Step 1: Solve this problem manually using this procedure.

Cycle 1

A	0	1	∞	2	∞
	A	B	C	D	E
B	0	1	3	2	∞
	A	B	C	D	E
C	0	1	3	2	11
	A	B	C	D	E
D	0	1	3	2	5
	A	B	C	D	E
E	0	1	3	2	5
	A	B	C	D	E

Cycle 2

A	0	1	3	2	5
	A	B	C	D	E
B	0	1	3	2	5
	A	B	C	D	E
C	0	1	3	2	5
	A	B	C	D	E
D	0	1	3	2	5
	A	B	C	D	E
E	0	1	3	2	5
	A	B	C	D	E

The process ends at Cycle 2 because none of the vertices is changed. The shortest path is A-B-D-C-E and the minimum time it takes for all the nodes to receive the signal is 5.

Step 2: Study 743. Network Delay Time 50.3 (local copy)- Medium

Step 3: Ask ChatGPT: "How to use Bellman-Ford's Algorithm to implement a Python code to solve the LeetCode question: '743. Network Delay Time', with the test data: 'Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2; Output: 2'"

To solve the LeetCode question '743. Network Delay Time' using the Bellman-Ford algorithm, we can modify the algorithm to find the shortest time it takes for all nodes to receive the signal from a given source node.

Here's the Python code that implements the Bellman-Ford algorithm for the given test data:

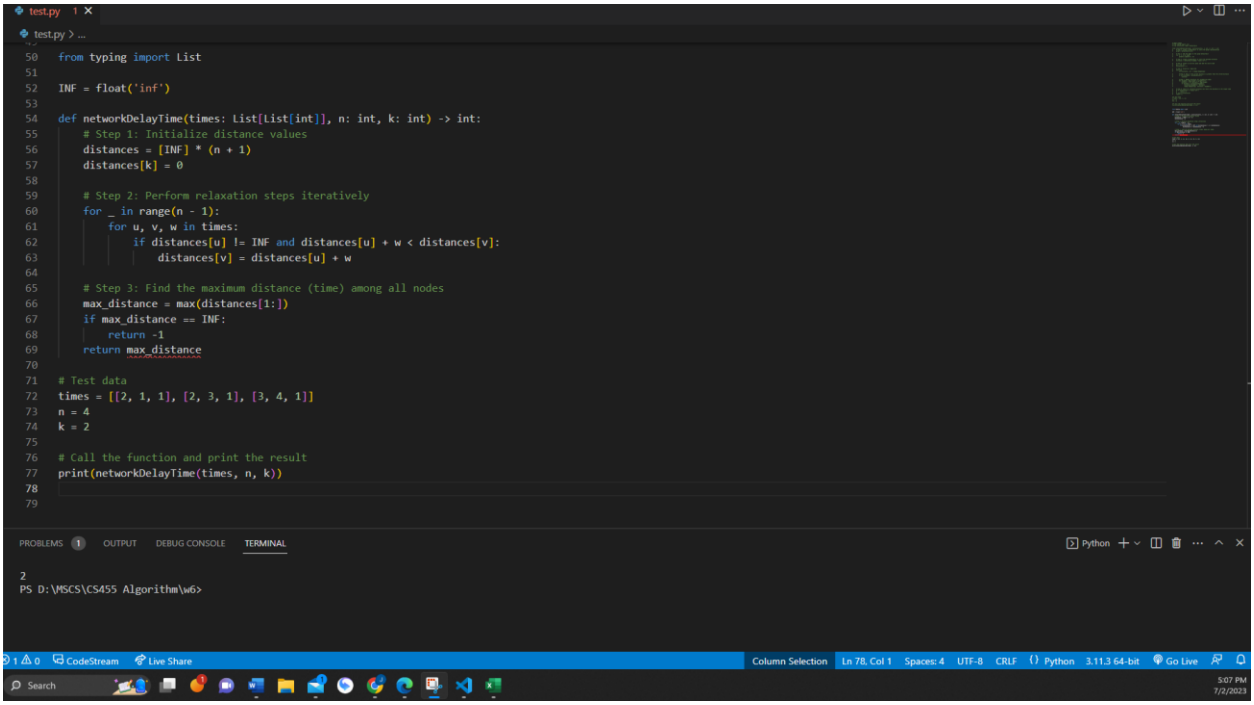
python Copy code

```python
from typing import List

INF = float('inf')

def networkDelayTime(times: List[List[int]], n: int, k: int) -> int:
    # Step 1: Initialize distance values
    distances = [INF] * (n + 1)
    distances[k] = 0

    # Step 2: Perform relaxation steps iteratively
    for _ in range(n - 1):
        for u, v, w in times:
            if distances[u] != INF and distances[u] + w < distances[v]:
                distances[v] = distances[u] + w

    # Step 3: Find the maximum distance (time) among all nodes
    max_distance = max(distances[1:])
    if max_distance == INF:
        return -1
    return max_distance
```

```python
# Test data
times = [[2, 1, 1], [2, 3, 1], [3, 4, 1]]
n = 4
k = 2

# Call the function and print the result
print(networkDelayTime(times, n, k))
```

When you run this code, it will output `2`, which is the expected result for the given test data. The code applies the Bellman-Ford algorithm to find the shortest time it takes for all nodes to receive the signal from the given source node. If it is impossible for all nodes to receive the signal, the code returns `-1`.

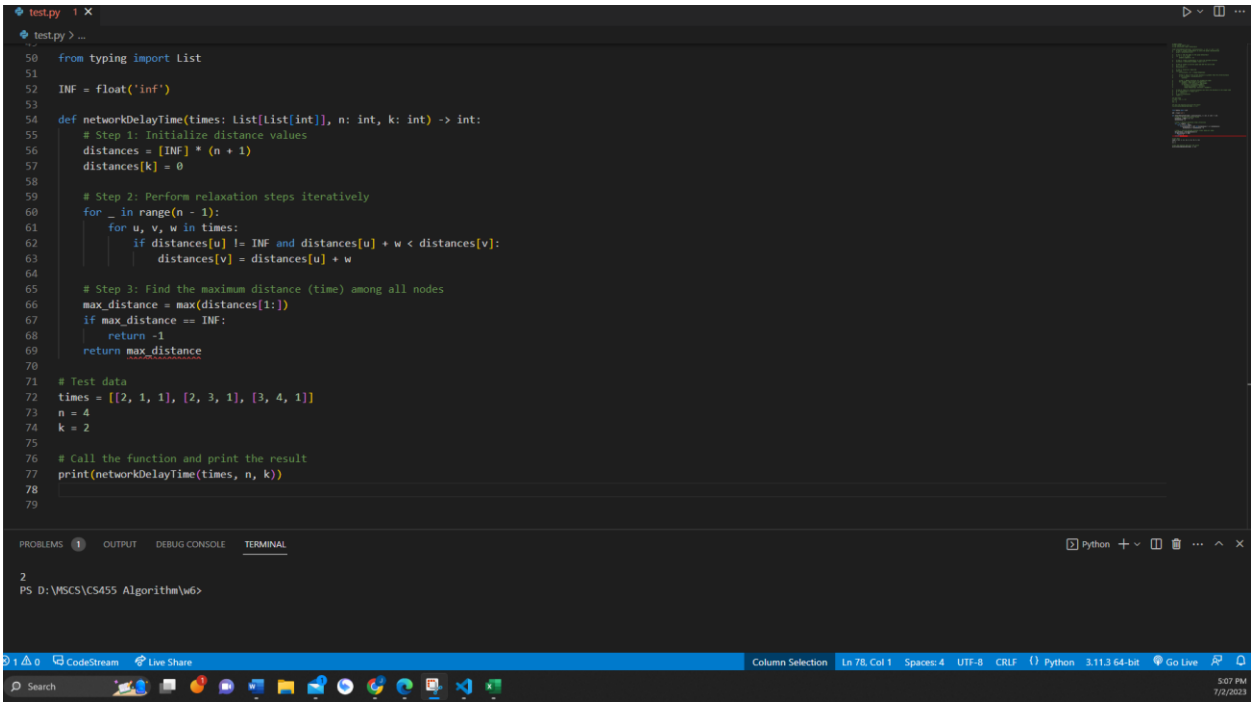
Step 4: Implement the Python code created by ChatGPT



```python
from typing import List

INF = float('inf')

def networkDelayTime(times: List[List[int]], n: int, k: int) -> int:
    # Step 1: Initialize distance values
    distances = [INF] * (n + 1)
    distances[k] = 0

    # Step 2: Perform relaxation steps iteratively
    for _ in range(n - 1):
        for u, v, w in times:
            if distances[u] != INF and distances[u] + w < distances[v]:
                distances[v] = distances[u] + w

    # Step 3: Find the maximum distance (time) among all nodes
    max_distance = max(distances[1:])
    if max_distance == INF:
        return -1
    return max_distance

# Test data
times = [[2, 1, 1], [2, 3, 1], [3, 4, 1]]
n = 4
k = 2

# Call the function and print the result
print(networkDelayTime(times, n, k))
```

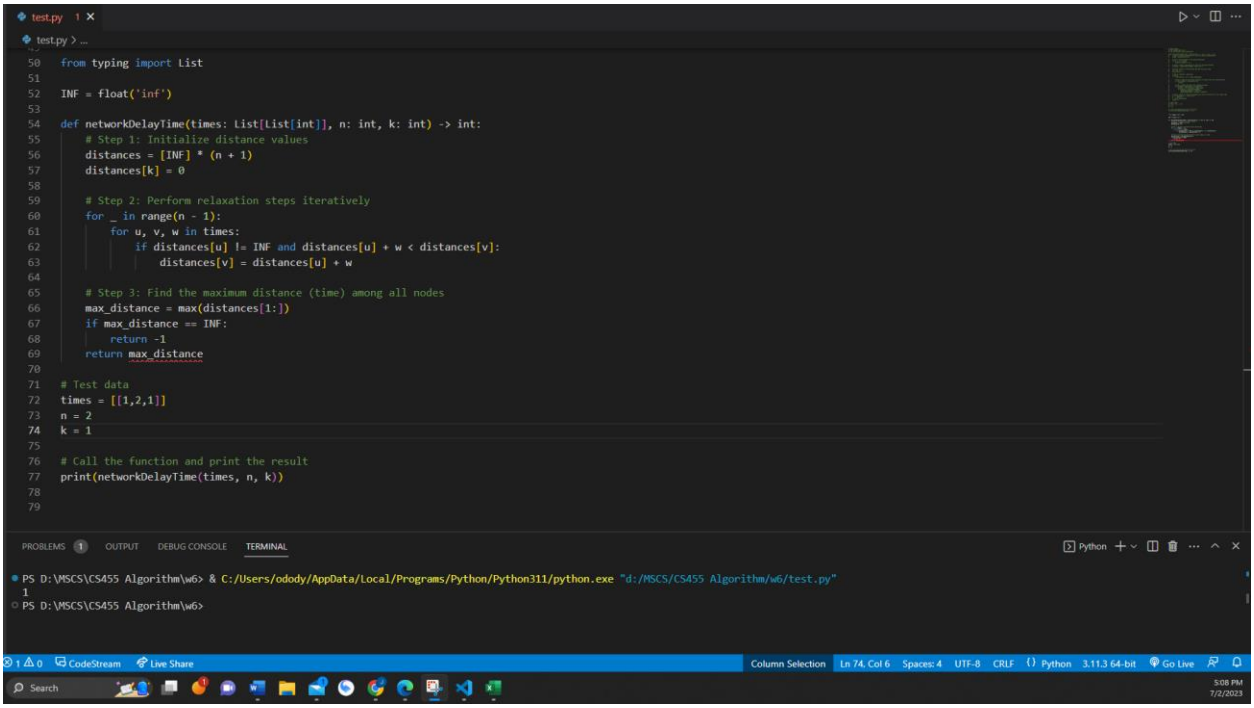
```
2
PS D:\MSCS\CS455 Algorithm\w6>
```

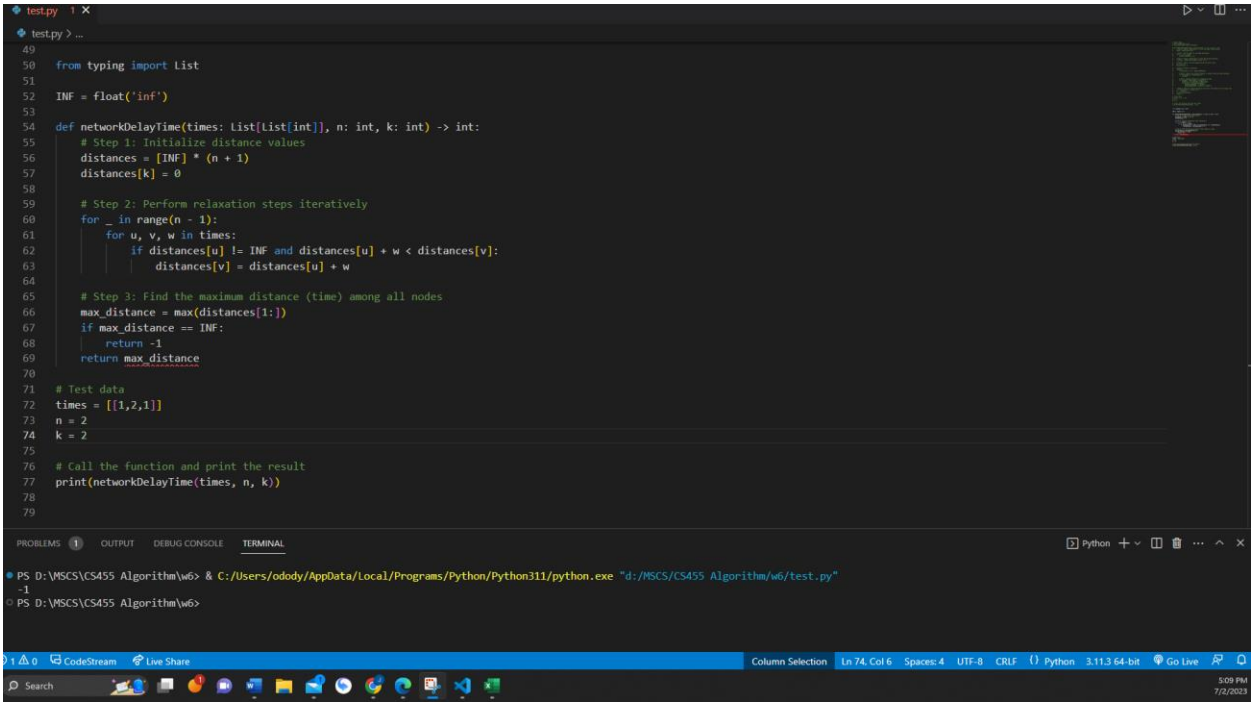
Step 4: Test the Python code with all the test cases provided by 743. Network Delay Time 50.3 (local copy)- Medium



```python
from typing import List

INF = float('inf')

def networkDelayTime(times: List[List[int]], n: int, k: int) -> int:
    # Step 1: Initialize distance values
    distances = [INF] * (n + 1)
    distances[k] = 0

    # Step 2: Perform relaxation steps iteratively
    for _ in range(n - 1):
        for u, v, w in times:
            if distances[u] != INF and distances[u] + w < distances[v]:
                distances[v] = distances[u] + w

    # Step 3: Find the maximum distance (time) among all nodes
    max_distance = max(distances[1:])
    if max_distance == INF:
        return -1
    return max_distance

# Test data
times = [[2, 1, 1], [2, 3, 1], [3, 4, 1]]
n = 4
k = 2

# Call the function and print the result
print(networkDelayTime(times, n, k))
```

```
2
PS D:\MSCS\CS455 Algorithm\w6>
```


test.py 1
test.py > ...
50 from typing import List
51
52 INF = float('inf')
53
54 def networkDelayTime(times: List[List[int]], n: int, k: int) -> int:
55 # Step 1: Initialize distance values
56 distances = [INF] * (n + 1)
57 distances[k] = 0
58
59 # Step 2: Perform relaxation steps iteratively
60 for _ in range(n - 1):
61 for u, v, w in times:
62 if distances[u] != INF and distances[u] + w < distances[v]:
63 distances[v] = distances[u] + w
64
65 # Step 3: Find the maximum distance (time) among all nodes
66 max_distance = max(distances[1:])
67 if max_distance == INF:
68 return -1
69 return max_distance
70
71 # Test data
72 times = [[1,2,1]]
73 n = 2
74 k = 1
75
76 # Call the function and print the result
77 print(networkDelayTime(times, n, k))
78
79
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
Python
PS D:\MSCS\CS455 Algorithm\w6> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w6/test.py"
1
PS D:\MSCS\CS455 Algorithm\w6>
1 0 CodeStream Live Share
Column Selection Ln 74, Col 6 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live
Search
5:08 PM
7/2/2023


test.py 1
test.py > ...
49
50 from typing import List
51
52 INF = float('inf')
53
54 def networkDelayTime(times: List[List[int]], n: int, k: int) -> int:
55 # Step 1: Initialize distance values
56 distances = [INF] * (n + 1)
57 distances[k] = 0
58
59 # Step 2: Perform relaxation steps iteratively
60 for _ in range(n - 1):
61 for u, v, w in times:
62 if distances[u] != INF and distances[u] + w < distances[v]:
63 distances[v] = distances[u] + w
64
65 # Step 3: Find the maximum distance (time) among all nodes
66 max_distance = max(distances[1:])
67 if max_distance == INF:
68 return -1
69 return max_distance
70
71 # Test data
72 times = [[1,2,1]]
73 n = 2
74 k = 2
75
76 # Call the function and print the result
77 print(networkDelayTime(times, n, k))
78
79
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
Python
PS D:\MSCS\CS455 Algorithm\w6> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w6/test.py"
-1
PS D:\MSCS\CS455 Algorithm\w6>
1 0 CodeStream Live Share
Column Selection Ln 74, Col 6 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live
Search
5:09 PM
7/2/2023