HOMEWORK 5 REPORT

PART 1

Explanation of Algorithm: We need to do the job maximum rated job first according to their "W/T" rates for minimize the cost.So i calculated the "W/T" rates for every job.Then i calculate the cost as "Total Time Spent * Current Job Weight" and i deleted the job from my both lists.

Time Complexity:

```
def algo1(arr):
    totalCost = 0
    totalTime = 0
    order = []
    rates = []
    for i in range(0, len(arr)):  #Calculc
        rates.append(arr[i][2]/arr[i][1])
    while rates:    #Loop until all jobs c
        index=rates.index(max(rates))   #F
        totalTime = totalTime + arr[index]
        totalCost = totalCost + totalTime*
        order.append(arr[index][0])  #Appe
        del rates[index]   #Delete the com
        del arr[index]     #Delete the com
    print("Job Order: ", order)
    print("Total Cost: ", totalCost)
```

There are 2 loops in the code.They both will iterate "n" times because we decrease the size of the list by 1 in "while loop".So the time complexity will be O(N).

PART 2 – a

for i= 1 to n

  if $N_i < S_i$ then

    Output "NY in Month i"

  else

    Output "SF in Month i"

end

Let's suppose n=4, M=10 and the operating costs are given by the following table

#	Month 1	Month 2	Month 3	Month 4
NY	**1**	**10**	**9**	**10**
SF	**10**	**9**	**10**	**9**

According the code given above the sequence will be:

NY->SF->NY->SF

According the this sequence the cost will be = 1 + 9 + 9 + 9 + 10 + 10 + 10 = 58

But the optimal path which is **NY->NY->NY->NY** cost is = 1 + 10 + 9 + 10 = 30

As we can see the given code is not working well because we need the consider the M value as well.

PART 2 - b

Explanation of Algorithm: Every iterate of the loop i calculate the costs for the current step.The lower cost means optimal plan for current month.Therefore i used as dynamically for calculate the next step.So the minimum of the last element of the array1 and array2 is my final optimal plan cost.And i find the optimal plan by comparing 2 arrays.If the array1 index is lower than array2, that means the optimal city for this month is “NY”, else its “SF”

Time Complexity:

```
def algo2(n, M, sequenceNY, sequenceSF):
    arr1 = [0]
    arr2 = [0]
    plan = []
    for i in range(0, n):
        arr1.append(sequenceNY[i] + min(arr1[i], M+arr2[i]))
        arr2.append(sequenceSF[i] + min(arr2[i], M+arr1[i]))
    for cost in range(n, 0, -1):   #Loop for print the optimal
        if arr1[cost]<arr2[cost]:
            plan.insert(0, "NY")
        elif arr2[cost]<arr1[cost]:
            plan.insert(0, "SF")
        elif cost<4:
            plan.insert(0, plan[0])
        else:
            plan = ["NY"] * n
            break
    print("Optimal Plan: ", plan)
    return min(arr1[n], arr2[n])  #Return the optimal paths cos
```

There are 2 loops in the code.They both time complexity is O(N).So the time complexity will be **O(N).**