Report ADSA

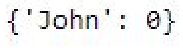
LIM Kévin LOUCHART Boris

Step 1: To organize the tournament

1. **Propose a data structure to represent a Player and its Score**

As our tournament is a ranking system, we should choose a Data Structure which is able to be read, modified, and evaluated.

We need to use a structure to store the name of a player and its score. To do so we will use a dictionary entry.
For the representation of one player we will use one dictionary entry, the key will be the name of the player and the value will be its score

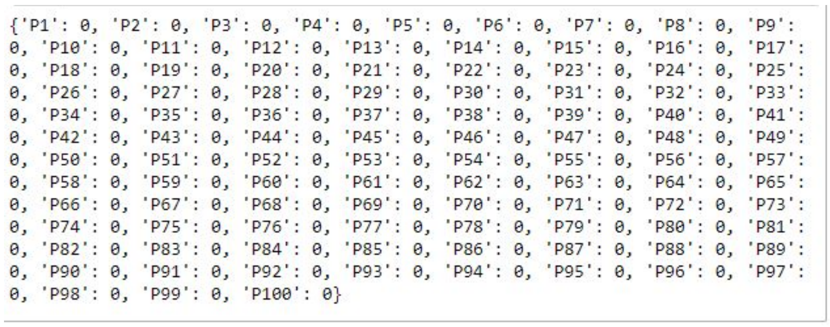


```
{'John': 0}
```

2. **Propose a most optimized data structures for the tournament (called database in the following questions)**

We chose to use a dictionary for the tournament. Each entry of the dictionary corresponds to one player. The key is the name of the player and the value will be its score.

As the tournament goes, we will order the entries by its value in a descending order to obtain a ranking of the player who has the most points.



```
{'P1': 0, 'P2': 0, 'P3': 0, 'P4': 0, 'P5': 0, 'P6': 0, 'P7': 0, 'P8': 0, 'P9':
0, 'P10': 0, 'P11': 0, 'P12': 0, 'P13': 0, 'P14': 0, 'P15': 0, 'P16': 0, 'P17':
0, 'P18': 0, 'P19': 0, 'P20': 0, 'P21': 0, 'P22': 0, 'P23': 0, 'P24': 0, 'P25':
0, 'P26': 0, 'P27': 0, 'P28': 0, 'P29': 0, 'P30': 0, 'P31': 0, 'P32': 0, 'P33':
0, 'P34': 0, 'P35': 0, 'P36': 0, 'P37': 0, 'P38': 0, 'P39': 0, 'P40': 0, 'P41':
0, 'P42': 0, 'P43': 0, 'P44': 0, 'P45': 0, 'P46': 0, 'P47': 0, 'P48': 0, 'P49':
0, 'P50': 0, 'P51': 0, 'P52': 0, 'P53': 0, 'P54': 0, 'P55': 0, 'P56': 0, 'P57':
0, 'P58': 0, 'P59': 0, 'P60': 0, 'P61': 0, 'P62': 0, 'P63': 0, 'P64': 0, 'P65':
0, 'P66': 0, 'P67': 0, 'P68': 0, 'P69': 0, 'P70': 0, 'P71': 0, 'P72': 0, 'P73':
0, 'P74': 0, 'P75': 0, 'P76': 0, 'P77': 0, 'P78': 0, 'P79': 0, 'P80': 0, 'P81':
0, 'P82': 0, 'P83': 0, 'P84': 0, 'P85': 0, 'P86': 0, 'P87': 0, 'P88': 0, 'P89':
0, 'P90': 0, 'P91': 0, 'P92': 0, 'P93': 0, 'P94': 0, 'P95': 0, 'P96': 0, 'P97':
0, 'P98': 0, 'P99': 0, 'P100': 0}
```

3. **Present and argue about a method that randomize player score at each game (between 0 point to 12 points)**

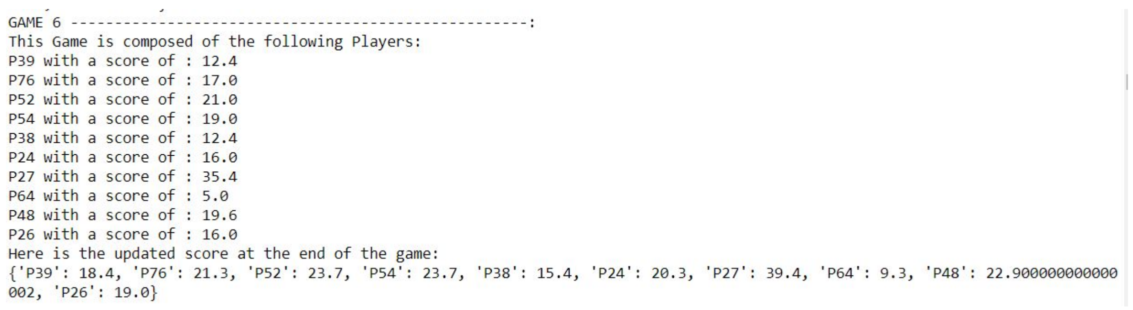
The methods that randomize the score of a player is the following. We generate and sum a number between 0 and 12 the number of games we play. For example, 3 in the first phase of the tournament.

4. **Present and argue about a method to update Players score and the database**

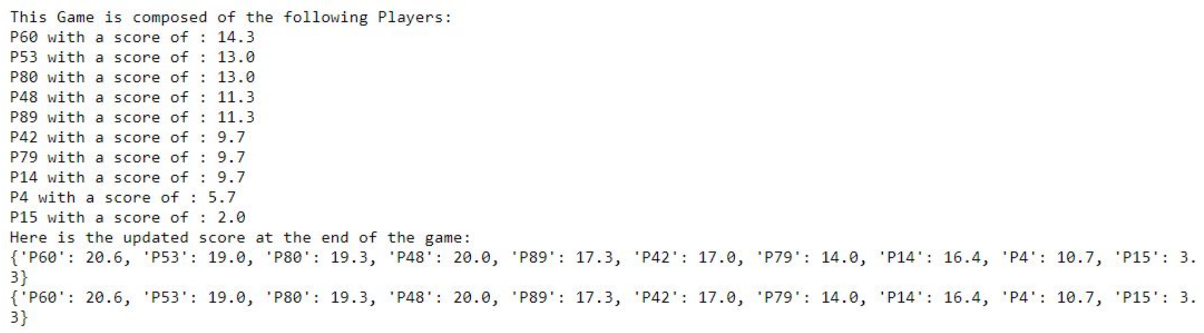
For each players of a game, we will obtain its current score then add the score he obtains after the game. Once the score updated, we put the player back into the dictionary.

5. **Present and argue about a method to create random games based on the database**

In order to create random games based on the database we will use a variable that will take a number between 0 and the length of the dictionary minus one. That will randomly select a player from the tournament.



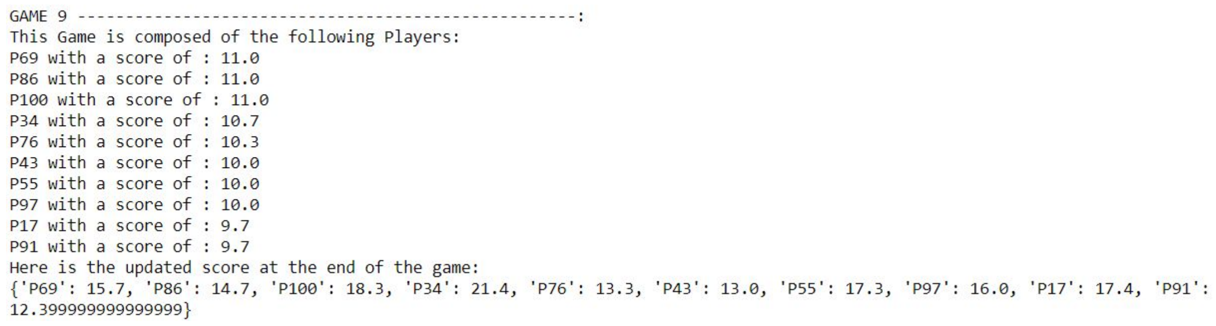
```
GAME 6 ---------------------------------------------------:
This Game is composed of the following Players:
P39 with a score of : 12.4
P76 with a score of : 17.0
P52 with a score of : 21.0
P54 with a score of : 19.0
P38 with a score of : 12.4
P24 with a score of : 16.0
P27 with a score of : 35.4
P64 with a score of : 5.0
P48 with a score of : 19.6
P26 with a score of : 16.0
Here is the updated score at the end of the game:
{'P39': 18.4, 'P76': 21.3, 'P52': 23.7, 'P54': 23.7, 'P38': 15.4, 'P24': 20.3, 'P27': 39.4, 'P64': 9.3, 'P48': 22.900000000000
002, 'P26': 19.0}
```



```
This Game is composed of the following Players:
P60 with a score of : 14.3
P53 with a score of : 13.0
P80 with a score of : 13.0
P48 with a score of : 11.3
P89 with a score of : 11.3
P42 with a score of : 9.7
P79 with a score of : 9.7
P14 with a score of : 9.7
P4 with a score of : 5.7
P15 with a score of : 2.0
Here is the updated score at the end of the game:
{'P60': 20.6, 'P53': 19.0, 'P80': 19.3, 'P48': 20.0, 'P89': 17.3, 'P42': 17.0, 'P79': 14.0, 'P14': 16.4, 'P4': 10.7, 'P15': 3.
3}
{'P60': 20.6, 'P53': 19.0, 'P80': 19.3, 'P48': 20.0, 'P89': 17.3, 'P42': 17.0, 'P79': 14.0, 'P14': 16.4, 'P4': 10.7, 'P15': 3.
3}
```

6. **Present and argue about a method to create games based on ranking**

To create a game based on ranking, we need to take the first 10 players from the ranking to create the game, and we do that until all players has played.



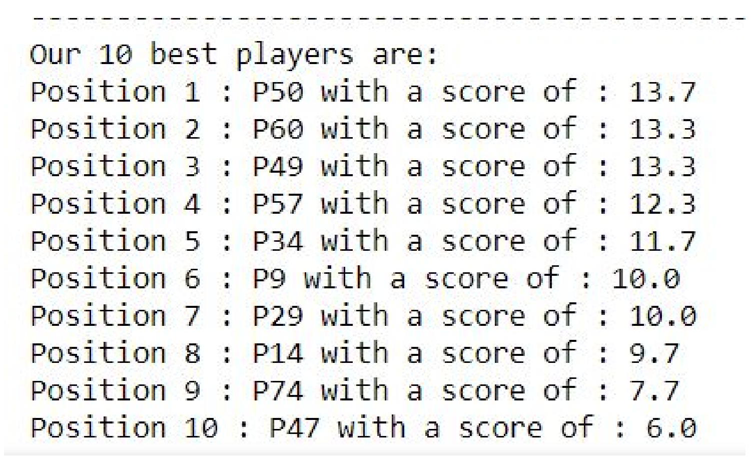
```
GAME 9 ---------------------------------------------------:
This Game is composed of the following Players:
P69 with a score of : 11.0
P86 with a score of : 11.0
P100 with a score of : 11.0
P34 with a score of : 10.7
P76 with a score of : 10.3
P43 with a score of : 10.0
P55 with a score of : 10.0
P97 with a score of : 10.0
P17 with a score of : 9.7
P91 with a score of : 9.7
Here is the updated score at the end of the game:
{'P69': 15.7, 'P86': 14.7, 'P100': 18.3, 'P34': 21.4, 'P76': 13.3, 'P43': 13.0, 'P55': 17.3, 'P97': 16.0, 'P17': 17.4, 'P91':
12.399999999999999}
```

7. **Present and argue about a method to drop the players and to play game until the last 10 players**

To drop players which have the worst score, we will take the ranking and drop the last 10 players after each round of the tournament.

8. **Present and argue about a method which display the TOP10 players and the podium after the final game.**

Before doing the last game, we need to reset the score of each player. After the score is reset, we will run the 5-last game with the 10 remaining players. And then order the rank by the score.
We will then have the top 10 players and the podium.



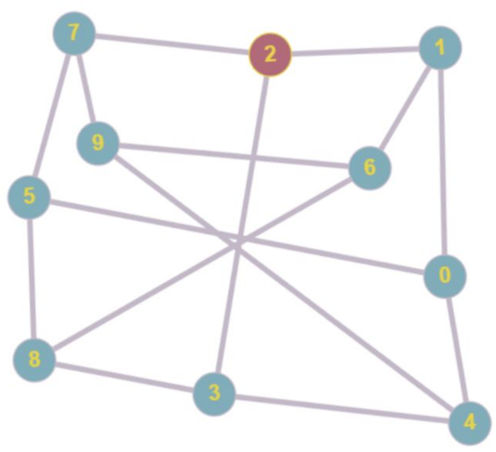
```
------------------------------------------
Our 10 best players are:
Position 1 : P50 with a score of : 13.7
Position 2 : P60 with a score of : 13.3
Position 3 : P49 with a score of : 13.3
Position 4 : P57 with a score of : 12.3
Position 5 : P34 with a score of : 11.7
Position 6 : P9 with a score of : 10.0
Position 7 : P29 with a score of : 10.0
Position 8 : P14 with a score of : 9.7
Position 9 : P74 with a score of : 7.7
Position 10 : P47 with a score of : 6.0
```

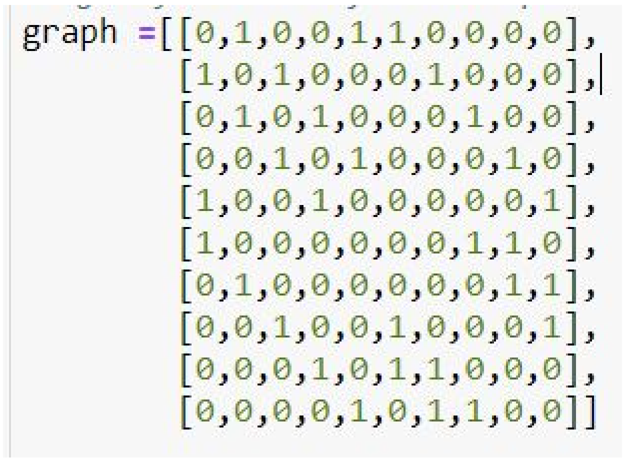
Step 2: Professor Layton < Guybrush Threepwood < You

1. **Represent the relation (have seen) between players as a graph, argue about your model.**

To represent the “have seen” relation between each player we can use an undirected graph.

We will represent this graph using the adjacency matrix on our program.



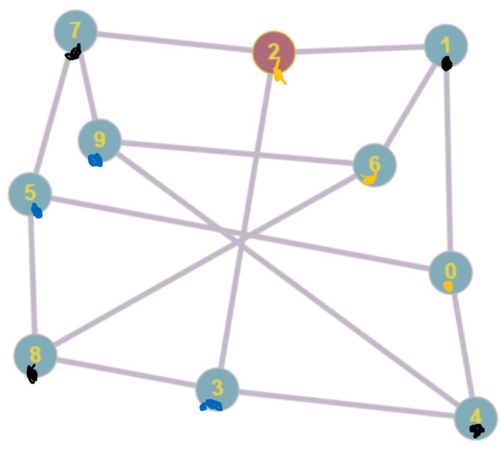


```
graph =[[0,1,0,0,1,1,0,0,0,0],
        [1,0,1,0,0,0,1,0,0,0],
        [0,1,0,1,0,0,0,1,0,0],
        [0,0,1,0,1,0,0,0,1,0],
        [1,0,0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,1,1,0],
        [0,1,0,0,0,0,0,0,1,1],
        [0,0,1,0,0,1,0,0,0,1],
        [0,0,0,1,0,1,1,0,0,0],
        [0,0,0,0,1,0,1,1,0,0]]
```

2. **Thanks to a graph theory problem, present how to find a set of probable impostors.**

The graph problem can be solved using graph coloring. As player 0 has been reported dead, we will consider player 1,4,5 as the first imposter at the time, and then see a set of probable second imposter.

The example below shows the case where 1 is the imposter. We can then color 2,6 and 0 as crew and see the remaining players that may be imposters.



3. **Argue about an algorithm solving your problem.**

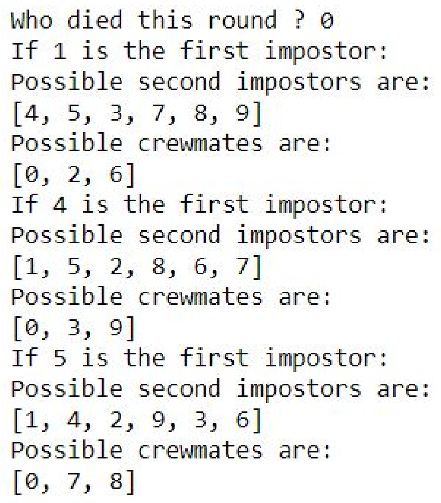
We will go through each possible first imposter and find the players that he has seen. Therefore, they cannot be the second imposter, we will call them possible crewmates. We will now list all the players that the possible crewmates have seen. These players are possible second imposters because the first imposter has not seen them.

This is a pseudo-code for the algorithm:

```
FOR each players that the murdered has seen
    Add players to possible first imposter
    FOR each players that the first imposter has seen
        Add players to innocent
        FOR each players that the innocent has seen
            Add players to probable second imposters
        Print probable imposters if current player is the first imposter
```

4. **Implement the algorithm and show a solution.**

After implementation of the solution here is an example

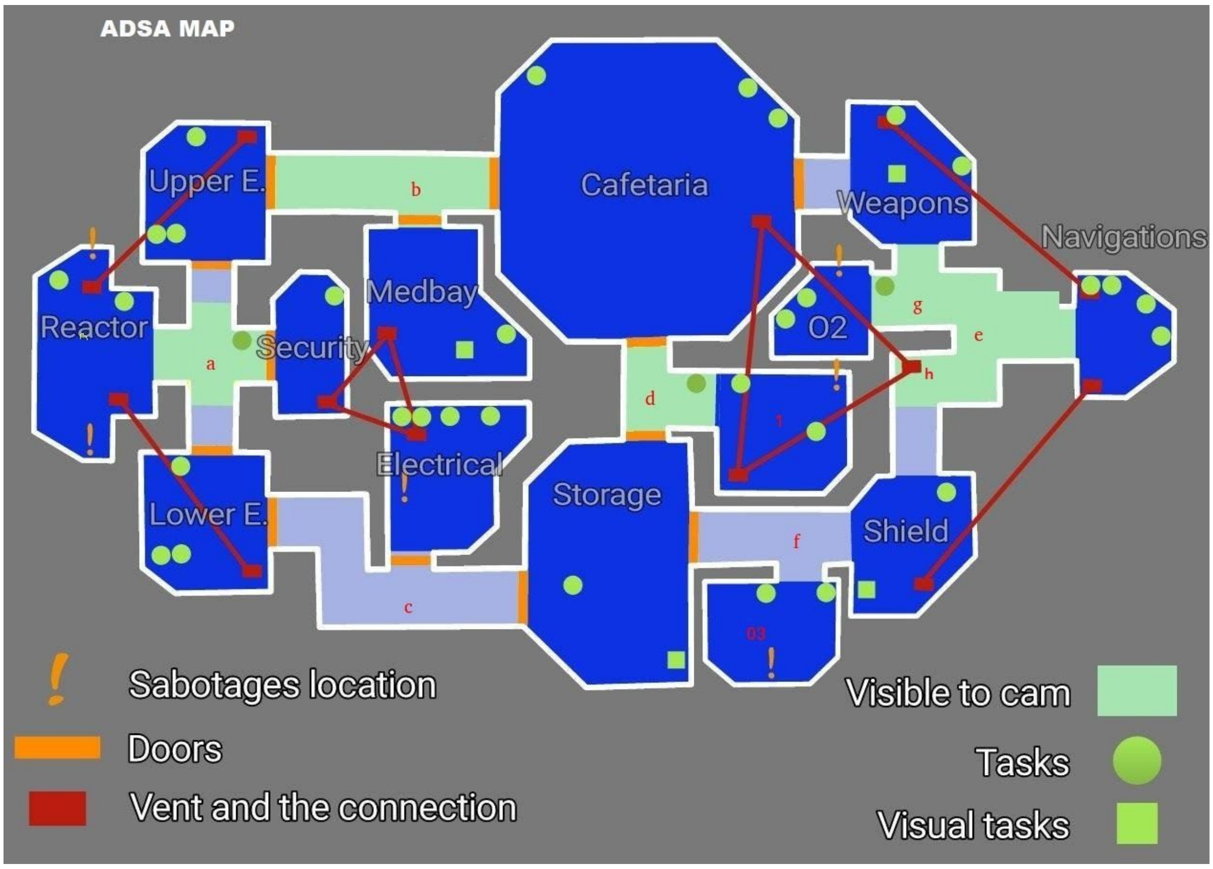


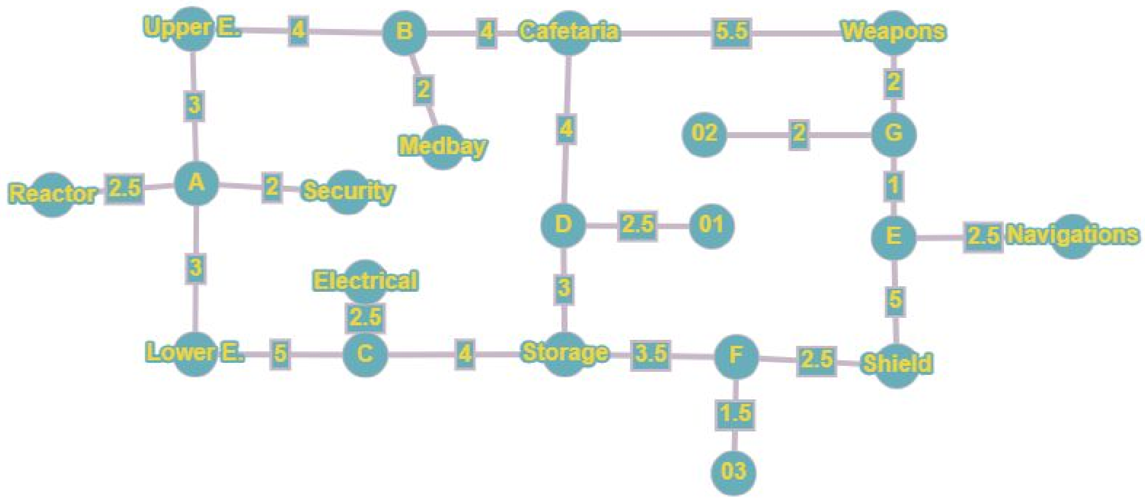
```
Who died this round ? 0
If 1 is the first impostor:
Possible second impostors are:
[4, 5, 3, 7, 8, 9]
Possible crewmates are:
[0, 2, 6]
If 4 is the first impostor:
Possible second impostors are:
[1, 5, 2, 8, 6, 7]
Possible crewmates are:
[0, 3, 9]
If 5 is the first impostor:
Possible second impostors are:
[1, 4, 2, 9, 3, 6]
Possible crewmates are:
[0, 7, 8]
```

Step 3: I don't see him, but I can give proof he vents!

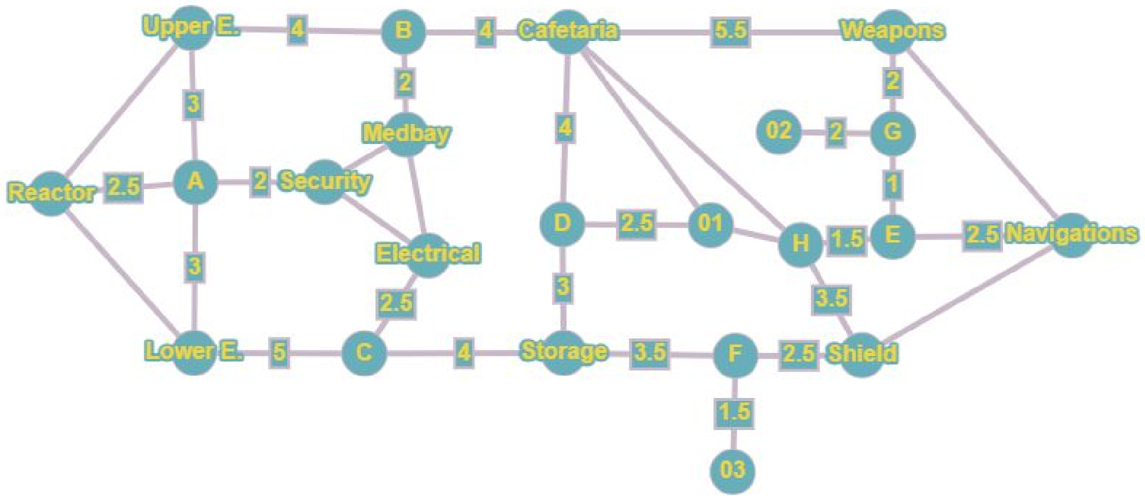
1. Present and argue about the two models of the map.

We have two types of maps, the first one corresponds to the map when the traveler is a Crewmate and the second one when the traveler is an impostor. When the latter is an impostor, he can use the ventilations to travel, the use of the ventilations does not imply any time, so the connection between two rooms does not take any time. To solve this problem, we had to convert the map into a graph with each room corresponding to a node. In addition, to simplify the graph we added nodes in the different corridors connecting the rooms. The nodes in the graphs are connected with weights corresponding to the centimeters on the map. Therefore, the time between two pieces can be recovered by calculating the sum of the vertices weights.





Graph when the traveler is a crewmate



Graph when the traveler is an impostor

The graphs are almost identical, the only difference is the addition for breakdowns. The vents do not add time, they correspond on the graph to branches without weights.

2. Argue about a pathfinding algorithm to implement.

In order to quickly know the distance between each pair of rooms we can use the Floyd-Warshall algorithm, in fact this algorithm calculates a cost matrix for each path between two rooms.

The use of other algorithms is however possible but it would take more time and complexity, it would be necessary to calculate the shortest trip piece by piece. For example, the use of the Dijkstra algorithm should have been overloaded with an iterator function for each piece of the map.

The Floyd Warshall algorithm finds all the shortest paths between pairs of nodes. It allows us to calculate these distances between two nodes in cases of graphs with direction or without direction. In our case, we are in the presence of a graph without direction.

```
FLOYD-WARSHALL($W$)
1. $n \leftarrow rows[W]$
2. $D^{(0)} \leftarrow W$
3. for $k \leftarrow 1$ to $n$
4.     do for $i \leftarrow 1$ to $n$
5.         do for $j \leftarrow 1$ to $n$
6.             $d_{ij}^{(k)} \leftarrow min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
7. return $D^{(n)}$
```

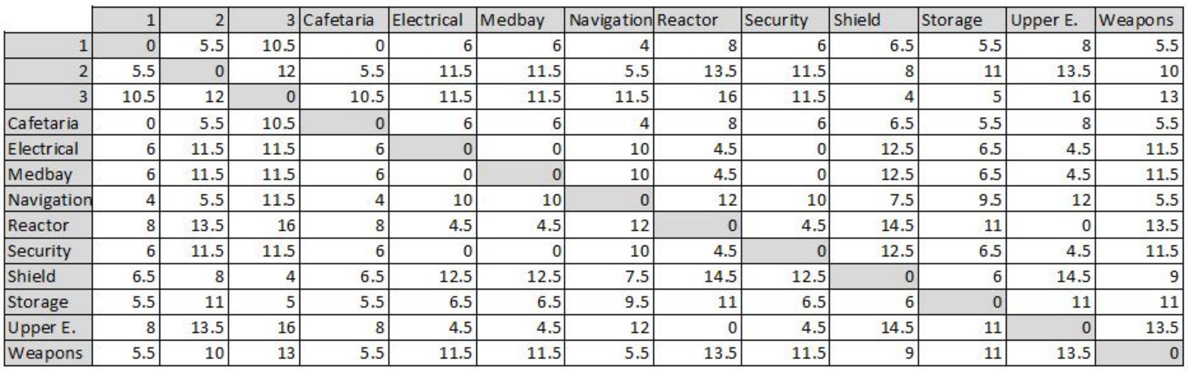
Floyd-Warshall algorithm (pseudo-code)

3. Implement the method and show the time to travel for any pair of rooms for both models

Thanks to the Floyd Warshal algorithm we recover a matrix corresponding to the distances between all the points in the graph. The computation is done for all pairs of nodes in the graph. If afterwards, the distances for all points for each pair of nodes are presented in table form, obtained directly with the use of the algorithm implemented in python.

	1	2	3	Cafetaria	Electrical	Medbay	Navigation	Reactor	Security	Shield	Storage	Upper E.	Weapons
1	0	16	10.5	6.5	12	12.5	17.5	20	19.5	11.5	5.5	14.5	12
2	16	0	12	9.5	21.5	15.5	5.5	23	22.5	8	15	17.5	26.5
3	10.5	12	0	12	11.5	18	11.5	19.5	19	4	5	20	17.5
Cafetaria	6.5	9.5	12	0	13.5	6	11	13.5	13	13	7	8	5.5
Electrical	12	21.5	11.5	13.5	0	19.5	20	13	12.5	12.5	6.5	13.5	19
Medbay	12.5	15.5	18	6	19.5	0	17	11.5	11	19	13	6	11.5
Navigation	17.5	5.5	11.5	11	20	17	0	24.5	24	7.5	13.5	19	5.5
Reactor	20	23	19.5	13.5	13	11.5	24.5	0	4.5	20.5	14.5	5.5	19
Security	19.5	22.5	19	13	12.5	11	24	4.5	0	20	14	5	18.5
Shield	11.5	8	4	13	12.5	19	7.5	20.5	20	0	6	21	18.5
Storage	5.5	15	5	7	6.5	13	13.5	14.5	14	6	0	15	12.5
Upper E.	14.5	17.5	20	8	13.5	6	19	5.5	5	21	15	0	13.5
Weapons	12	26.5	17.5	5.5	19	11.5	5.5	19	18.5	18.5	12.5	13.5	0

Distance between each pair of room where the traveler is a crewMate

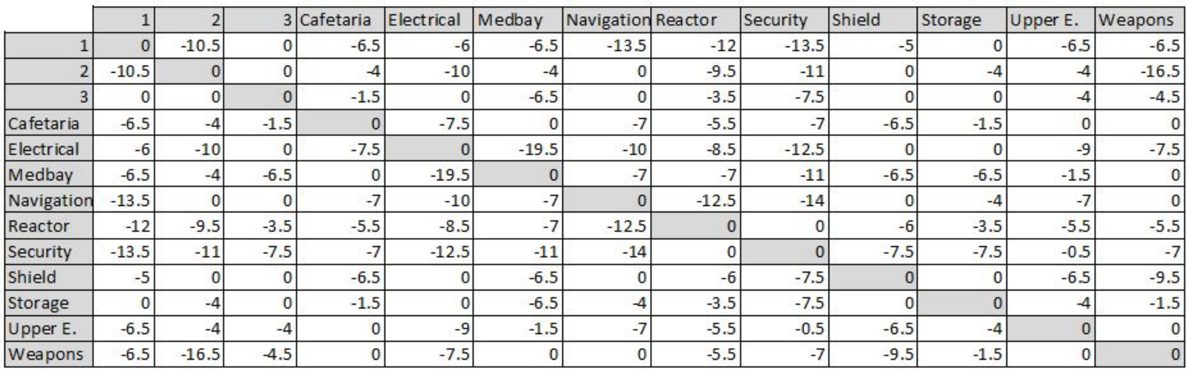


	1	2	3	Cafetaria	Electrical	Medbay	Navigation	Reactor	Security	Shield	Storage	Upper E.	Weapons
1	0	5.5	10.5	0	6	6	4	8	6	6.5	5.5	8	5.5
2	5.5	0	12	5.5	11.5	11.5	5.5	13.5	11.5	8	11	13.5	10
3	10.5	12	0	10.5	11.5	11.5	11.5	16	11.5	4	5	16	13
Cafetaria	0	5.5	10.5	0	6	6	4	8	6	6.5	5.5	8	5.5
Electrical	6	11.5	11.5	6	0	0	10	4.5	0	12.5	6.5	4.5	11.5
Medbay	6	11.5	11.5	6	0	0	10	4.5	0	12.5	6.5	4.5	11.5
Navigation	4	5.5	11.5	4	10	10	0	12	10	7.5	9.5	12	5.5
Reactor	8	13.5	16	8	4.5	4.5	12	0	4.5	14.5	11	0	13.5
Security	6	11.5	11.5	6	0	0	10	4.5	0	12.5	6.5	4.5	11.5
Shield	6.5	8	4	6.5	12.5	12.5	7.5	14.5	12.5	0	6	14.5	9
Storage	5.5	11	5	5.5	6.5	6.5	9.5	11	6.5	6	0	11	11
Upper E.	8	13.5	16	8	4.5	4.5	12	0	4.5	14.5	11	0	13.5
Weapons	5.5	10	13	5.5	11.5	11.5	5.5	13.5	11.5	9	11	13.5	0

Distance between each pair of room where the traveler is an Impostor

4. Display the interval of time for each pair of rooms where the traveler is an impostor.

In order to get the interval of time for each pair of rooms where the traveler is an impostor we just need to subtract the previous matrix.



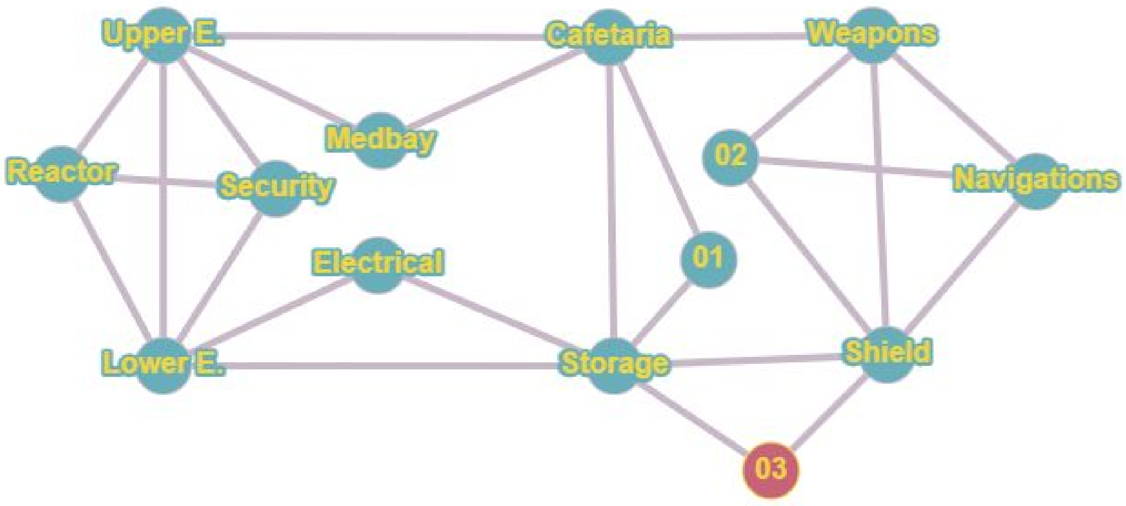
	1	2	3	Cafetaria	Electrical	Medbay	Navigation	Reactor	Security	Shield	Storage	Upper E.	Weapons
1	0	-10.5	0	-6.5	-6	-6.5	-13.5	-12	-13.5	-5	0	-6.5	-6.5
2	-10.5	0	0	-4	-10	-4	0	-9.5	-11	0	-4	-4	-16.5
3	0	0	0	-1.5	0	-6.5	0	-3.5	-7.5	0	0	-4	-4.5
Cafetaria	-6.5	-4	-1.5	0	-7.5	0	-7	-5.5	-7	-6.5	-1.5	0	0
Electrical	-6	-10	0	-7.5	0	-19.5	-10	-8.5	-12.5	0	0	-9	-7.5
Medbay	-6.5	-4	-6.5	0	-19.5	0	-7	-7	-11	-6.5	-6.5	-1.5	0
Navigation	-13.5	0	0	-7	-10	-7	0	-12.5	-14	0	-4	-7	0
Reactor	-12	-9.5	-3.5	-5.5	-8.5	-7	-12.5	0	0	-6	-3.5	-5.5	-5.5
Security	-13.5	-11	-7.5	-7	-12.5	-11	-14	0	0	-7.5	-7.5	-0.5	-7
Shield	-5	0	0	-6.5	0	-6.5	0	-6	-7.5	0	0	-6.5	-9.5
Storage	0	-4	0	-1.5	0	-6.5	-4	-3.5	-7.5	0	0	-4	-1.5
Upper E.	-6.5	-4	-4	0	-9	-1.5	-7	-5.5	-0.5	-6.5	-4	0	0
Weapons	-6.5	-16.5	-4.5	0	-7.5	0	0	-5.5	-7	-9.5	-1.5	0	0

Interval of time for each pair of rooms where the traveler is an impostor

Step 4: Secure the last tasks

1. Present and argue about the model of the map.

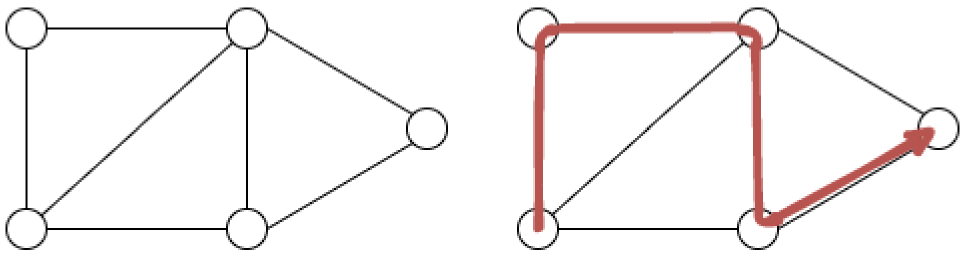
The map of Among Us can be converted to an unweighted graph. Each room corresponds to a node and each vertex corresponds to a path connecting each joint room.



Undirected graph of Among Us map

2. Thanks to a graph theory problem, present how to find a route passing through each room only one time.

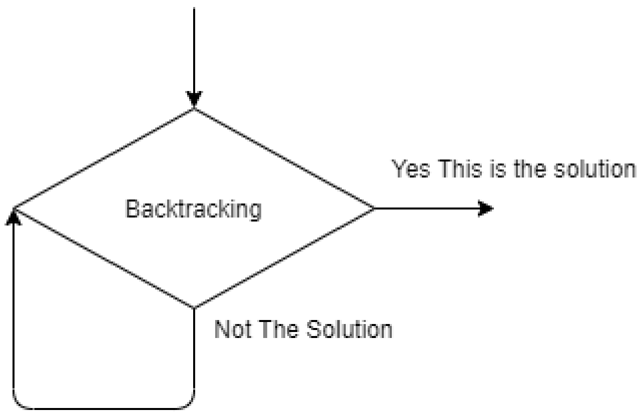
Our goal is to find the path through each node only once. The path passing through all the points of a graph only once is called hamiltonian path. The principle is simple, everyone knows this problem where you have to pass through each point only once per line and per point without lifting the pen. Our goal is to implement an algorithm that finds this path for our graph.



hamiltonian path

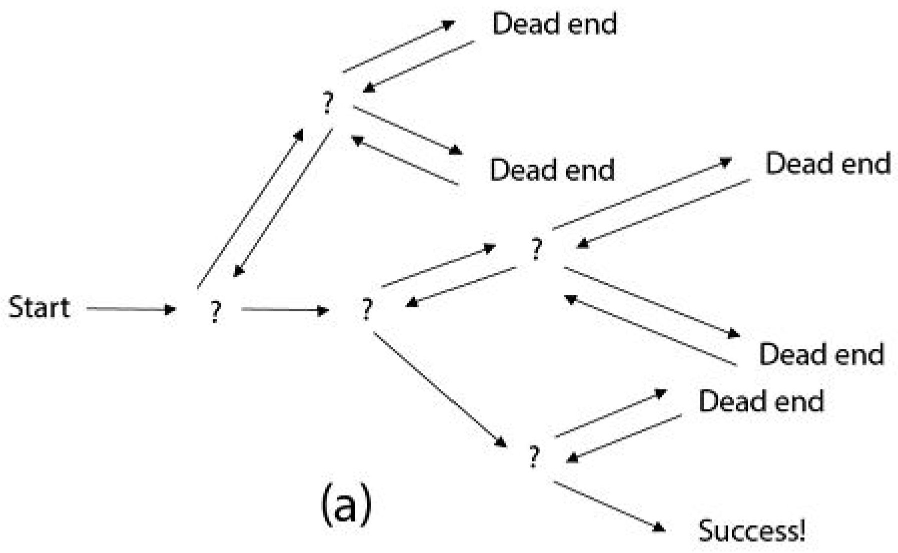
3. Argue about an algorithm solving your problem.

In order to solve this problem and to know the paths passing through the different points only once, we used a recursive backtracking algorithm. This same type of algorithm allows finding the path to solve a maze or to solve sudoku for example. The python implementation of this type of algorithm is very simple and allows in record time to have the path satisfying the required characteristics.



Main idea of the BackTracking algorithm

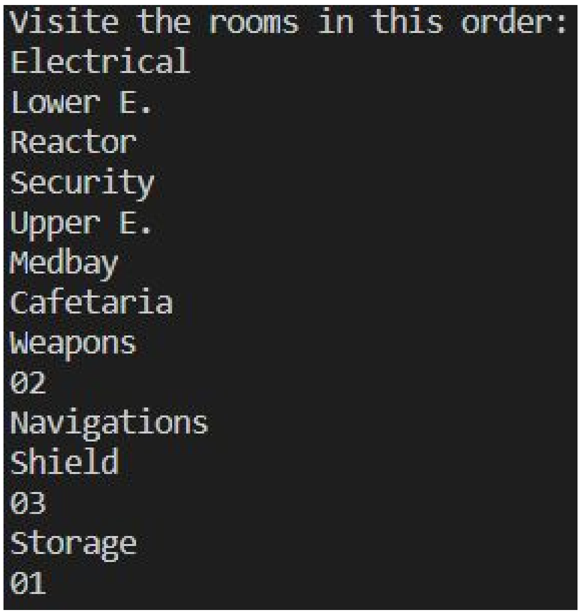
A backtracking algorithm is a recursive algorithm that at each iteration looks for a new path. If the algorithm finds a terminal branch that does not satisfy the required conditions, it backtracks and so on. Once the branch satisfying the conditions is found, the algorithm returns the values of these branches.



Main idea of the BackTracking algorithm

4. Implement the algorithm and show a solution.

A solution for this problem can be :



```
Visite the rooms in this order:
Electrical
Lower E.
Reactor
Security
Upper E.
Medbay
Cafetaria
Weapons
02
Navigations
Shield
03
Storage
01
```