1101. The Earliest Moment When Everyone Become Friends

There are n people in a social group labeled from `0` to `n - 1`. You are given an array `logs` where `logs[i] = [timestamp, xi, yi]` indicates that `xi` and `yi` will be friends at the time `timestamp`.

Friendship is symmetric. That means if `a` is friends with `b`, then `b` is friends with `a`. Also, person `a` is acquainted with a person `b` if `a` is friends with `b`, or `a` is a friend of someone acquainted with `b`.

Return *the earliest time for which every person became acquainted with every other person*. If there is no such earliest time, return `-1`.

Example 1:

```
Input: logs =
[[20190101,0,1],[20190104,3,4],[20190107,2,3],[20190211,1,5],[20190224,2,4],[20
190301,0,3],[20190312,1,2],[20190322,4,5]], n = 6
Output: 20190301
Explanation:
The first event occurs at timestamp = 20190101 and after 0 and 1 become friends
we have the following friendship groups [0,1], [2], [3], [4], [5].
The second event occurs at timestamp = 20190104 and after 3 and 4 become
friends we have the following friendship groups [0,1], [2], [3,4], [5].
The third event occurs at timestamp = 20190107 and after 2 and 3 become friends
we have the following friendship groups [0,1], [2,3,4], [5].
The fourth event occurs at timestamp = 20190211 and after 1 and 5 become
friends we have the following friendship groups [0,1,5], [2,3,4].
The fifth event occurs at timestamp = 20190224 and as 2 and 4 are already
friends anything happens.
The sixth event occurs at timestamp = 20190301 and after 0 and 3 become friends
we have that all become friends.
```

Example 2:

```
Input: logs = [[0,2,0],[1,0,1],[3,0,3],[4,1,2],[7,3,1]], n = 4
Output: 3
```

Constraints:

- `2 <= n <= 100`
- `1 <= logs.length <= 10`^{4}
- `logs[i].length == 3`
- `0 <= timestamp <= 10`^{9}
- `0 <=` x_i, y_i `<= n - 1`
- x_i `!=` y_i
- All the values of `timestamp` are unique.
- All the pairs (x_i, y_i) occur at most one time in the input.

Code:

```
class Solution:
    def earliestAcq(self, logs: List[List[int]], n: int) -> int:

        rank = [0]*101
        parent = [i for i in range (101)]
        component = n
        def find(x):
            if parent[x]==x:
                return x
            else:
                parent[x] = find(parent[x])
                return parent[x]

        def union(x,y):
            a = find(x)
            b = find(y)
            if (a==b):
                return False
            else:
```

```
        if rank[a]>rank[b]:
            parent[b]=a
        elif rank[b]>rank[a]:
            parent[a]=b
        else:
            parent[b]=a
            rank[a]+=1
        return True
count = 0
ans = []

logs = sorted(logs)



for log in logs:
    time, u, v = log
    if union(u,v):
        ans = time
        component-=1

if component > 1:
    return -1
else:
    return (ans)
```