In [ ]:
```
# Roll No : 3310
# Name : Srushti Prakash Bhoite
# Batch A
# Problem Statement : Identify and Implement heuristic and search strategy for
                      #Travelling Salesperson Problem
```

In [6]:
```
import sys
```

In [7]:
```
r=int(input("Enter the number of rows: "))
```

```
Enter the number of rows: 4
```

In [8]:
```
dist_matrix=[]
print("Enter the entries rowwise: ")
for i in range(r):
    r=list(map(int,input().split()))
    dist_matrix.append(r)
print("\nThe distance matrix is:\n")
for i in dist_matrix:
    print(i)
```

```
Enter the entries rowwise:
0 5 15 4
5 0 35 25
15 35 0 30
4 25 30 0

The distance matrix is:

[0, 5, 15, 4]
[5, 0, 35, 25]
[15, 35, 0, 30]
[4, 25, 30, 0]
```

In [9]:
```
def nearest_neighbor(curr,unvisited,dist_matrix): #Returns the nearest neighbor to the
    nearest=sys.maxsize
    neighbor=None
    for city in unvisited:
        if dist_matrix[curr][city]<nearest:
            nearest=dist_matrix[curr][city]
            neighbor=city
    return neighbor,nearest

def tsp_nn(dist_matrix): #Solves the traveling salesman problem using the nearest neig
    n=len(dist_matrix)
    Tourlist=[]
    Costlist=[]
    for j in range(0,n):
        tour=[j]*(n+1) #Initialize the tour
        unvisited=set(range(0,n)) #set of unvisited cities
        unvisited.remove(j)
        curr_city=j; #Starting city

        for i in range(1,n):
            next_city,dist=nearest_neighbor(curr_city,unvisited,dist_matrix)
            tour[i]=next_city
            curr_city=next_city
            unvisited.remove(next_city)
        #return to the starting city
        tour[0]=j
        #Calculate the total cost of the tour
        cost=sum(dist_matrix[tour[i]][tour[i+1]] for i in range(n-1))
        cost+=dist_matrix[tour[n-1]][tour[0]]
        print("For travelling starting from city ",j,":")
        print("Tour: ",tour)
        print("Total Cost: ",cost)
        Tourlist.append(tour)
        Costlist.append(cost)
    print("\nThe minimum cost is:",min(Costlist))
    index=Costlist.index(min(Costlist))
    print("The optimised path is:",Tourlist[index])

tsp_nn(dist_matrix)
```

```
For travelling starting from city  0 :
Tour:  [0, 3, 1, 2, 0]
Total Cost:  79
For travelling starting from city  1 :
Tour:  [1, 0, 3, 2, 1]
Total Cost:  74
For travelling starting from city  2 :
Tour:  [2, 0, 3, 1, 2]
Total Cost:  79
For travelling starting from city  3 :
Tour:  [3, 0, 1, 2, 3]
Total Cost:  74

The minimum cost is: 74
The optimised path is: [1, 0, 3, 2, 1]
```