120CS0124_AILab7

September 26, 2023

[25]:
```python
# Maloth Aditya
# 120CS0124
# 26 September
# Alpha-Beta Pruning

class Node:
    def __init__(self,name=None,value=None,children=None):
        self.name = name
        self.value = value
        self.children = children

    def isLeaf(self):
        #Checks if a node is a leaf node
        return not self.children


def maxValue(node,alpha,beta):

    if node.isLeaf():
        return node.value
    value = -float("inf")
    for child in node.children:
        value = max(value,minValue(child,alpha,beta))
        if value >= beta:
            print(f"{node.name}:\t alpha: {alpha}, \tbeta: {beta}")
            return value
        alpha = max(alpha,value)

    print(f"{node.name}:\t alpha: {alpha}, \tbeta: {beta}")
    return value

def minValue(node,alpha,beta):

    if node.isLeaf():
        return node.value
    value = float("inf")
    for child in node.children:
```

```
        value = min(value,maxValue(child,alpha,beta))
        if value <= alpha:
            print(f"{node.name}:\t alpha: {alpha}, \tbeta: {beta}")
            return value
        beta = min(beta,value)

    print(f"{node.name}:\t alpha: {alpha}, \tbeta: {beta}")
    return value

def alphaBetaSearch(node):
    return maxValue(node,-float("inf"),float("inf"))


# main code
tree =␣
 ↪Node('A',children=[Node('B',children=[Node('D',children=[Node('H',children=[Node(value=3),Nod

optimalValue = alphaBetaSearch(tree)
print(f"Value: {optimalValue}")
```

```
H:      alpha: -inf,    beta: 3
I:      alpha: 3,       beta: inf
D:      alpha: 3,       beta: inf
J:      alpha: -inf,    beta: 3
E:      alpha: -inf,    beta: 3
B:      alpha: -inf,    beta: 3
L:      alpha: 3,       beta: inf
M:      alpha: 3,       beta: inf
F:      alpha: 3,       beta: inf
C:      alpha: 3,       beta: inf
A:      alpha: 3,       beta: inf
Value: 3
```

[ ]:

[ ]: