LC 1646

class Solution:

def getMaximumGenerated(self, n: int) -> int:

if n == 0:

return 0

lst = [0] \* (n+1)

lst[1] = 1

res = 1

for i in range(2,n+1):

if i % 2 == 0:

lst[i] = lst[i // 2]

else:

lst[i] = lst[i // 2] + lst[i // 2 + 1]

res = max(res,lst[i])

return res

LC 1653

class Solution:

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

i = len(s)-1

numa = 0

numb = 0

res = 0

while i >= 0:

if s[i] == 'a':

numa += 1

else:

numb += 1

res = min(res,numa-numb)

i -= 1

return res+numb

LC 188

class Solution:

def maxProfit(self, k: int, prices: List[int]) -> int:

if k == 0:

return 0

costs = [float('inf') for \_ in range(k)]

profits = [0 for \_ in range(k)]

for price in prices:

for i in range(k):

if i == 0:

costs[i] = min(costs[i],price)

else:

costs[i] = min(costs[i],price-profits[i-1])

profits[i] = max(profits[i],price-costs[i])

return profits[-1]

LC 329

(inspired by the official solution)

class Solution:

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

m = len(matrix)

n = len(matrix[0])

mat = [[-1] \* (n+2) for \_ in range(m+2)]

directions = {(0,1),(0,-1),(1,0),(-1,0)}

outdegree = [[0] \* (n+2) for \_ in range(m+2)]

for i in range(m):

for j in range(n):

mat[i+1][j+1] = matrix[i][j]

for i in range(1,m+1):

for j in range(1,n+1):

for d in directions:

if mat[i][j] < mat[i+d[0]][j+d[1]]:

outdegree[i][j] += 1

n += 2

m += 2

leaves = []

for i in range(1,m-1):

for j in range(1,n-1):

if outdegree[i][j] == 0:

leaves.append((i,j))

height = 0

while leaves:

height += 1

newLeaves = []

for l in leaves:

for d in directions:

x = l[0] + d[0]

y = l[1] + d[1]

if mat[l[0]][l[1]] > mat[x][y]:

outdegree[x][y] -= 1

if outdegree[x][y] == 0:

newLeaves.append((x,y))

leaves = newLeaves

return height

LC 518

class Solution:

def change(self, amount: int, coins: List[int]) -> int:

arr = [0] \* (amount + 1)

arr[0] = 1

for c in coins:

for i in range(1, amount + 1):

if i >= c:

arr[i] += arr[i-c]

return arr[-1]