20. Valid Parentheses

```
class Solution:
    def isValid(self, s: str) -> bool:
        parenthesePairs = {'(': ')', '[': ']', '{': '}'}
        stack = []
        for i in s:
            if i in '([{':
                stack.append(i)
            else:
                if not stack or parenthesePairs[stack.pop()] != i:
                    return False
        return stack == []
```

323. Number of Connected Components in an Undirected Graph

Bfs

```
from collections import defaultdict
class Solution:
    def countComponents(self, n: int, edges: List[List[int]]) -> int:
        visited = set()
        memo = defaultdict(set)
        result = 0
        for s, e in edges:
            memo[s].add(e)
            memo[e].add(s)

        for i in range(n):
            if i not in visited:
                result += 1
            cur = set([i])
            visited.add(i)
            while cur:
                next_ite = set()
                for cur_i in cur:
                    for neighbor in memo[cur_i]:
                        if neighbor not in visited:
                            next_ite.add(neighbor)
                            visited.add(neighbor)
                cur = next_ite
        return result
```

Dfs:

```
from collections import defaultdict
class Solution:
```

```
    def countComponents(self, n: int, edges: List[List[int]]) -> int:
        visited = [False] * n
        memo = defaultdict(set)
        result = 0
        for s, e in edges:
            memo[s].add(e)
            memo[e].add(s)

        def dfs(cur):
            for i in memo[cur]:
                if not visited[i]:
                    visited[i] = True
                    dfs(i)
        for i in range(n):
            if visited[i]:
                continue

            result += 1
            dfs(i)

        return result
```

Union find:

```
from collections import defaultdict
class Solution:
    def countComponents(self, n: int, edges: List[List[int]]) -> int:
        parents = {}
        def findParents(child):
            ori_child = child
            while child in parents and parents[child] != child:
                child = parents[child]
            parents[ori_child] = child
            return child

        for s, e in edges:
            s_parent = findParents(s)
            e_parent = findParents(e)
            if s_parent != e_parent:
                parents[s_parent] = e_parent
                n -= 1
        return n
```

Partial sort

```python
def mySort(s):
    l = list(s)
    i = 0
    result = []
    while i < len(l):
        cur_i = i
        while i + 1 < len(l) and l[i].isdigit() == l[i + 1].isdigit():
            i += 1
        result.extend(sorted(l[cur_i:i + 1]))
        i += 1
    return ''.join(result)


print(mySort('AYUKB17053UI903TBC') == "ABKUY01357IU039BCT")
```