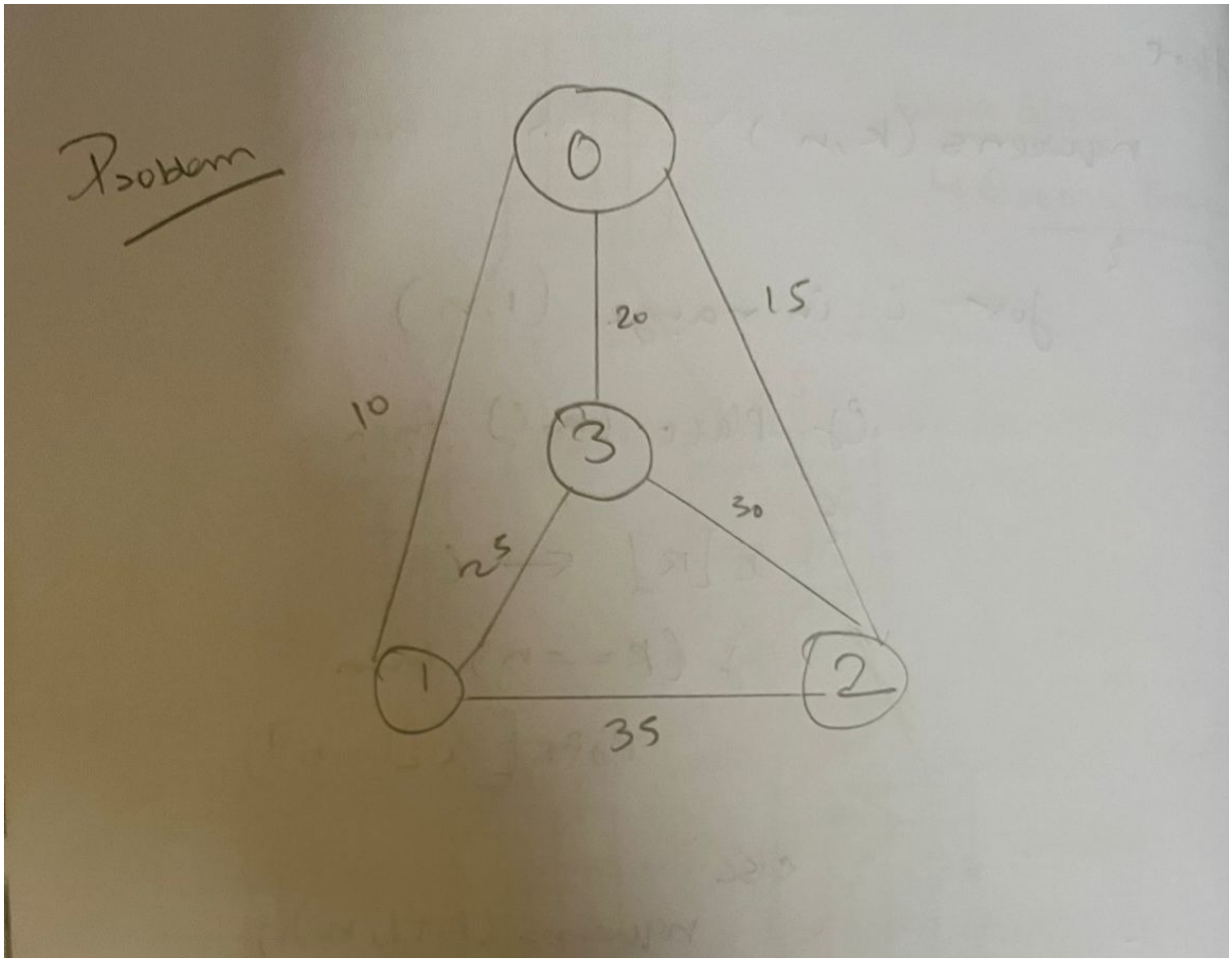
EX2
Developing Agent Programs for Real-World Problems



Experiment 2

Traveling Salesman Problem

AIM ~ To find the shortest possible route for a traveling salesman who have to visit multiple cities given the condition he can only visit them once and return back to the starting city.

ALGORITHM

→ Consider city 0 as the starting and ending point. Since the ~~routing~~ route is cyclic, we can consider any point as a ~~starting~~ point.

→ Generate all $(n-1)!$ permutations of cities

→ Calculate the cost of every permutation and keep track of minimum cost permutation

→ Return the permutation with minimum cost.

Possible Solution

Minimum weight of the hamilton cycle

Dist(0-1) + Dist(2-3) + Dist(3-2) + Dist(2-0)

= 10 + 25 + 30 + 15 = 80

```python
from sys import maxsize
from itertools import permutations
v=4

def tsm(graph,s):

    vertex=[]

    for i in range(v):
        if i!=s:
            vertex.append(i)

    pvertex=[0,vertex]
    print("vertex",pvertex)
    min_path=maxsize
    next_p=permutations(vertex)

    for i in next_p:
        current_pw=0

        k=s
        print("Path",i,"-",end=" ")
        for j in i:

            print(graph[k][j],end=" ")
            current_pw+=graph[k][j]


            k=j

        print(graph[k][s],end=" ")
        current_pw+= graph[k][s]
        print("- Weight:",current_pw)


        min_path = min(min_path, current_pw)
```

```
    return min_path

if __name__ == "__main__":

      graph = [[0, 10, 15, 20], [10, 0, 35, 25],
            [15, 35, 0, 30], [20, 25, 30, 0]]

      s = 0
      print("\nMinimum weight of Hamiltonian Cycle :",tsm(graph, s))
```