91. Decode ways
This question is quite DP style. If s[i - 1:i + 1] is between 10 and 26, we will add number num[i - 2], if s[i] > 0, we will add num[i - 1], then set num[i] to the number we get. TC is O(n).

```
class Solution:
    def numDecodings(self, s: str) -> int:
        pre1, pre2, cur = 1, 1, 0
        length = len(s)
        plus_one_mark = False

        if int(s[0]) == 0:
            return 0

        for i in range(1, length):
            cur = 0
            new_num_two = int(s[i-1: i+1])
            new_num_one = int(s[i])
            if 10 <= new_num_two <= 26:
                cur += pre1
            if new_num_one > 0:
                cur += pre2
            if cur == 0:
                return 0
            pre1, pre2 = pre2, cur


        return pre2
```

332. Reconstruct Itinerary
For this question, we need to sort our tickets and map it as (start, to_array). Then we need a visited map to memorize its every ticket use time to prevent over used. Then we will use dfs until we find the final itinerary. The TC is O(nlogn)

```
from collections import defaultdict
class Solution:
    def findItinerary(self, tickets: List[List[str]]) -> List[str]:
        tickets.sort(key=lambda k: (k[0], k[1]))
        length = len(tickets)
        tickets_memo = defaultdict(list)
        visited = defaultdict(int)

        def dfs(visited, tickets_memo, cur, ite, length):
            for i in tickets_memo[cur]:
```

```
            if visited[(cur, i)]:
                visited[(cur, i)] -= 1
                if length == len(ite):
                    return ite + [i]
                temp = dfs(visited, tickets_memo, i, ite + [i], length)
                if temp:
                    return temp
                visited[(cur, i)] += 1

        for src, dest in tickets:
            tickets_memo[src].append(dest)
            visited[(src, dest)] += 1

        return dfs(visited, tickets_memo, 'JFK', ['JFK'], length)
```

124. Binary Tree Maximum Path Sum

We traverse Binary tree from bottom up. We use 0 to dump all negative numbers. For every node, we compare the sum of node.val + left_val + right_val to get the current max one. Then we return the max(node.val + left_val, node.val + right_val) up for the future use. TC is O(n)

```
class Solution:
    def maxPathSum(self, root: TreeNode) -> int:
        self.max_sum = -float('inf')

        def traverse(node):
            if not node:
                return 0
            left = max(0, traverse(node.left))
            right = max(0, traverse(node.right))
            self.max_sum = max(self.max_sum, node.val + left + right)

            return max(left, right) + node.val

        traverse(root)
        return self.max_sum
```

609. Find Duplicate File in System.

This question is very simple. Nothing more to say. TC is O(mn):

```
from collections import defaultdict
class Solution:
    def findDuplicate(self, paths: List[str]) -> List[List[str]]:
        memo = defaultdict(list)

        for path in paths:
```

```
            parsed = path.split(' ')
            for s in parsed[1:]:
                title, content = s.split('(')
                memo[content[:-1]].append(parsed[0] + '/' + title)
        return list(filter(lambda x: len(x) > 1, memo.values()))
```

393. UTF-8 Validation

This question is quite easy, we just follow the instruction from last one to the first one, check whether it follows the format. And return the final result. TC is O(n)

```
class Solution:
    def validUtf8(self, data: List[int]) -> bool:
        length = len(data)
        ind = 0

        while ind < length:
            num = data[ind] & int('11111000', 2)
            if num >> 3 == int('11110', 2):
                ind += 1
                for i in range(3):
                    if ind == length:
                        return False
                    if (data[ind] & int('11111000', 2)) >> 6 != 2:
                        return False
                    ind += 1
            elif num >> 4 == int('1110', 2):
                ind += 1
                for i in range(2):
                    if ind == length:
                        return False
                    if (data[ind] & int('11111000', 2)) >> 6 != 2:
                        return False
                    ind += 1
            elif num >> 5 == int('110', 2):
                ind += 1
                if ind == length:
                    return False
                if (data[ind] & int('11111000', 2)) >> 6 != 2:
                    return False
                ind += 1
            elif num >> 7 == int('0', 2):
                ind += 1
            else:
```

```
        return False
return True
```