[from, to]로 구성된 항공권 목록을 이용해 JFK에서 출발하는 여행 일정을 구성하라. 여러 일정이 있는 경우 사전 어휘순으로 방문한다.

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

1.dfs
```python
class Solution:
    def findItinerary(self, tickets: List[List[str]]) -> List[str]:
        def dfs(cur, nex):
            if len(result) > 0:
                return
            if len(nex) == 0:
                path = []
                for i in range(len(cur)):
                    path.append(cur[i][1])
                result.append(path)
                return
            for i, n in enumerate(nex):
                if n[0] == cur[-1][1]:
                    _cur = cur[:]
                    _nex = nex[:]
                    _cur.append(_nex.pop(i))
                    dfs(_cur, _nex)

        result = []
        tickets.sort(key = lambda t: (t[0], t[1]))
        dfs([['JFK','JFK'], ], tickets)
        return result[0]
```

import collections

```python
class Solution:
    def findItinerary(self, tickets: List[List[str]]) -> List[str]:
        def dfs(f):
            while graph[f]:
                dfs(graph[f].pop(0))
            route.append(f)

        route = []
        graph = collections.defaultdict(list)
        for f, t in sorted(tickets):
            graph[f].append(t)
        dfs('JFK')
        return route[::-1]
```

    ‣ pop(0)을 pop(0)으로 줄여보자.
        • 그래프 구성을 애초에 역순으로 하면 pop(0)으로 가능

import collections

```python
class Solution:
    def findItinerary(self, tickets: List[List[str]]) -> List[str]:
        def dfs(f):
            while graph[f]:
                dfs(graph[f].pop())
            route.append(f)

        route = []
        graph = collections.defaultdict(list)
        for f, t in sorted(tickets, reverse = True):
            graph[f].append(t)
        dfs('JFK')
        return route[::-1]
```

2.스택
```python
import collections

class Solution:
    def findItinerary(self, tickets: List[List[str]]) -> List[str]:
        graph = collections.defaultdict(list)
        for f, t in sorted(tickets):
            graph[f].append(t)
        route, stack = [], ['JFK']
        while stack:
            while graph[stack[-1]]:
                stack.append(graph[stack[-1]].pop(0))
            route.append(stack.pop())
        return route[::-1]
```