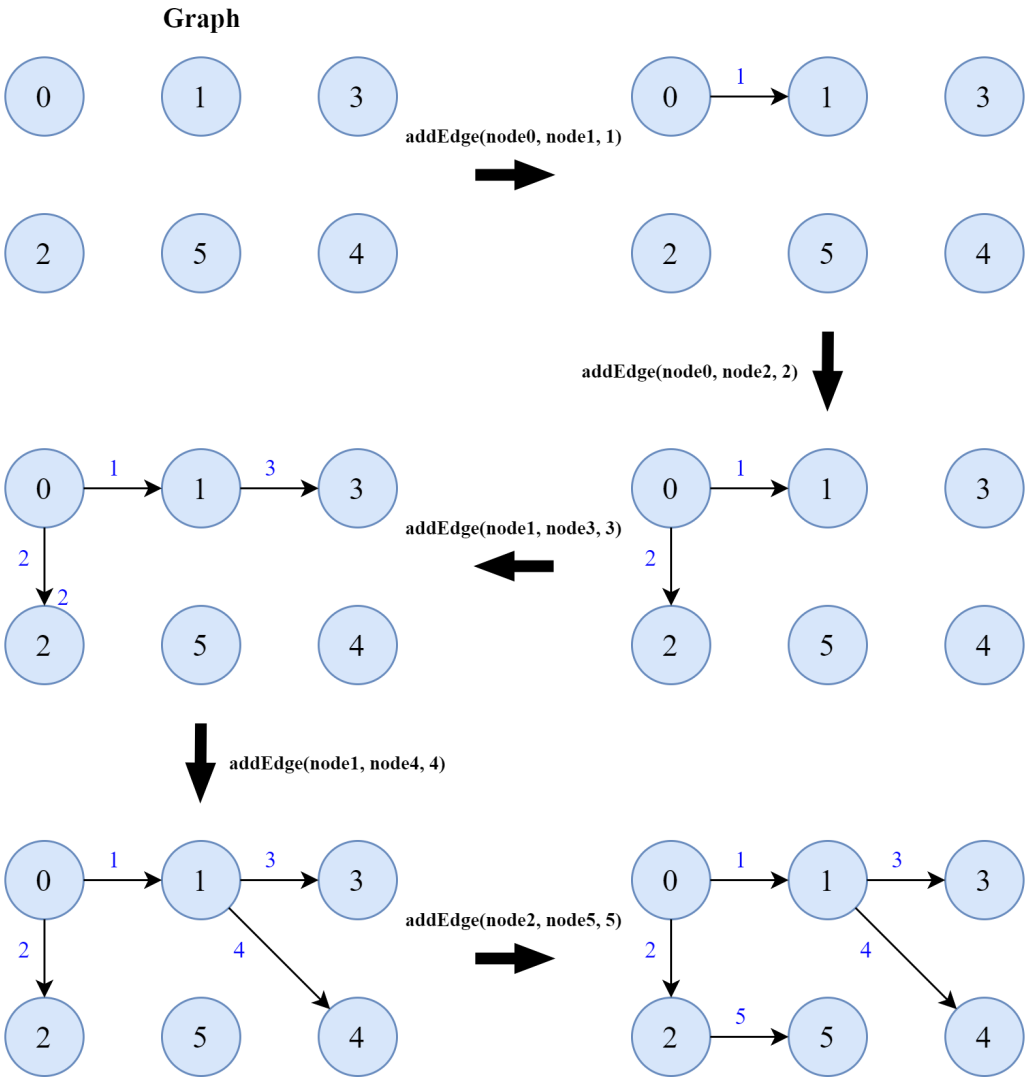
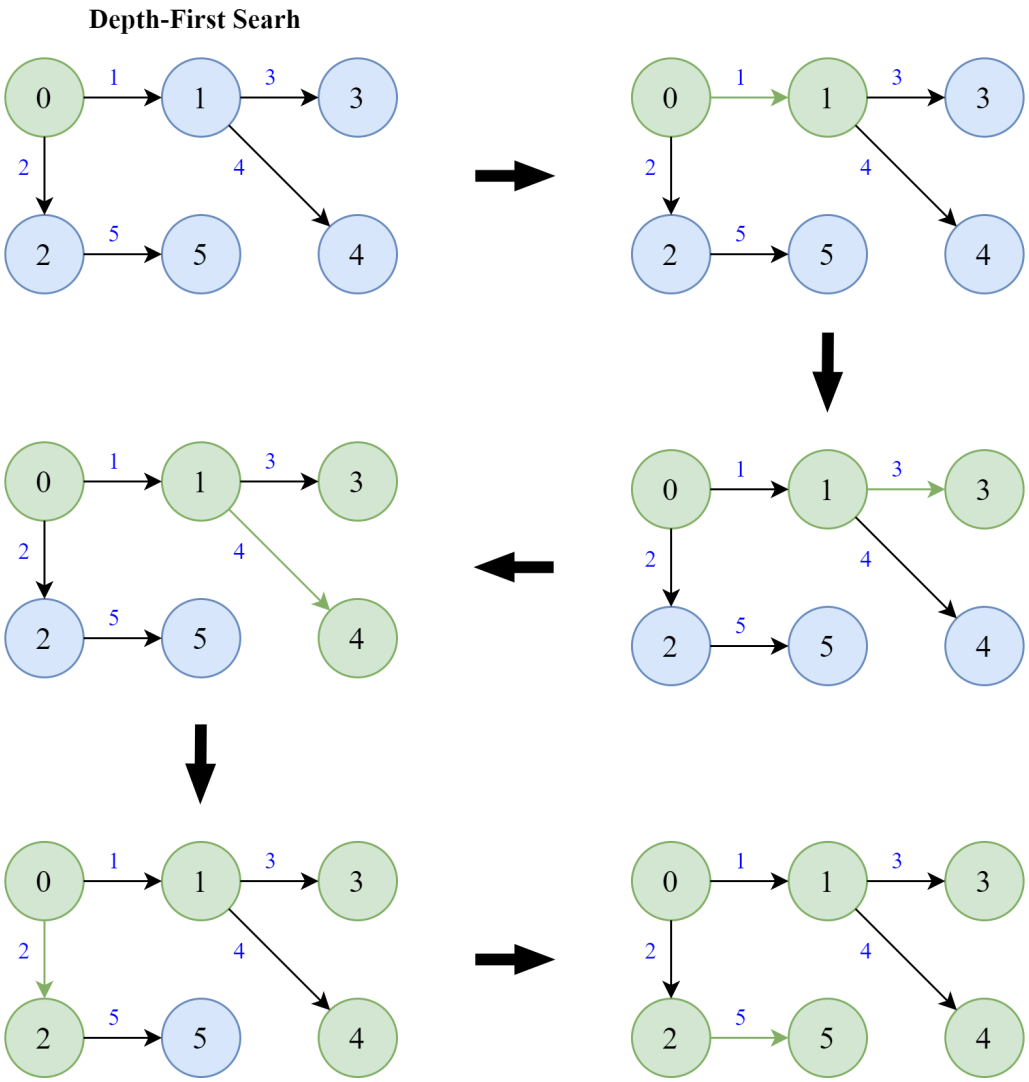
Q9.

In this problem, you need to implement a **depth-first search (DFS)** function that starts at the **first added node x** in the **directional graph** and traverses the entire **subgraph** which is connected to the node x. Your output should include: **1. the DFS traversal order** and **2. the total length of all edges in the subgraph**.

Please do not modify the template we have provided.

Hint:
The following is an example of the directional graph.
The order of adding edges is the same as the first sample input.





Input Format

The first line consists of an integer c, indicating the number of cases. In each case, the first line contains an integer m ($1 \leq m \leq 100$), representing the number of nodes in the graph. After that, there will be a single line containing m elements. The third line contains an integer n ($1 \leq n \leq 100$), representing the number of edges in the graph. Following that, there will be n lines, each containing three values (p1, p2, L), denoting the start node element, end node element, and the length of the edge. The **edges are directional**, pointing from the start node to the end node. It is assumed that the node elements will not be repeated. If the node in the new edge does not exist, add it to the graph.

Output Format

Start from the first added node and print the elements of all the nodes visited in the order they are traversed using depth-first search (DFS), enclosed in brackets. If a node is connected with multiple edges, traverse starting from the edge that was added first. Following that, enclose the total length of all edges in the subgraph in brackets as well.

Sample Input

```
2
6
0 1 2 3 4 5
5
0 1 1
0 2 2
1 3 3
1 4 4
2 5 5
6
1 2 3 4 6 9
4
1 3 1
1 6 10
1 9 100
2 4 1000
```

Sample Output

```
[0 1 3 4 2 5] [15]
[1 3 6 9] [111]
```