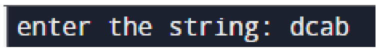
1. You are given a string s, and an array of pairs of indices in the string pairs where pairs[i] = [a, b] indicates 2 indices(0-indexed) of the string.You can swap the characters at any pair of indices in the given pairs any number of times. Return the lexicographically smallest string that s can be changed to after using the swaps.

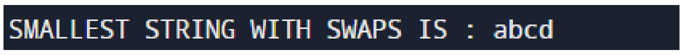
   PROGRAM:

```
class UnionFind:
  def _init_(self, n):
    self.parent = list(range(n))
    self.rank = [0] * n

  def find(self, x):
    if self.parent[x] != x:
      self.parent[x] = self.find(self.parent[x])
    return self.parent[x]

  def union(self, x, y):
    root_x = self.find(x)
    root_y = self.find(y)
    if root_x != root_y:
      if self.rank[root_x] < self.rank[root_y]:
        self.parent[root_x] = root_y
      elif self.rank[root_x] > self.rank[root_y]:
        self.parent[root_y] = root_x
      else:
        self.parent[root_y] = root_x
        self.rank[root_x] += 1

def smallestStringWithSwaps(s, pairs):
  n = len(s)
  uf = UnionFind(n)
  for pair in pairs:
    uf.union(pair[0], pair[1])

  groups = {}
  for i in range(n):
    root = uf.find(i)
    if root not in groups:
      groups[root] = []
    groups[root].append(s[i])

  for root in groups:
    groups[root].sort()

  result = []
  for i in range(n):
    root = uf.find(i)
```

```
        result.append(groups[root].pop(0))

    return ''.join(result)

s = "dcab"
pairs = [[0,3],[1,2],[0,2]]
print(smallestStringWithSwaps(s, pairs))
```

INPUT: 

```
enter the string: dcab
```

OUTPUT: 

```
SMALLEST STRING WITH SWAPS IS : abcd
```

TIME COMPLEXITY: O(n)