122. Best Time to Buy and Sell Stock II

Best Time to Buy and Sell Stock II

```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        best=0

        for i in range(1,len(prices)):
            if prices[i]>prices[i-1]:
                best+=prices[i]-prices[i-1]

        return best
```

944. Delete Columns to Make Sorted

r = 0 for i in range(len(A[0])): array = [] for item in A: array.append(item[i]) if array != sorted(array): r += 1 return r

1005. Maximize Sum Of Array After K Negations

Maximize Sum Of Array After K Negations

```
class Solution:
    def largestSumAfterKNegations(self, A: List[int], K: int) -> int:
        lst = sorted(A)
        count = K
        i = 0
        while count > 0:
            if lst[i] <= lst[i + 1]:
                lst[i] = lst[i] * -1
                count -= 1
            else:
                i += 1
        return sum(lst)
```

1029. Two City Scheduling

Two City Scheduling

```
class Solution:
    def twoCitySchedCost(self, costs: List[List[int]]) -> int:
        # first assume that everyone is going to city A
        # then identify N people to go to city B
        # according to how much they can help save when
        # changing destination
        # to city B
        N = len(costs)//2
        total_cost_A = 0
        d = []
    #
        for i in range(2*N):
            total_cost_A += costs[i][0]
            d.append(costs[i][0]-costs[i][1])
            # sort the difference (from large to small)
            d.sort(reverse=True)
        return(total_cost_A - sum(d[0:N]))
```

1046. Last Stone Weight

Last Stone Weight

```
class Solution:
    def lastStoneWeight(self, stones: List[int]) -> int:
        stones.sort()
        while(len(stones)>1):
            stone1=stones.pop()
            stone2=stones.pop()
            diff=abs(stone1-stone2)
            stones.append(diff)
            stones.sort()
        return stones[0]
```

1217. Play with Chips

Play with Chips

```
class Solution:
    def minCostToMoveChips(self, chips: List[int]) -> int:
        num_odd=0
        num_even=0
        for c in chips:
            if c%2==0:
                num_even+=1

            else:
                num_odd+=1

        return min(num_odd,num_even)
```

1221. Split a String in Balanced Strings

Split a String in Balanced Strings

```
class Solution:
    def balancedStringSplit(self, s: str) -> int:
        count=0
        word=0

        for i in range(len(s)):
            if s[i] == 'L':
                count += 1
            else:
                count -=1

            if count == 0 :
                word += 1
        return word
```

1403. Minimum Subsequence in Non-Increasing Order

Minimum Subsequence in Non-Increasing Order

```
class Solution:
    def minSubsequence(self, nums: List[int]) -> List[int]:
        result=[]
        nums.sort(reverse=True)

        for i in range(len(nums)):
            if(sum(result)>(sum(nums)-sum(result))):
                return result
            else:
                result.append(nums[i])

        return result
```
