LC 332. Reconstruct Itinerary

Question

Given a list of airline tickets represented by pairs of departure and arrival airports `[from, to]`, reconstruct the itinerary in order. All of the tickets belong to a man who departs from `JFK`. Thus, the itinerary must begin with `JFK`.

Note:

1. If there are multiple valid itineraries, you should return the itinerary that has the smallest lexical order when read as a single string. For example, the itinerary `["JFK", "LGA"]` has a smaller lexical order than `["JFK", "LGB"]`.
2. All airports are represented by three capital letters (IATA code).
3. You may assume all tickets form at least one valid itinerary.

Example 1:

```
Input: [["MUC", "LHR"], ["JFK", "MUC"], ["SFO", "SJC"], ["LHR", "SFO"]]
Output: ["JFK", "MUC", "LHR", "SFO", "SJC"]
```

Example 2:

```
Input: [["JFK","SFO"],["JFK","ATL"],["SFO","ATL"],["ATL","JFK"],["ATL","SFO"]]
Output: ["JFK","ATL","JFK","SFO","ATL","SFO"]
Explanation: Another possible reconstruction is ["JFK","SFO","ATL","JFK","ATL","SFO"].
             But it is larger in lexical order.
```

Solution

```python
class Solution:
    def findItinerary(self, tickets: List[List[str]]) -> List[str]:
        #Solution 2
        #defaultdict(<class 'list'>, {'SFO': ['ATL'], 'JFK': ['SFO', 'ATL'],
'ATL': ['SFO', 'JFK']})
        #res列表:
        #['SFO']
        #['SFO', 'ATL']
        #['SFO', 'ATL', 'SFO']
        #['SFO', 'ATL', 'SFO', 'JFK']
        #['SFO', 'ATL', 'SFO', 'JFK', 'ATL']
        #['SFO', 'ATL', 'SFO', 'JFK', 'ATL', 'JFK']
        graph = collections.defaultdict(list)
        for s, t in sorted(tickets, reverse=True):
            graph[s].append(t)
        res = []
        def query(s):
            while graph[s]:
```

```
                query(graph[s].pop())
            res.append(s)
        query("JFK")
        return res[::-1]

        #Solution
        graph = collections.defaultdict(list)
        for frm, to in tickets:
            graph[frm].append(to)
        for frm, tos in graph.items():
            tos.sort(reverse=True)
        res = []
        self.dfs(graph, "JFK", res)
        return res[::-1]

    def dfs(self, graph, source, res):
        while graph[source]:
            v = graph[source].pop()
            self.dfs(graph, v, res)
        res.append(source)
```