

CSES Problem Set

Message Route

TASK | SUBMIT | RESULTS | STATISTICS | TESTS | QUEUE

Submission details

Task:	<u>Message Route</u>	
Sender:	tonykk	
Submission time:	2024-11-10 11:59:27 +0200	
Language:	Python3 (CPython3)	
Status:	READY	
Result:	ACCEPTED	

Test results

test	verdict	time	
#1	ACCEPTED	0.02 s	<u>>></u>
#2	ACCEPTED	0.02 s	<u>>></u>
#3	ACCEPTED	0.02 s	<u>>></u>
#4	ACCEPTED	0.02 s	<u>>></u>
#5	ACCEPTED	0.02 s	<u>>></u>
#6	ACCEPTED	0.77 s	<u>>></u>
#7	ACCEPTED	0.77 s	<u>>></u>
#8	ACCEPTED	0.78 s	<u>>></u>
#9	ACCEPTED	0.76 s	<u>>></u>
#10	ACCEPTED	0.71 s	<u>>></u>
#11	ACCEPTED	0.02 s	<u>>></u>
#12	ACCEPTED	0.51 s	<u>>></u>

Code -

```
from collections import deque, defaultdict
2
3
4 def legrovidebb_ut_keresese():
5
       # Adatok beolvasása
       n, m = map(int, input().split())
6
7
8
       # Gráf inicializálása
       graf = defaultdict(list)
9
10
       for _ in range(m):
           a, b = map(int, input().split())
11
           graf[a].append(b)
12
13
           graf[b].append(a)
14
15
       # inicializálás (BFS)
16
       sor = deque([1])
17
       tavolsag = {1: 1} # Az 1-es csúcsról indulunk, távolsá
```

Graph Algorithms

Counting Rooms	_
Labyrinth	_
Building Roads	_
Message Route	✓
Building Teams	_
Round Trip	_
Monsters	_
Shortest Routes I	_
•••	

Your submissions

2024-11-10	11:59:27	~
2024-11-10	11:15:11	~
2024-11-10	11:13:50	×
2024-11-07	10:07:37	~
2024-11-07	10:03:59	×

```
elozo_csucs = {1: None} # Az út visszakövetéséhez
18
19
20
       # BFS ciklus
21
       while sor:
22
           csucs = sor.popleft()
23
24
           # Ha elérjük a célt (az n-edik csúcsot), akkor megá
           if csucs == n:
25
               break
26
27
           # Szomszédok bejárása
28
29
           for szomszed in graf[csucs]:
               if szomszed not in tavolsag: # Még nem jártunk
30
31
                   tavolsag[szomszed] = tavolsag[csucs] + 1
32
                   elozo_csucs[szomszed] = csucs
33
                   sor.append(szomszed)
34
35
       # Ellenőrizzük, hogy elértük-e az n-edik csúcsot
       if n not in tavolsag:
36
37
           print("IMPOSSIBLE")
38
           return
39
       # Útvonal visszakövetése az 1-estől az n-edikig
40
       utvonal = []
41
42
       jelenlegi_csucs = n
43
       while jelenlegi_csucs is not None:
44
           utvonal.append(jelenlegi_csucs)
45
           jelenlegi_csucs = elozo_csucs[jelenlegi_csucs]
46
47
       utvonal.reverse() # Az út visszafelé van, így megfordí
       print(tavolsag[n]) # Az út hossza
48
49
       print(" ".join(map(str, utvonal))) # Az út kiírása
50
51
52
53 legrovidebb ut keresese()
```

SHARE CODE TO OTHERS

Test details -

Test 1

Verdict: ACCEPTED

```
input

10 20
8 9
6 7
9 10
```

```
    correct output

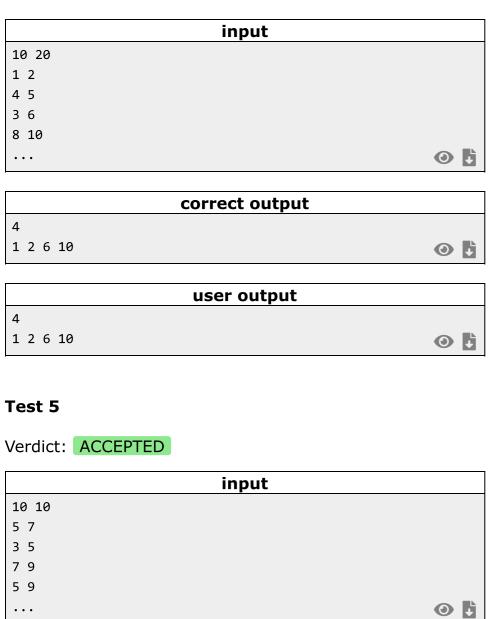
    5

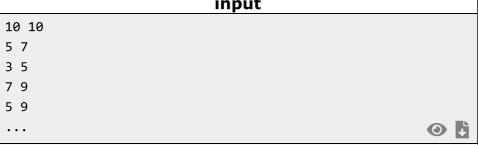
    1 4 5 8 10
```

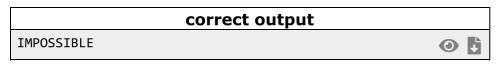
user output 1 4 5 8 10 **O** Test 2 Verdict: ACCEPTED input 10 20 3 4 7 8 3 6 2 4 **O** correct output 1 2 4 7 10 **O** user output 1 2 4 7 10 **O** Test 3 Verdict: ACCEPTED input 10 20 2 4 8 10 2 5 8 9 **O** correct output 1 2 5 7 10 **②** user output 1 2 5 7 10 **O**

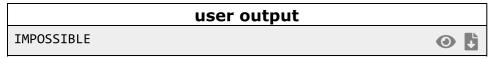
Test 4

Verdict: ACCEPTED



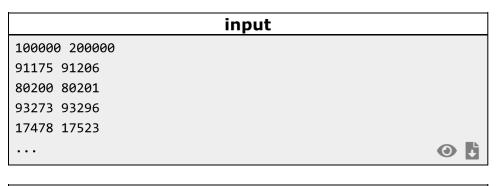




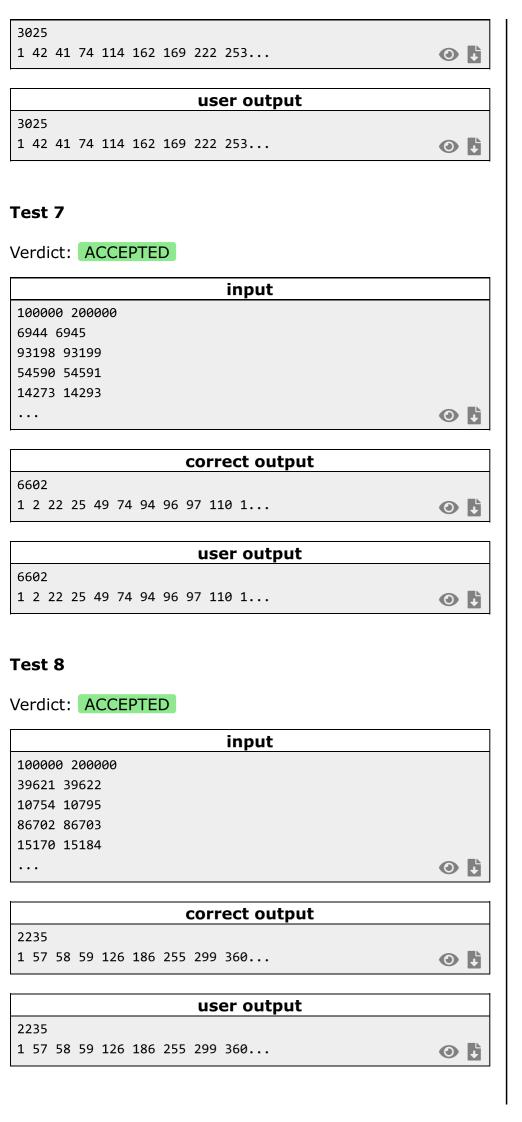


Test 6

Verdict: ACCEPTED



correct output



Test 9 Verdict: ACCEPTED input 100000 200000 89513 89514 21947 21954 76082 76083 15033 15034 0 correct output 3702 1 2 32 33 77 76 107 139 140 16... **O** user output 3702 1 2 32 33 77 76 107 139 140 16... **O** Test 10 Verdict: ACCEPTED input 100000 200000 44678 44704 84851 84861 16615 16641 32838 32876 **O** correct output **IMPOSSIBLE O** user output **IMPOSSIBLE O** Test 11

Verdict: ACCEPTED

input	
2 1	
1 2	©

correct output

