```python
class Node:
    def __init__(self, value, alpha, beta):
        self.value = value
        self.alpha = alpha
        self.beta = beta
        self.children = []

def min_max(node, maximizing_player):
    if not node.children:
        return node.value

    if maximizing_player:
        max_eval = float('-inf')
        for child in node.children:
            eval = min_max(child, False)
            max_eval = max(max_eval, eval)
        return max_eval
    else:
        min_eval = float('inf')
        for child in node.children:
            eval = min_max(child, True)
            min_eval = min(min_eval, eval)
        return min_eval

def alpha_beta_pruning(node, alpha, beta, maximizing_player):
    if not node.children:
        return node.value

    if maximizing_player:
        max_eval = float('-inf')
        for child in node.children:
            eval = alpha_beta_pruning(child, alpha, beta, False)
            max_eval = max(max_eval, eval)
            alpha = max(alpha, eval)
            if beta <= alpha:
                break
        return max_eval
    else:
        min_eval = float('inf')
        for child in node.children:
            eval = alpha_beta_pruning(child, alpha, beta, True)
            min_eval = min(min_eval, eval)
            beta = min(beta, eval)
            if beta <= alpha:
                break
        return min_eval

root = Node(6, float('-inf'), float('inf'))
```

```
root.children = [Node(3, float('-inf'), 3), Node(2, 3, 2), Node(2, 3, 2)]
root.children[0].children = [Node(3, float('-inf'), 3), Node(12, 3,
float('inf')), Node(8, 3, float('inf'))]
root.children[1].children = [Node(2, float('-inf'), 2), Node(4, 2,
float('inf')), Node(6, 2, float('inf'))]
root.children[2].children = [Node(14, 3, float('inf')), Node(5, 3,
float('inf')), Node(2, 3, float('inf'))]

value_min_max = min_max(root, True)
print(f"a. Giá trị của nút gốc (Min-Max): {value_min_max}")


value_alpha_beta = alpha_beta_pruning(root.children[1], 2, 3, True)
print(f"b. Giá trị alpha-beta của nút A2 sau khi duyệt A21:
{value_alpha_beta}")

value_alpha_beta_root = alpha_beta_pruning(root, float('-inf'), float('inf'),
True)
print(f"c. Giá trị alpha-beta của nút gốc sau khi duyệt cây:
{value_alpha_beta_root}")

branches_cut = 0

def count_cut_branches(node):
    global branches_cut
    if node.alpha >= node.beta:
        branches_cut += 1
    for child in node.children:
        count_cut_branches(child)

count_cut_branches(root)
print(f"d. Số nhánh được cắt khi xác định giá trị của nút gốc:
{branches_cut}")
```

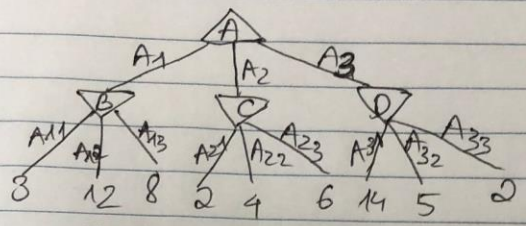
Thứ..... Ngày .....

Phạm Đức Đô - 64CNTT2 - 2251061742

BT.

a, Max

Min



a, Đối với nút B = min (3; 12; 8) = 3

Đối với nút C = min (2; 4; 6) = 2

Đối với nút D = min (14; 5; 2) = 2

⇒ Đối với nút A = max (3; 2; 2) = 3.

b, Xét nút A với $\alpha = -\infty$ và $\beta = +\infty$

→ di chuyển giá trị xuống nút con B.

ta tính được giá trị của β là:

min (3; 12; 8) = 3

⇒ B có giá trị $\alpha = -\infty$ và $\beta = 3$.

⇒ A có giá trị $\alpha = 3$; $\beta = +\infty$.

Di chuyển giá trị xuống nút C.

⇒ C có $\alpha = 3$, $\beta = +\infty$

Xét ngược từ nhánh dưới xác định β là:

min (2; 4; 6) = 2

⇒ C có $\alpha = 3$ và $\beta = 2$

c, Tiếp tục trả câu b.

⇒ di chuyển giả thử nút A xuống D.

⇒ D có $\alpha = 3$ và $\beta = +\infty$

Xét ngược như trên.

⇒ β của D là Min (14; 5; 2) = 2.

⇒ D có $\alpha = 3$ và $\beta = 2$

Từ đây ta xác định được β của A là:

Max (3; 2; 2) = 3

⇒ $\alpha = 2$ và $\beta = 3$. (Gốc)

d, Xét nút C ($\alpha = 3$, $\beta = 2$)

Vì $\alpha \geq \beta$ nên kế thừa phải sẽ loại bỏ.

⇒ Có 2 nhánh bị loại.