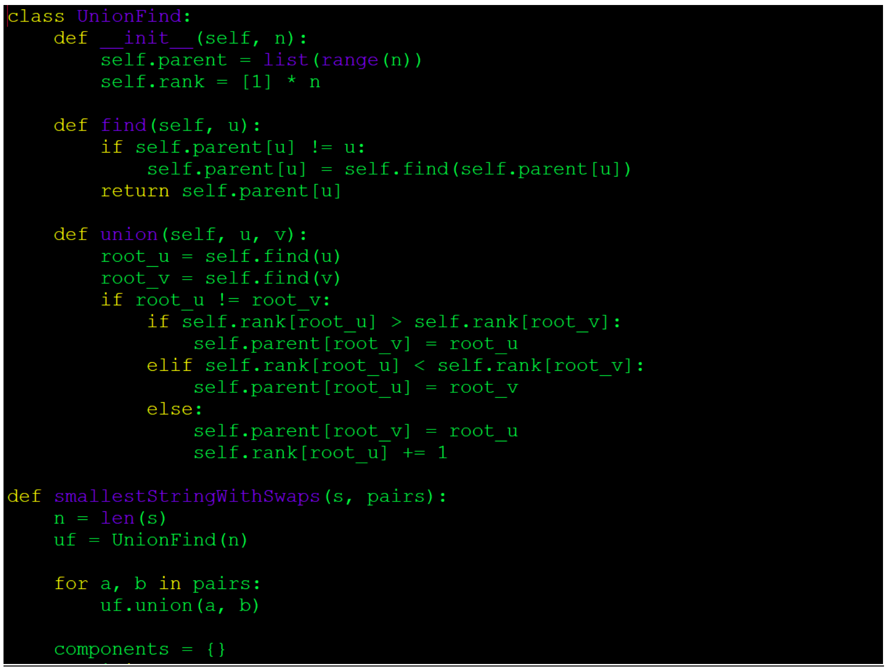
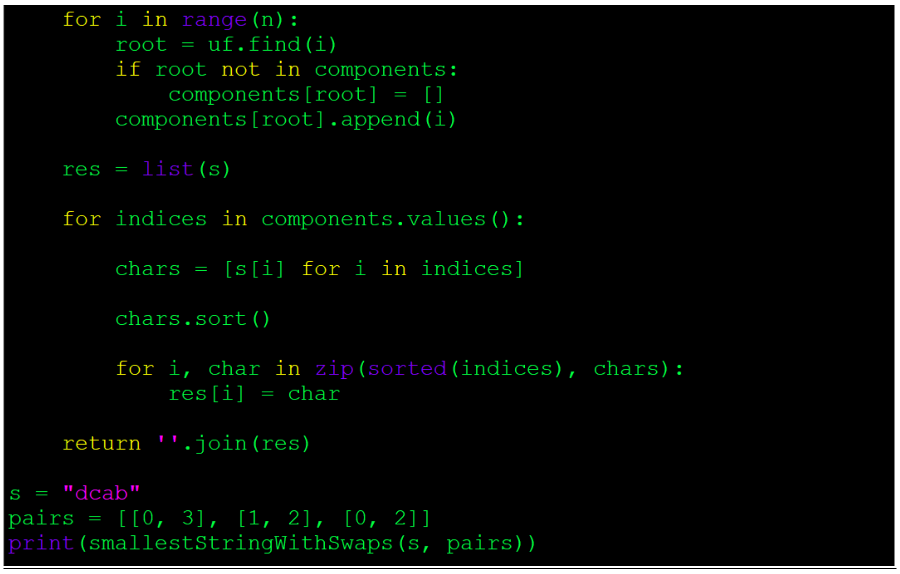
21. You are given a string s, and an array of pairs of indices in the string pairs where pairs[i] = [a, b] indicates 2 indices(0-indexed) of the string.You can swap the characters at any pair of indices in the given pairs any number of times. Return the lexicographically smallest string that s can be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

Code:

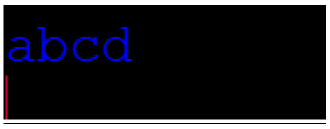


```python
class UnionFind:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [1] * n

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

    for a, b in pairs:
        uf.union(a, b)

    components = {}
```



```python
    for i in range(n):
        root = uf.find(i)
        if root not in components:
            components[root] = []
        components[root].append(i)

    res = list(s)

    for indices in components.values():

        chars = [s[i] for i in indices]

        chars.sort()

        for i, char in zip(sorted(indices), chars):
            res[i] = char

    return ''.join(res)

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

<u>Output:</u>



```
abcd
```

<u>Time Complexity:</u>

- T(n)= O(v)