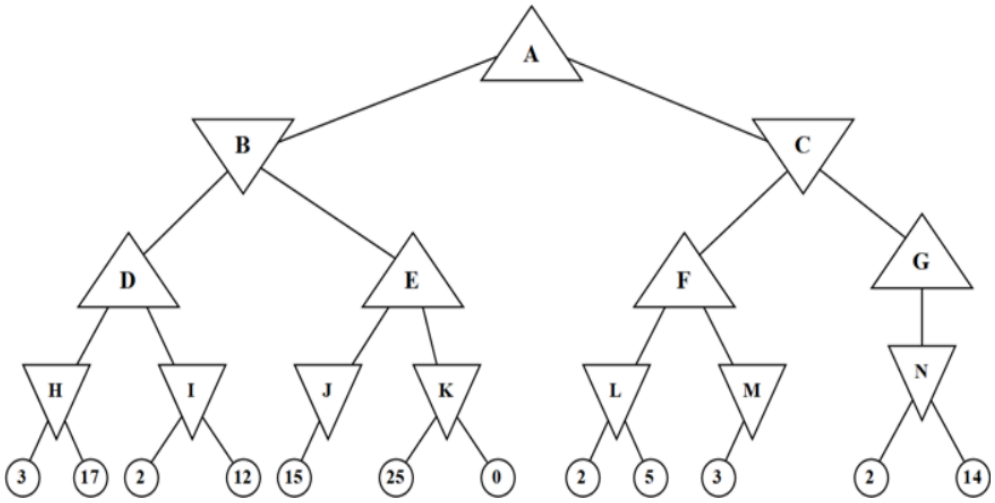
## Assignment 7

Calculate the Min-Max value of each non-leaf node in a given search tree.
Assume 1 ) The MAX plays first at node A, followed by MIN. 2 ) Actions are taken in alphabetical order




## Code:

```
_INF = -float('inf')
INF = float('inf')

node_vals = [
    ['A', 1, None],
    ['B', -1, None],
    ['C', -1, None],
    ['D', 1, None],
    ['E', 1, None],
    ['F', 1, None],
    ['G', 1, None],
    ['H', -1, None],
    ['I', -1, None],
    ['J', -1, None],
    ['K', -1, None],
    ['L', -1, None],
    ['M', -1, None],
    ['N', -1, None],
    ['l1', 0, 3],
    ['l2', 0, 17],
    ['l3', 0, 2],
    ['l4', 0, 12],
    ['l5', 0, 15],
    ['l6', 0, 25],
    ['l7', 0, 8],
    ['l8', 0, 2],
    ['l9', 0, 5],
    ['l10', 0, 3],
```

```
    ['l11', 0, 2],
    ['l12', 0, 14],
]
tree = {
    0: [1, 2],
    1: [3, 4],
    2: [5, 6],
    3: [7, 8],
    4: [9, 10],
    5: [11, 12],
    6: [13],
    7: [14, 15],
    8: [16, 17],
    9: [18],
    10: [19, 20],
    11: [21, 22],
    12: [23],
    13: [24, 25]
}

def utility(node_no):
    return node_vals[node_no][2]

pruned_nodes, val = alpha_beta_prune(tree, node_vals, 0)

print('Pruned Nodes')
for node in pruned_nodes:
    print(node_vals[node][0], node_type(node_vals[node][1]),
node_vals[node][2])

print('result value =', val)
```

**Output:**

```
Pruned Nodes
l4 leaf 12
K min None
l9 leaf 5
G max None
result value = 3
```