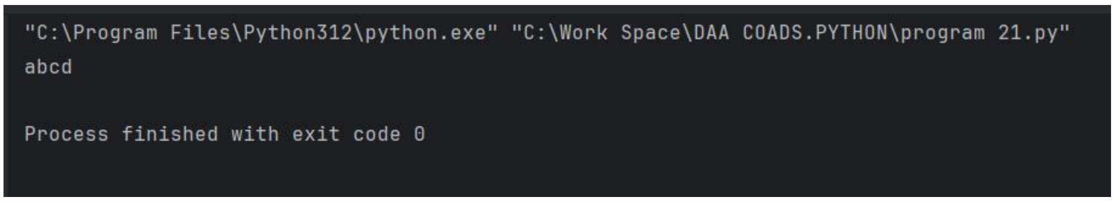
PROGRAM21:You are given a string s, and an array of pairs of indices in the string pairs where pairs[i] = [a, b] indicates 2 indices(0-indexed) of the string.You can swap the characters at any pair of indices in the given pairs any number of times. Return the lexicographically smallest string that s can be changed to after using the swaps.
Progarm:

```
class UnionFind:
   def __init__(self, n):
      self.parent = list(range(n))
      self.rank = [1] * n

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

   def union(self, u, v):
      root_u = self.find(u)
      root_v = self.find(v)
      if root_u != root_v:
         if self.rank[root_u] > self.rank[root_v]:
            self.parent[root_v] = root_u
         elif self.rank[root_u] < self.rank[root_v]:
            self.parent[root_u] = root_v
         else:
            self.parent[root_v] = root_u
            self.rank[root_u] += 1

def smallestStringWithSwaps(s, pairs):
   n = len(s)
   uf = UnionFind(n)

   # Step 2: Union operations for each pair
   for a, b in pairs:
      uf.union(a, b)

   # Step 3: Group indices by their root parents
   from collections import defaultdict
   groups = defaultdict(list)
   for i in range(n):
      root = uf.find(i)
      groups[root].append(i)

   # Step 4: Sort characters within each group and reconstruct the string
   char_list = list(s)
   for group in groups.values():
      sorted_chars = sorted(char_list[i] for i in group)
      for i, char in zip(sorted(group), sorted_chars):
         char_list[i] = char

   return ''.join(char_list)

# Example usage:
s = "dcab"
pairs = [[0, 3], [1, 2], [0, 2]]
print(smallestStringWithSwaps(s, pairs))  # Output: "abcd"
```

OUTPUT:



```
"C:\Program Files\Python312\python.exe" "C:\Work Space\DAA COADS.PYTHON\program 21.py"
abcd

Process finished with exit code 0
```

TIME COMPLEXITY:
O(n log n)