In [8]:
```python
#Using BFS
from queue import Queue

def bfs(graph, start, destination):
    visited = set()
    queue = Queue()
    queue.put((start, [start], 0))
    while not queue.empty():
        dequeue = queue.get()
        current, path, current_cost = dequeue
        if current == destination:
            return "Path Found " + str(path) + "\nTotal Cost: " + str(current_cost)
        if current not in visited:
            visited.add(current)
            for neighbor, t_cost in graph[current].items():
                if neighbor not in visited:
                    updated_cost = current_cost + t_cost
                    queue.put((neighbor, path + [neighbor], updated_cost))

    return "No Such Destination Exist"

graph = {
    'A': {'S': 140, 'T': 118, 'Z': 75},
    'S': {'A': 140, 'F': 99, 'O': 151, 'R': 80},
    'T': {'A': 118, 'L': 111},
    'Z': {'A': 75, 'O': 71},
    'F': {'S': 99, 'B': 211},
    'O': {'S': 151, 'Z': 71},
    'R': {'S': 80, 'P': 97, 'C': 146},
    'L': {'T': 111, 'M': 70},
    'B': {'F': 211, 'P': 101, 'G': 90, 'U': 85},
    'P': {'R': 97, 'B': 101, 'C': 138},
    'C': {'R': 146, 'P': 138, 'D': 120},
    'M': {'L': 70, 'D': 75},
    'G': {'B': 90},
    'U': {'B': 85, 'H': 98, 'V': 142},
    'D': {'M': 75, 'C': 120},
    'H': {'U': 98, 'E': 86},
    'E': {'H': 86},
    'V': {'U': 142, 'I': 92},
    'I': {'V': 92, 'N': 87},
    'N': {'I': 87}
}

start_city = input("Enter starting point ")
destination_city = input("Enter Destination")

result = bfs(graph, start_city, destination_city)

print(result)
```

```
Enter starting point A
Enter DestinationD
Path Found ['A', 'S', 'R', 'C', 'D']
Total Cost: 486
```

In [6]:
```python
#Using UCS
from queue import PriorityQueue

def ucs(graph, start, destination):
    priority_queue = PriorityQueue()
    priority_queue.put((0, start, []))
    visited = set()

    while not priority_queue.empty():
        current_cost, current, path = priority_queue.get()

        if current in visited:
            continue

        visited.add(current)

        if current == destination:
            return "To reach at "+ current+" this path would be followed \n"+ str(path) +"\nCost to this destin

        for neighbor, edge_cost in graph[current].items():
            if neighbor not in visited:
                priority_queue.put((current_cost + edge_cost, neighbor, path + [current]))

    return None


graph = {
    'A': {'S': 140, 'T': 118, 'Z': 75},
    'S': {'A': 140, 'F': 99, 'O': 151, 'R': 80},
    'T': {'A': 118, 'L': 111},
    'Z': {'A': 75, 'O': 71},
    'F': {'S': 99, 'B': 211},
    'O': {'S': 151, 'Z': 71},
    'R': {'S': 80, 'P': 97, 'C': 146},
    'L': {'T': 111, 'M': 70},
    'B': {'F': 211, 'P': 101, 'G': 90, 'U': 85},
    'P': {'R': 97, 'B': 101, 'C': 138},
    'C': {'R': 146, 'P': 138, 'D': 120},
    'M': {'L': 70, 'D': 75},
    'G': {'B': 90},
    'U': {'B': 85, 'H': 98, 'V': 142},
    'D': {'M': 75, 'C': 120},
    'H': {'U': 98, 'E': 86},
    'E': {'H': 86},
    'V': {'U': 142, 'I': 92},
    'I': {'V': 92, 'N': 87},
    'N': {'I': 87}
}

start_city = input("Enter starting point ")
destination_city = input("Enter Destination")

result = ucs(graph, start_city, destination_city)

print(result)
```

```
Enter starting point A
Enter DestinationI
To reach at I this path would be followed
['A', 'S', 'R', 'P', 'B', 'U', 'V']
Cost to this destination is 737
```

In [ ]: