A. Second best friends

You are given a network of persons which can have direct connections with persons in the same network. Given a specific name of a person you should be able to identify the persons with which it has a direct connection with inside a network. Based on this given context your task is to solve the following problem: given a name of a person, your program should return a **sorted** list of person names which don't have a direct connection with the input person, but a direct connection with the direct connections of the given person.

- Read the input from a file (`input.txt`)
- Write the output to a file (`output.txt`)
- Break down your implementation into multiple functions
- Find & use data structures that seem suitable from your point of view

Example `input.txt` contents:

```
8 A // Number of persons in network followed by starting person name
A B C E G
B A C H
C A B D H
D C
E A F // Person with name "E" has a direct connection with "A" and "F"
F E
G A
H B C
```

Example `output.txt` contents:

```
D F H // Second best friends of "A"
```

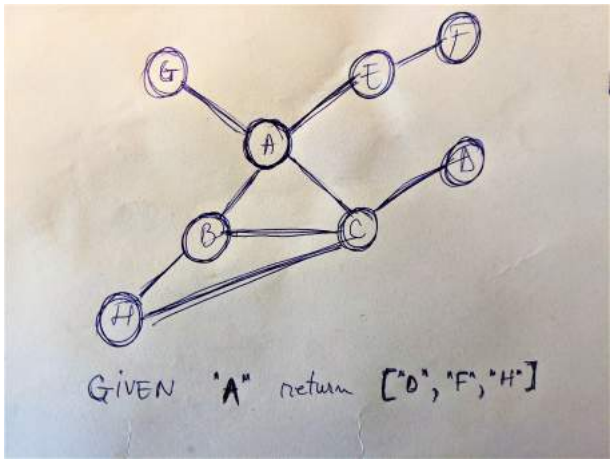

Figure 1: Example network & result

Requirements

1. Read the input data and build the data structure.
2. Implement a function to search a node by name given a reference into the network. Print a message `"node does not exist or not reachable"` in case you cannot find the node or if it's not reachable.
3. Properly handle already 'visited' nodes so that your program doesn't go intro an infinite loop
4. Build and print the final list to the output file. If the given person has no second best friends simply output the `"no second best friends"` message.
5. Make sure that the output list is sorted in an increasing order by the person names.