Για την υλοποίηση του δικτύου διανομής πληροφοριών αρχικά δημιουργήσαμε ενα αρχείο τύπου txt το data.txt στο οποίο αποθηκεύσαμε της πληροφορίες που μας δόθηκαν απο την άσκηση. Στην συνέχεια χρησιμοποιώντας τις γνώσεις για τον αλγόριθμο του Traveling Sales Person και αναδρομής υλοποιήσαμε ενα πρόγραμμα με σκοπό να βρουμε την βέλτιστη διαδρομή που θα κάνει ο αρχηγός της ομάδας με το μικρότερο ρίσκο σύλληψης. Παράτηρουμε μετα την υλοποιήση των παραπάνω αλγορίθμων οτι πρόκειται για ενα κυκλο Hamilton ο οποίος έχει την ιδιότητα του μικρότερου κόστους μονοπατιού.

Αρχικοποίηση της κλάσης και του γραφήματος

```python
class travelingSalesperson:
    def __init__(self,data):
        self.graph = {}
        with open(data, 'r') as file:
            # Skip the header
            next(file)
            for line in file:
                sender, receiver, risk = line.strip().split(',')
                risk = int(risk)
                if sender not in self.graph:
                    self.graph[sender] = {}
                if receiver not in self.graph:
                    self.graph[receiver] = {}
                self.graph[sender][receiver] = risk
```

Αλγόριθμος tsp

```python
def tsp(self):
    agents = list(self.graph.keys())
    for start_agent in agents:
        shortest_path, shortest_distance = self.tsp_backtracking(start_agent, agents)
        print(f"Shortest Path from {start_agent}: {shortest_path}")
        print(f"Shortest Distance Cost: {shortest_distance}\n")
```

```python
def tsp_backtracking(self,start_agent, agents):
    shortest_path = [None]
    shortest_distance = [float('inf')]
    self.backtrack([start_agent], [agent for agent in agents if agent != start_agent], shortest_path, shortest_distance,start_agent,"")

    return shortest_path, shortest_distance[0]
```

```
Sender,Receiver,Risk
ALPHA,BRAVO,13
ALPHA,CHARLIE,51
ALPHA,FOXTROT,70
ALPHA,DELTA,68
ALPHA,ECHO,51
BRAVO,ALPHA,13
BRAVO,CHARLIE,60
BRAVO,FOXTROT,70
BRAVO,DELTA,68
BRAVO,ECHO,61
CHARLIE,ALPHA,51
CHARLIE,BRAVO,60
CHARLIE,FOXTROT,56
CHARLIE,DELTA,35
CHARLIE,ECHO,2
FOXTROT,ALPHA,70
FOXTROT,BRAVO,70
FOXTROT,CHARLIE,56
FOXTROT,DELTA,21
FOXTROT,ECHO,57
DELTA,ALPHA,68
DELTA,BRAVO,68
DELTA,CHARLIE,35
DELTA,FOXTROT,21
DELTA,ECHO,36
ECHO,ALPHA,51
ECHO,BRAVO,61
ECHO,CHARLIE,2
ECHO,FOXTROT,57
ECHO,DELTA,36
```

Αναδρομικός αλγόριθμος

```python
def backtrack(self,curr_path, remaining_agents, shortest_path, shortest_distance,start_agent,current_agent):
    if not remaining_agents:
        distance = self.calculate_distance(curr_path,start_agent,current_agent)
        if distance < shortest_distance[0]:
            shortest_distance[0] = distance
            shortest_path[0] = curr_path.copy()
        return
    for agent in remaining_agents:
        new_path = curr_path + [agent]
        new_remaining = [n for n in remaining_agents if n != agent]
        self.backtrack(new_path, new_remaining, shortest_path, shortest_distance,start_agent,agent)
```

Υπολογισμός απόστασης

```python
def calculate_distance(self,path,start_agent,last_agent):
    total_distance = 0
    for i in range(len(path) - 1):
        total_distance += self.graph[path[i]][path[i + 1]]
    total_distance +=self.graph.get(start_agent).get(last_agent)
    return total_distance
```

Αποτελέσματα

```
[Running] python -u "f:\UOI\TheoriaGraf\erg4\erg4.4.py"
Shortest Path from ALPHA: [['ALPHA', 'BRAVO', 'FOXTROT', 'DELTA', 'CHARLIE', 'ECHO']]
Shortest Distance Cost: 192

Shortest Path from BRAVO: [['BRAVO', 'ALPHA', 'ECHO', 'CHARLIE', 'DELTA', 'FOXTROT']]
Shortest Distance Cost: 192

Shortest Path from CHARLIE: [['CHARLIE', 'DELTA', 'FOXTROT', 'BRAVO', 'ALPHA', 'ECHO']]
Shortest Distance Cost: 192

Shortest Path from FOXTROT: [['FOXTROT', 'BRAVO', 'ALPHA', 'ECHO', 'CHARLIE', 'DELTA']]
Shortest Distance Cost: 192

Shortest Path from DELTA: [['DELTA', 'CHARLIE', 'ECHO', 'ALPHA', 'BRAVO', 'FOXTROT']]
Shortest Distance Cost: 192

Shortest Path from ECHO: [['ECHO', 'ALPHA', 'BRAVO', 'FOXTROT', 'DELTA', 'CHARLIE']]
Shortest Distance Cost: 192


[Done] exited with code=0 in 0.061 seconds
```