beta_pruning(0, 0, True, values, alpha, beta)
print("Optimal Value:", result)

```

Snapshot of the Result:

