Castillo, Anjelica M.
MCSCC 123-MSCS12S1

Intelligent Agents Adversarial Search

The **Minimax Algorithm** is a decision-making method in game theory and AI for optimal play, while **Alpha-Beta Pruning** optimizes it by eliminating branches that won't influence the final decision.

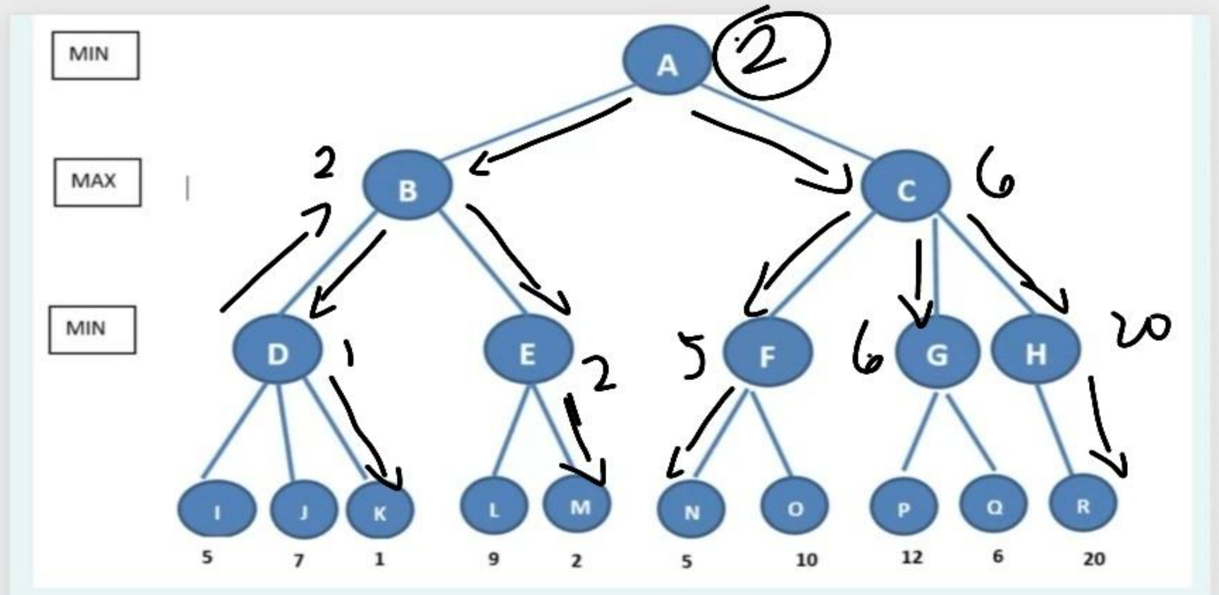


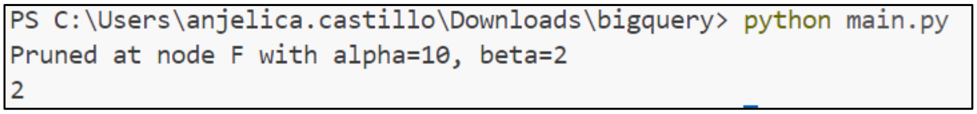
Figure 1

- **D's children (MIN level)** → {5, 7, 1} → **MIN = 1**
- **E's children (MIN level)** → {9, 2} → **MIN = 2**
- **F's child (MIN level)** → {10, 5} → **MIN = 5**
- **G's child (MIN level)** → {12,6} → **MIN = 6**
- **H's children (MIN level)** → {20} → **MIN = 20**
- **B's children D, E(MAX level)** → {1,2} → **MAX= 2**
- **C's children F, G, H (MAX level)** → {5,6,20} → **MAX = 20**
- **A's children B,C(MAX level)** → {2,20} → **MIN= 2**

Castillo, Anjelica M.
MCSCC 123-MSCS12S1

This implementation follows the Minimax algorithm with Alpha-Beta pruning.

```python
import math

def minimax(node, depth, is_min, alpha, beta, values, tree):

    if node not in tree:  # If it's a leaf node, return its value
        return values[node]

    if is_min:
        min_eval = math.inf
        for child in tree[node]:
            eval = minimax(child, depth + 1, False, alpha, beta, values, tree)
            min_eval = min(min_eval, eval)
            beta = min(beta, eval)
            if beta <= alpha:
                print(f"Pruned at node {child} with alpha={alpha}, beta={beta}")
                break  # Alpha cutoff (Pruning)
        return min_eval
    else:
        max_eval = -math.inf
        for child in tree[node]:
            eval = minimax(child, depth + 1, True, alpha, beta, values, tree)
            max_eval = max(max_eval, eval)
            alpha = max(alpha, eval)
            if beta <= alpha:
                print(f"Pruned at node {child} with alpha={alpha}, beta={beta}")
                break  # Beta cutoff (Pruning)
        return max_eval

# Tree representation
values = {
    'I': 5, 'J': 7, 'K': 1, 'L': 9, 'M': 2, 'N': 5, 'O': 10,
    'P': 12, 'Q': 6, 'R': 20
}

tree = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F', 'G', 'H'],
    'D': ['I', 'J', 'K'],
    'E': ['L', 'M', 'N'],
    'F': ['O'],
    'G': ['P'],
    'H': ['Q', 'R']
}

optimal_value = minimax('A', 0, True, -math.inf, math.inf, values, tree)
print(optimal_value)
```



```
PS C:\Users\anjelica.castillo\Downloads\bigquery> python main.py
Pruned at node F with alpha=10, beta=2
2
```