Olala_michelle ★ 165 Last Edit: October 25, 2019 9:20 AM 236 VIEWS

5

BFS w/o memo: beats 5.81%

```
class Solution:
    def smallestEquivalentString(self, A, B, S):
        '''step1: Create a mapping from each char to its direct equivalents.'''
        neighbors = collections.defaultdict(set)
        for a, b in zip(A, B):
            neighbors[a].add(b)
            neighbors[b].add(a)

        '''step2: For each char of S, explore the map of all equivalents (marke as seen)
        and memoize the minimum equivalents.'''
        d = {i:i for i in S}

        def bfs(ch):
            res = ch
            seen = set()
            queue = {ch}

            while queue:
                c = queue.pop()
                if c in seen: continue
                seen.add(c)
                res = min(res, c)
                queue |= neighbors[c]

            for v in seen:
                d[v] = res

            return res
        return ''.join(bfs(c) for c  in S)
```

BFS w/ memo: beats 93.35%

```
class Solution:
    def smallestEquivalentString(self, A, B, S):
        neighbors = collections.defaultdict(set)
        for a, b in zip(A, B):
            neighbors[a].add(b)
            neighbors[b].add(a)

        memo = {}

        def bfs(ch):
            if ch in memo: return memo[ch]
            res = ch
            seen = set()
            queue = {ch}

            while queue:
                c = queue.pop()
                if c in seen: continue
                seen.add(c)
                res = min(res, c)
                queue |= neighbors[c]

            for v in seen:
                memo[v] = res

            return res
        return ''.join(bfs(c) for c  in S)
```

DFS: beats 5.8%

```
class Solution(object):
    def smallestEquivalentString(self, A, B, S):
        neighbors = collections.defaultdict(set)
        for a, b in zip(A, B):
            neighbors[a].add(b)
            neighbors[b].add(a)

        visited = set()
        def dfs(ch, minChar, visited):
            visited.add(ch)
            res = minChar

            for nei in neighbors[ch]:
                if nei not in visited:
                    res = min(res, dfs(nei, min(minChar, nei), visited))
            return res


        return ''.join([dfs(c, c, set()) for c in S])
```

Union Find: beats 97.42%

```
class Solution:
    def smallestEquivalentString(self, A: str, B: str, S: str) -> str:
        '''
        step1: model these equalities as edges in a graph
        step2: compute connected components of the graph ==> {node: connID}
        step3: convert 'S'
        '''
        # step 1 & 2
        d = {i:i for i in string.ascii_lowercase}

        def find(x):
            if d[x] != x:
                d[x] = find(d[x])
            return d[x]

        def union(x, y):
            rx, ry = find(x), find(y)
            if d[rx] < d[ry]:
                d[ry] = rx
            else:
                d[rx] = ry

        for a, b in zip(A, B):
            union(a, b)

        # step3
        ans = ''
        for x in S:
            ans += find(x)
        return ans
```

Complexity:

- time: O(max(n, m)) where n is length of A, and m is length of S.
- space: O(1). Since we only need to store the 26 English characters.