最小路径

class Solution:

    def minPathSum(self, grid: List[List[int]]) -> int:

        if not grid or not grid[0]:

            return 0

        row, columns=len(grid),len(grid[0])

        dp=[[0]\*columns for \_ in range(row)]

        dp[0][0]=grid[0][0]

        for i in range(1,row):

            dp[i][0]=dp[i-1][0]+grid[i][0]

        for j in range(columns):

            dp[0][j]=dp[0][j-1]+grid[0][j]

        for i in range(1,row):

            for j in range(1,columns):

                dp[i][j]=min(dp[i-1][j],dp[i][j-1])+grid[i][j]

        return dp[row-1][columns-1]

解码方法

class Solution:

    def numDecodings(self, s: str) -> int:

        dp=[0]\*(len(s)+1)

        dp[0]=1

        if s[0]=='0':

            dp[1]=0

        else:

            dp[1]=1

        for i in range(2,len(s)+1):

            if s[i-1]!='0' and int(s[i-2]+s[i-1])>=1 and int(s[i-2]+s[i-1])<=26:

                if s[i-2]!='0':

                    dp[i]=dp[i-1]+dp[i-2]

                if s[i-2]=='0':

                    dp[i]=dp[i-1]

            else:

                if s[i-1]=='0':

                    if int(s[i-2]+s[i-1])>=1 and int(s[i-2]+s[i-1])<=26:

                        dp[i]=dp[i-2]

                    else:

                        dp[i]=0

                elif int(s[i-2]+s[i-1])>26:

                    dp[i]=dp[i-1]

        return dp[len(s)]

最大正方形

class Solution:

    def maximalSquare(self, matrix: List[List[str]]) -> int:

        if len(matrix)==0 or len(matrix[0])==0:

            return 0

        maxSide=0

        rows,columns=len(matrix),len(matrix[0])

        dp=[[0]\*columns for \_ in range(rows)]

        for i in range(rows):

            for j in range(columns):

                if matrix[i][j]=='1':

                    if i==0 or j==0:

                        dp[i][j]=1

                    else:

                        dp[i][j]=min(dp[i-1][j],dp[i][j-1],dp[i-1][j-1])+1

                    maxSide=max(maxSide,dp[i][j])

        maxSquare=maxSide\*maxSide

        return maxSquare

任务调度器

class Solution:

    def leastInterval(self, tasks: List[str], n: int) -> int:

        c,t = Counter(tasks),0

        big\_n = c.most\_common(1)[0][1]

        for k in c.values():

            if k == big\_n: t += 1

        res = (big\_n-1)\*(n+1)+t

        return res if res >= len(tasks) else len(tasks)

回文子串

class Solution:

    def countSubstrings(self, s: str) -> int:

        if not s:

            return 0

        length = len(s)

        count = 0

        for center in range(2\*length-1):

            left = center // 2

            right = left + center % 2

            while left >= 0 and right < length and s[left] == s[right]:

                count += 1

                left -= 1

                right += 1

        return count