51m left

ALL

1

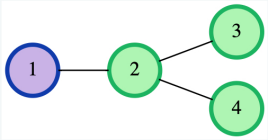
2

3

1. Delivery Management System

A manufacturing company is located in a certain city. Their goods need to be shipped to other cities that are connected with bidirectional roads, though some cities may not be accessible because roads don't connect to them. The order of deliveries is determined first by distance, then by priority. Given the number of cities, their connections via roads, and what city the manufacturing company is located in, determine the order of cities where the goods will be delivered.

For example, let's say that the number of cities is *cityNodes = 4*, where *cityFrom = [1, 2, 2]*, *cityTo = [2, 3, 4]*, and *company = 1*. *In other words, the manufacturing company is located in city 1, and the roads run between cities 1 and 2, cities 2 and 3, and cities 2 and 4, like so:*



In this case, the cities would be visited based on the following logic:

- The closest city (or cities) is visited first. This is city 2, which is 1 unit from the manufacturing company.
- The next-closest city (or cities) is visited next. This is city 3 and city 4, which are both 2 units from the manufacturing company.
  - In this case, priority is then calculated, visiting the smaller-numbered city first (city 3) and continuing in ascending order (city 4).

Therefore, the order is [2, 3, 4], which is the answer you would return.

Function Description
Complete the function *order* in the editor below.

order has the following parameters:
int *cityNodes:* the number of cities
int *cityFrom[n]:* an array of integers denoting the first city node where there is a bidirectional node
int *cityTo[n]:* an array of integers denoting the second city node where there is a bidirectional node
int *company:* employee who invites all other employees, the node where the route starts
Returns:
int[]: an array of integers denoting the cities where the goods will be delivered in the order they will be delivered

Constraints

- $2 \le cityNodes \le 10^5$
- $1 \le n \le \min(^{(cityNodes \times (cityNodes - 1))}/_{2}, 10^5)$
- $1 \le cityFrom[i], cityTo[i], company \le n$
- $cityFrom[i] \ne cityTo[i]$

▶ **Input Format For Custom Testing**

▼ **Sample Case 0**

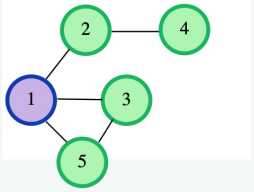
Sample Input For Custom Testing

```
STDIN     Function
-----     --------
5 5   →   cityNodes = 5, n = 5
1 2   →   cityFrom = 1, cityTo = 2
1 3   →   cityFrom = 1, cityTo = 3
2 4   →   cityFrom = 2, cityTo = 4
3 5   →   cityFrom = 3, cityTo = 5
1 5   →   cityFrom = 1, cityTo = 5
1     →   company = 1
```

Sample Output

```
2
3
5
4
```

Explanation



Cities 2, 3, and 5 are all 1 unit of distance away from the manufacturing company. These are visited based on priority in ascending order, so [2, 3, 5]. City 4 is 2 units of distance away from the manufacturing company, so it is visited next. Therefore, the final order is [2, 3, 5, 4].

Language Python 3

Autocomplete Ready

```
1  #!/bin/python3 ...
10
11 #
12 # Complete the 'order' function below.
13 #
14 # The function is expected to return an INTEGER_ARRAY.
15 # The function accepts following parameters:
16 #  1. UNWEIGHTED_INTEGER_GRAPH city
17 #  2. INTEGER company
18 #
19
20 #
21 # For the unweighted graph, <name>:
22 #
23 # 1. The number of nodes is <name>_nodes.
24 # 2. The number of edges is <name>_edges.
25 # 3. An edge exists between <name>_from[i] and <name>_to
   [i].
26 #
27 #
28
29 def order(city_nodes, city_from, city_to, company):
30     # Write your code here
31
32 if __name__ == '__main__': ...
```

Line: 31 Col: 1

Test Results | Custom Input | Run Code | Run Tests | Submit

▼ Sample Case 1

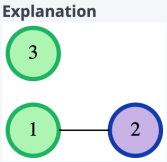
Sample Input For Custom Testing

```
STDIN       Function
-----       --------
3 1     →   cityNodes = 3, n = 1
1 2     →   cityFrom = 1, cityTo = 2
2       →   company = 2
```

Sample Output

```
1
```

Explanation



City 1 is located 1 unit of distance away from the manufacturing company. City 3 is not accessible because there are no roads connecting it to the manufacturing company's city. Therefore, the answer is [1].