HW4 - The Shadow Network

Background

In the amazing and advanced country of Wakunda, a secret spy had a very important job: to send top-secret information back to their headquarters. But this was tricky because Wakunda had a really good system to catch the top-secret information when spies contacted each other. To get around this, the spy came up with a clever plan: they would use regular people from Wakunda, who didn't know they were helping, to pass/ become the intermediary messages without getting caught.

The important process of message delivery can be highlighted as follow:

- To send the secret message to headquarters, the spies use text messages and pass them along through a network of people, including regular folks and other spies.
- These spies always use the shortest path to the headquarter for message delivery and avoid direct contact with other spies (excluding the headquarter).
- Since we are using text messages, **the distance** calculation between contacts will be **based on how many messages were transferred between each contact:**
  - The reasoning is: it will increase the chance of message delivery (chain of contacts with close relationships). Additionally, by smuggling the message through numerous text messages between contacts makes the message delivery less obvious and requires Wakunda's system to scan through more messages.
  - The message can be transmitted directly from the source to the headquarter if there is a high volume of text communication between them and the distance is shorter than alternative routes.

Wakunda became aware of this issue and decided to take action. They hired you to join their intelligence team and help them **create a network simulation process**. By creating this simulation, you will help the Wakunda intelligence team to analyze and find any weak points in their communication network. This will help the future of Wakunda to be more secure and better prepared to protect their intelligence from enemy spies.

Network Simulation in Graph

There are two important parts of the network simulation:

1) Network Structure Creation

- **Node:** Is the actor involved in the network/communication, each of the node has a name consisting of 1 word and has a unique ID number assigned based on the node/actor insertion order
  - There are four types of nodes in the network:
    - SOURCE: The **spy** who **initiates the message**.
    - SPY: Other spies in the network.
    - CIV: Civilians unwittingly involved.
    - HQ: Headquarter, **the message's destination**.
- **Edge:** Connections between actors in the form of "distance", distance calculation based on the number of messages exchanged
  - The formula of the distance between two nodes: **1000 divided by the total message between 2 nodes**, round to the **nearest 2 decimals**
    - A and B trade 3 messages, the distance will be 1000/3 = 333.33 (round down)
    - C and D trade 70 messages, the distance will be 1000/70 = 14.29 (round up)
    - If there's no message between actors, you can assume their distance is infinite:
      E and F don't don't have a shared edge; the distance will be infinite
  - Allowed connections:
    - SOURCE ----- CIV : Less-suspicious
    - SPY ----- CIV : Less-suspicious
    - CIV ----- CIV : Less-suspicious
    - SOURCE ----- HQ : Source (spy) have access to HQ
    - SPY ----- HQ : Spy have access to HQ
  - Unallowed connections:
    - SOURCE --X-- SPY : Both spy easily detected by Wakunda's system
    - SPY --X-- SPY : Both spy easily detected by Wakunda's system
    - CIV --X-- HQ: The civilian doesn't have access to the spy's headquarters
  - We use the undirected graph, so each node will share the same distance regardless of the direction
- We provide an example of allowed & unallowed sequence of contacts:
  - **Allowed:** SOURCE ----- CIV ----- CIV ----- SPY ----- HQ : all connections is allowed
  - **Unallowed:** SOURCE --X-- SPY --X-- SPY -----CIV ----- CIV --X-- HQ : 3 unallowed connections

2) Network Analysis

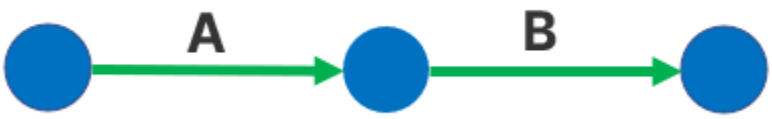
There are two main processes in the network analysis:

1. Determine the shortest path using the **Dijkstra algorithm**
2. Print three detailed information on the shortest-path:
    a. **The sequence of contacts that yields the shortest path** among all possible combinations from source to destination
    b. The total number of messages in the traversal of the shortest path
    c. The total distance in the traversal of the shortest path

After the analysis, we can print our output and end/terminate our application

Note 1: Calculating the total distance in the network: you need to calculate & round the distance between two nodes first before summing multiple distances to get the total distance.

For example, given shortest path consists of two edges with distances A and BL



- Do: round the distance of edges A & B first, then sum them as total = A + B
- Don't: sum the distance A & B first, then round them

Note 2: If two paths have the same length of the shortest path but a different sequence of contacts:

- We don't need to consider the number of total messages and number of contacts in the sequence (this is to simplify the edge cases in the problem)
- Instead, focus on the sequence of the contacts:

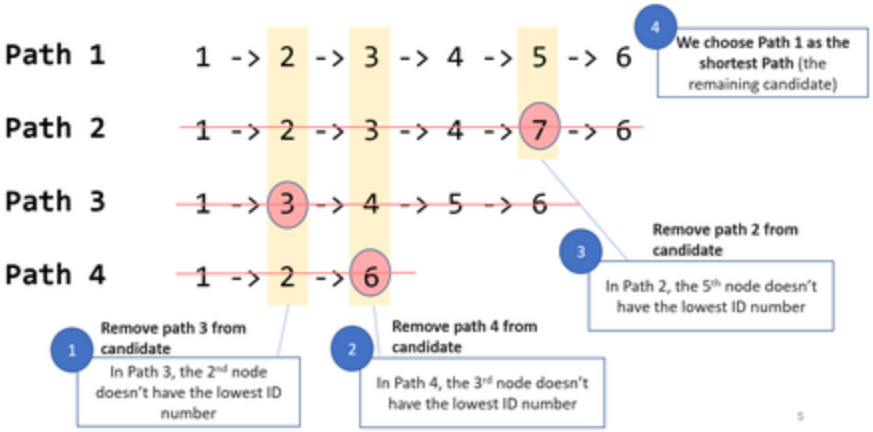
For a set of all shortest routes $a_z : a_{z,0} \rightarrow a_{z,1} \ldots \rightarrow a_{z,n} \rightarrow D$ where the distances of all $a_{z,0} \rightarrow T$ are equal, select the route a_x such that when the first node where $\text{index}(a_{x,j}) \neq \text{index}(a_{y,j})$, $\text{index}(a_{x,j}) < \text{index}(a_{y,j})$.

For instance, if there's a 4 different sequence of nodes (path) that yields the shortest path (same distance):

```
Path 1   1 -> 2 -> 3 -> 4 -> 5 -> 6

Path 2   1 -> 2 -> 3 -> 4 -> 7 -> 6

Path 3   1 -> 3 -> 4 -> 5 -> 6

Path 4   1 -> 2 -> 6
```

We will iterate from the beginning of the node sequences:

- First node: all paths share the same ID number = 1
- Second node: the lowest ID number is 2:
    - We keep the remaining path which has ID number = 2 in the second node (path 1, 2 & 4)
    - We eliminate path which has a higher ID number (path 3)
- Third node: the lowest ID number is 3:
    - We keep the remaining path which has ID number = 3 in the third node (path 1 & 2)
    - We eliminate path which has a higher ID number (path 4)
- Fourth node: all remaining paths share the same ID number = 4
- Fifth node: the lowest node is 5
    - We keep the remaining path which has ID number = 5 in the fifth node (path 1)
    - We eliminate path which has a higher ID number (path 2)
- Since we only have **one remaining candidate**, we **select Path 1 to represent the shortest path**



Functions in Network Simulation

1) Node Insertion

```
INSERT <nodetype> <name>
```

- **<nodetype>** values: SOURCE, SPY, CIV, HQ
- **<name>** : name of the actor always consist of 1 word
- Assign each a unique ID number based on insertion order, irrespective of node type (SOURCE, SPY, CIV, or HQ), starting from 1.

For example, if we insert the node based on the following sequence:

```
INSERT SOURCE Anya  // Anya ID number is 1
INSERT SPY Ben      // Ben ID number is 2
INSERT HQ Charlie   // Charlie ID number is 3
INSERT CIV Dave     // Dave ID number is 4
```

It's guaranteed that:

- The number of **INSERT SOURCE <name>** and **INSERT HQ <name>** happened once in each test case
- The number of **INSERT CIV <name>** happened at least once in each test case
- The **<name>** only contains one word

2) Edge Insertion

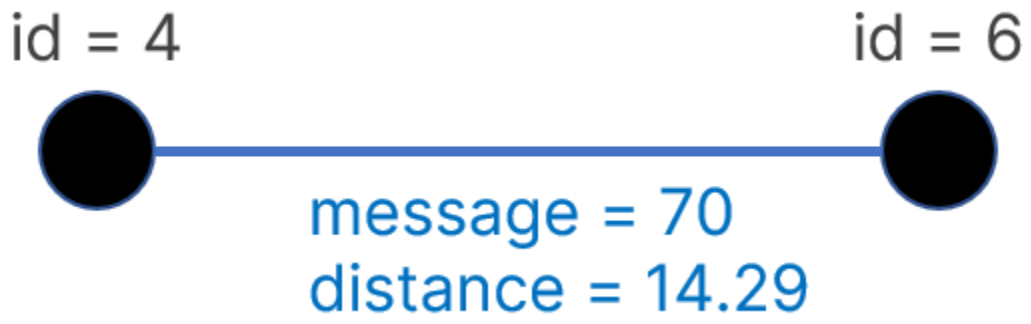
```
INSERT_EDGE <id_number1> <id_number2> <total_message>
```

For example, we want to connect the nodes with ID number 6 and 3, which has 70 messages:

```
INSERT_EDGE 6 3 70     // Equivalent to INSERT_EDGE 3 6 70 (distance =
1000/70 = 14.29)
```

It's guaranteed that:

- Edges consisting of the same pair will not be inserted more than once
- 1 <= id_number1 <= total number of nodes
- 1 <= id_number2 <= total number of nodes
- id_number1 != id_number 2
- <total_message> <= 10^5



3) Network Analysis

```
ANALYSIS
```

The function will generate the shortest path and print 3 lines of its properties with a newline before terminate the program:

```
<order of contacts in shortest path> // without newspace in the end of
the line
<total message in shortest path>     // without newspace in the end of
the line
<total distance in shortest path>    // without newspace in the end of
the line
<new line>
```

For example, if we have the following shortest path properties:

- Order of contacts: Anya -> Ben -> Dave -> Franky -> Eric -> Charlie
- Total message: 1070
- Total distance: 36.29

The output will be:

```
Anya -> Ben -> Dave -> Franky -> Eric -> Charlie
1070
36.29
```

To avoid presentation error

- No whitespace in each line
- A newline need to be added after the total distance output
- There is a space before and after the "->" symbol

Assumptions and Suggestions

Assumption

- The shadow network is an undirected graph
- The source is always a spy (they can't communicate to spy, but can communicate to civilian & headquarter)
- The destination is always a headquarter
- There is only one source and one headquarters in each testcases
- At least one spy (including source), one civilian and one headquarter in each testcases
- The insertion of the edge always happened once for each unique pair of nodes
- If there is no message between two nodes, you can assume the distance is infinite
- It's possible that multiple sequences of nodes (path) will result in the shortest path, read the "Network Analysis" part to find the instructions for selecting the path.
- The shortest path between the source and headquarter in every test case never consists of an infinite distance.

Restriction

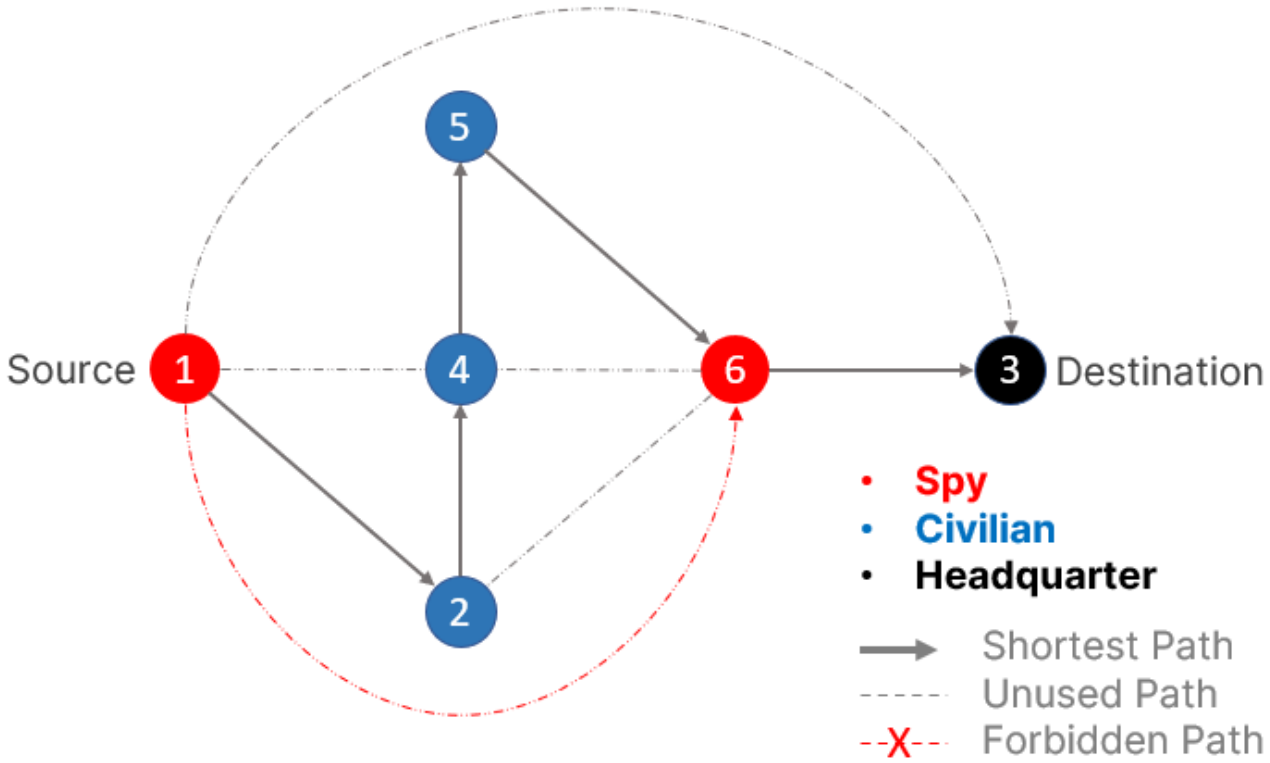
- Any form of plagiarism
- The code should be written in C/C++
- The code should only contain the following library & namespace, you might receive a score deduction if you're using the other libraries:

```
#include<iostream>
#include<vector>
#include<stack>
#include<queue>
#include<list>
#include<string.h>
#include<cmath>
using namespace std;
```

Tips

- Use classes to represent the objects
- Use **Dijkstra Algorithm** to find the shortest path
- Break down the problem, especially in the Dijkstra Algorithm (divide & conquer)
- If there's no message between actors, you can assume their distance is infinite
- Testcase 1 is the sample Input & Output
- Please contact the TA Edwin (any channel and use Teams chat for fast response / no reply from other channel) if you have any queries

Graph Illustration



Example

Input

```
INSERT SOURCE Anya
INSERT CIV Ben
INSERT HQ Charlie
INSERT CIV Dave
INSERT CIV Eric
INSERT SPY Franky
INSERT_EDGE 1 2 200
INSERT_EDGE 1 3 4
INSERT_EDGE 6 1 10000
INSERT_EDGE 2 4 100
INSERT_EDGE 2 6 30
INSERT_EDGE 4 5 200
INSERT_EDGE 6 4 80
INSERT_EDGE 5 6 500
INSERT_EDGE 6 3 70
ANALYZE
```

Output

```
Anya -> Ben -> Dave -> Franky -> Eric -> Charlie
1070
36.29
```