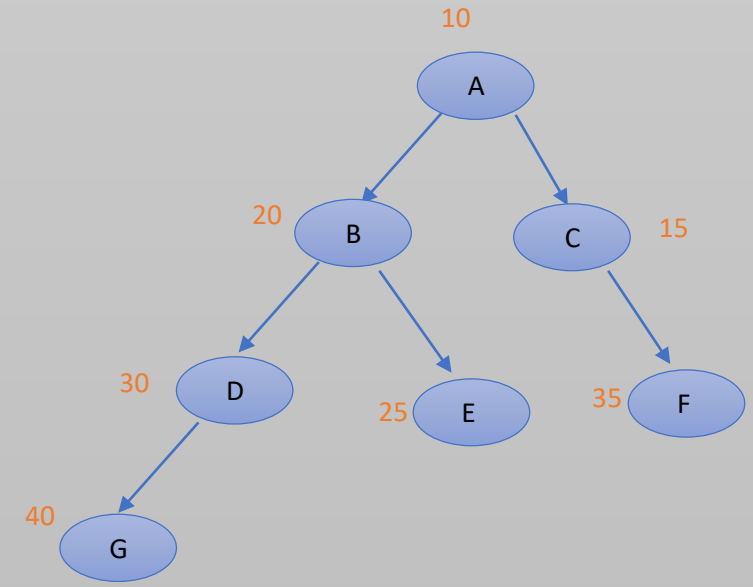
المحاضرة السادسة

Hill Climbing and Best First Search Algorithms

Hill Climbing Algorithm

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

    def add_child(self, child):
        self.children.append(child)

def hill_climbing_tree(root):
    current_node = root
    visited = set()
    while True:
        best_child = max(current_node.children,
                         key=lambda node: node.value,
                         default=None)
```

```
        if (best_child is None or best_child.value <=
current_node.value or   best_child in visited):
            return current_node.value

        visited.add(best_child)
        current_node = best_child
```

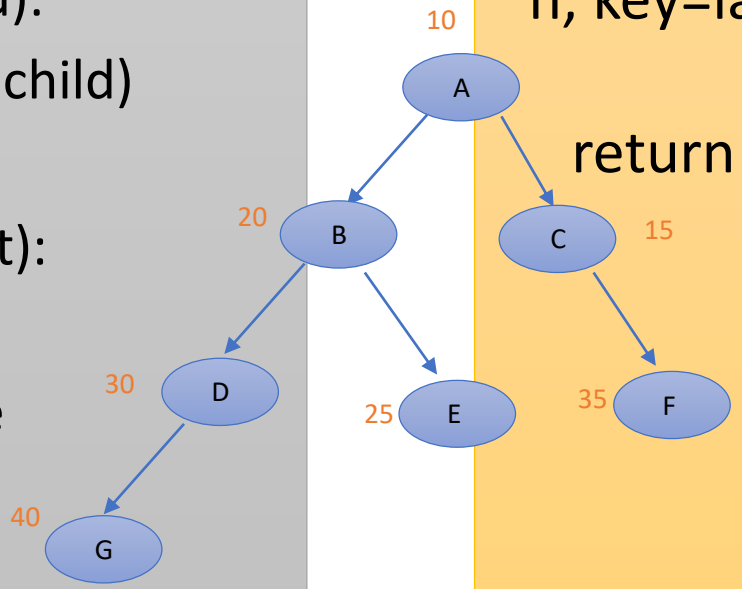


Hill Climbing Algorithm

```
root = Node(10)
child1 = Node(20)
child2 = Node(15)
child3 = Node(30)
child4 = Node(25)
child5 = Node(40)


root.add_child(child1)
root.add_child(child2)
child1.add_child(child3)
child1.add_child(child4)
child3.add_child(child5)
max_value = hill_climbing_tree(root)
print(f"أقصى قيمة وجدتها الخوارزمية: {max_value}")
```

Best First Search Algorithm

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

    def add_child(self, child):
        self.children.append(child)

def best_first_search(root):
    queue = [root]
    best_value = root.value
```




```
    while queue:
        current_node = queue.pop(0)
        best_value = max(best_value,
current_node.value)
        queue.extend(sorted(current_node.childre
n, key=lambda x: x.value, reverse=True))

    return best_value
```

```
root = Node(10)
child1 = Node(20)
child2 = Node(15)
child3 = Node(30)
child4 = Node(25)
child5 = Node(40)
child6 = Node(35)

root.add_child(child1)  # 10 → 20
root.add_child(child2)  # 10 → 15
child1.add_child(child3)  # 20 → 30
child1.add_child(child4)  # 20 → 25
child3.add_child(child5)  # 30 → 40
child2.add_child(child6)  # 15 → 35
max_value = best_first_search(root)
print(f"أقصى قيمة وجدتها الخوارزمية: {max_value}")
```