358. Rearrange String k Distance Apart
We could use a priority queue and queue to append character from s. TC is O(n)

```
from collections import Counter
from collections import deque
from heapq import *
class Solution:
    def rearrangeString(self, s: str, k: int) -> str:
        result = []
        queue = deque([])
        pq = []
        for i in map(lambda a: (-a[1], a[0]), Counter(s).items()):
            heappush(pq, i)
        while pq:
            num, c = heappop(pq)
            num = -num
            result.append(c)
            queue.append((num - 1, c))
            if len(queue) < k:
                continue
            num, c = queue.popleft();
            if num > 0:
                heappush(pq, (-num, c))
        return ''.join(result) if len(result) == len(s) else ""
```

767. Reorganize String

```
from collections import Counter
from collections import deque
from heapq import *
class Solution:
    def reorganizeString(self, S: str) -> str:
        result = []
        queue = deque([])
        pq = []
        for i in map(lambda a: (-a[1], a[0]), Counter(S).items()):
            heappush(pq, i)
        while pq:
            num, c = heappop(pq)
            num = -num
            result.append(c)
            queue.append((num - 1, c))
            if len(queue) < 2:
                continue
            num, c = queue.popleft();
```

```
            if num > 0:
                heappush(pq, (-num, c))
        return ''.join(result) if len(result) == len(S) else ""
```

269. Alien Dictionary
We will use indegree and outdegree to order our characters. TC is O(n)

```
from collections import defaultdict
class Solution:
    def alienOrder(self, words: List[str]) -> str:
        indegree = defaultdict(set)
        outdegree = defaultdict(set)
        prev = words[0]
        all_collec = set(list(prev));
        result = []
        for word in words[1:]:
            all_collec = all_collec | set(list(word))
            length = min(len(prev), len(word))
            for i in range(length):
                if prev[i] != word[i]:
                    outdegree[prev[i]].add(word[i])
                    indegree[word[i]].add(prev[i])
                    break
            prev = word
        cur = all_collec - set(indegree.keys())
        while cur:
            next_ite = set()
            for i in cur:
                result.append(i)
                for w in outdegree[i]:
                    indegree[w].remove(i)
                    if len(indegree[w]) == 0:
                        next_ite.add(w)
            cur = next_ite
        if len(all_collec) == len(result):
            return ''.join(result)
        else:
            return ''
```

953. Verifying an Alien Dictionary
We will do comparison one by one. TC is O(n)

```
class Solution:
    def isAlienSorted(self, words: List[str], order: str) -> bool:
```

```
        prev = words[0]
        for word in words[1:]:
            length = min(len(prev), len(word))
            mark = False
            for i in range(length):
                if word[i] != prev[i]:
                    mark = True
                    if order.index(prev[i]) > order.index(word[i]):
                        return False
                    break
            if not mark and len(prev) > len(word):
                return False
            prev = word
        return True

        207. Course Schedule
from collections import defaultdict
class Solution:
    def canFinish(self, numCourses: int, prerequisites: List[List[int]]) -> bool:
        indegree = defaultdict(set)
        outdegree = defaultdict(set)
        for s, f in prerequisites:
            indegree[s].add(f)
            outdegree[f].add(s)
        begin = set(range(numCourses)) - set(indegree.keys())
        total = len(begin)
        while begin:
            next_ite = set()
            for i in begin:
                if i in outdegree:
                    for j in outdegree[i]:
                        indegree[j].remove(i)
                        if len(indegree[j]) == 0:
                            next_ite.add(j)
                            total += 1
            begin = next_ite
        return total == numCourses
```