AO-star

October 24, 2024

[6]:

```python
class Node:
    def __init__(self, name, cost=0, is_and_node=False):
        self.name = name
        self.cost = cost
        self.is_and_node = is_and_node
        self.children = []

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

def ao_star_search(node):
    if not node.children:
        return node.cost

    total_cost = 0

    if node.is_and_node:

        for child in node.children:
            total_cost += ao_star_search(child)
        total_cost += node.cost
    else:

        min_cost = float('inf')
        for child in node.children:
            child_cost = ao_star_search(child)
            min_cost = min(min_cost, child_cost)
        total_cost = min_cost + node.cost  # Add the cost of the current node

    return total_cost

# And-Or graph construction
# A is an Or node, B and C are And nodes, D and E are leaf nodes
A = Node('A', 0, is_and_node=False)
B = Node('B', 2, is_and_node=True)
C = Node('C', 3, is_and_node=True)
D = Node('D', 1, is_and_node=False)
```

```python
E = Node('E', 4, is_and_node=False)

# graph
A.add_child(B)
A.add_child(C)
B.add_child(D)
B.add_child(E)


optimal_cost = ao_star_search(A)

print(f"The optimal cost to solve the And-Or graph is: {optimal_cost}")
```

```
The optimal cost to solve the And-Or graph is: 3
```

[ ]: