GREEDY BFS

In [18]:

```
def GREEDYBFS(graph, start, goal, heuristic):
    pq = [(heuristic[start], start, [start])]  # priority queue to store heuristic value, node, and path
    visited = set()  # create an empty set visited

    while pq:
        pq.sort(key=lambda x: x[0])  # sort list to get the smallest heuristic value
        heuristicvalue, currentnode, path = pq.pop(0)  # pop the first element

        if currentnode == goal:  # check if current node is equal to goal node
            return path  # return the path

        visited.add(currentnode)  # add current node to visited

        unvisitedneighbors = False

        for neighbor, _ in graph[currentnode]:  # check for all neighbors
            if neighbor not in visited:  # check if neighbor node is in visited
                pq.append((heuristic[neighbor], neighbor, path + [neighbor]))  # append the heuristic value of
                unvisitedneighbors = True

        if not unvisitedneighbors and not pq:  # If no unvisited neighbors and pq is empty
            print("Path does not exist")
            return None  # return None indicating no path

    print("Path does not exist")
    return None

# graph where a path exists from 'A' to 'F'
graph = {
    'A': [('B', 1), ('C', 1)],
    'B': [('A', 1), ('D', 1)],
    'C': [('A', 1), ('D', 1), ('E', 1)],
    'D': [('B', 1), ('C', 1)],
    'E': [('C', 1), ('F', 1)],
    'F': [('E', 1)]
}

heuristic = {
    'A': 6, 'B': 3, 'C': 4, 'D': 1, 'E': 2, 'F': 0
}

path = GREEDYBFS(graph, 'A', 'F', heuristic)
if path:
    print("Path found:", path)
```

```
Path found: ['A', 'C', 'E', 'F']
```