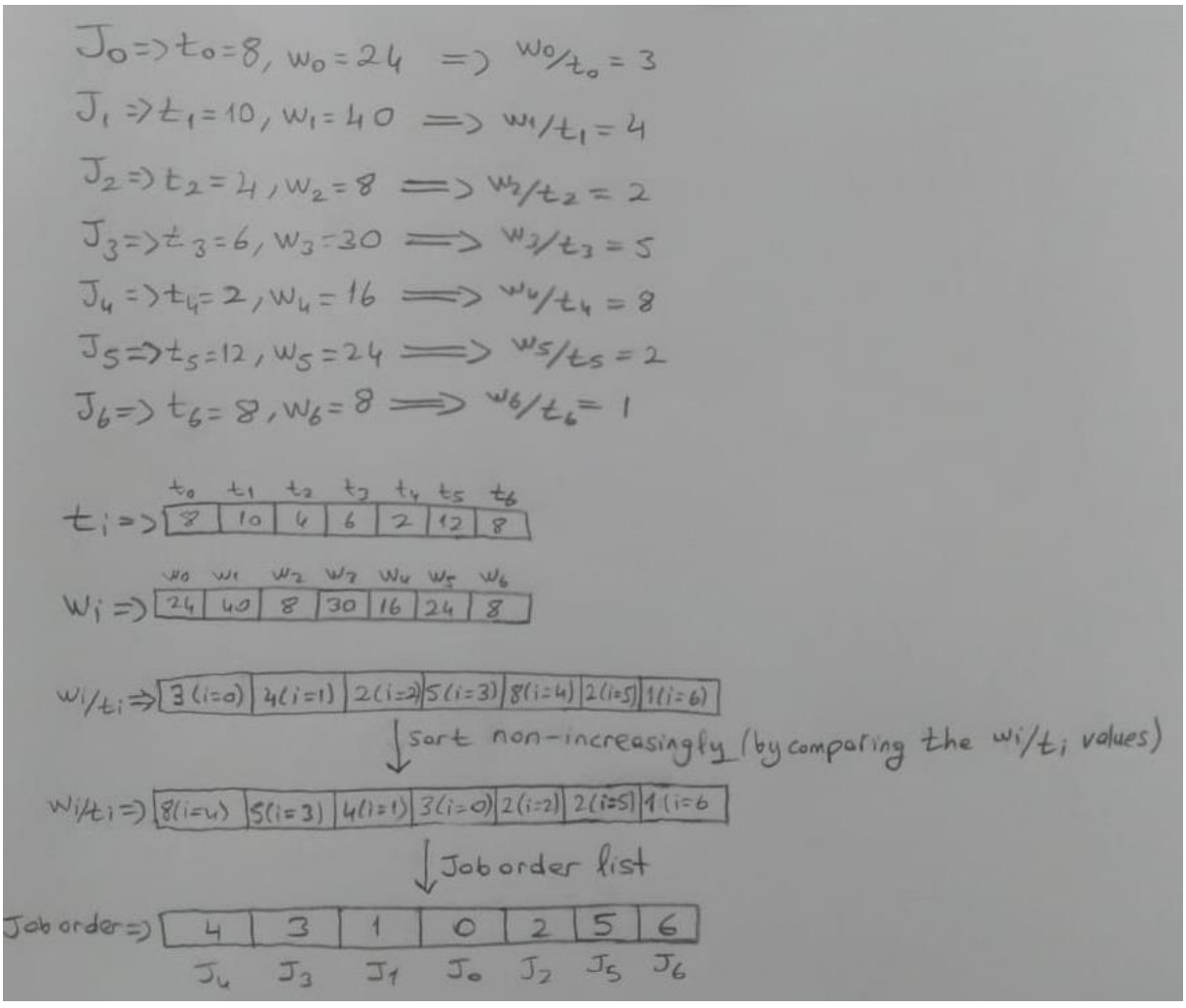
Q1

Explaining The Algorithm

I got two list(jobProcessingTimeList and jobWeightList) and return one list(jobOrderList).Indices of the jobProcessingTimeList and jobWeightList lists are equal the job numbers.For minimizing the weighted sum of the completion times, I must choose a job firstly if its weight is more and time is less. So, I must choose firstly the job which has biggest wi/ti value.

So I sort the list by comparing the wi/ti values of the jobs. And I return the ordered job list.

An example:



Time Complexity Analyze

I use 2 for loops and one insertion sort. for loops are takes O(n) time and insertion sort takes O(n^2) time.So time complexity is O(n+n+n^2) => O(n^2).

Q2

a)

Let n=4

M=10

N =[1,1,3,1]

S =[12,15,1,14]

This algorithm returns [NY,NY,SF,NY] with total cost=1+1+1+1+10+10=24 (it doesn't look the M value)

But optimal solution is [NY,NY,NY,NY] with total cost=1+1+3+1=6

b)

Explaining The Algorithm

Our optimal plan finishes either NY or SF. If it finishes with NY, there is an exact cost NY for size n and we add to this cost NY[n-1] or SF[n-1]+M(choose the minimum one). If it finishes with SF, there is an exact cost SF for size n and we add to this cost SF[n-1] or NY[n-1]+M(choose the minimum one).

So in the for loop , dynamic programming table is filled and we use its values for calculating the early costs.

Lastly , after the for loop there are two list and at the end of the lists(exact mounthCounts size) one is store the cost of the optimal plan ended with NY the other one is store the cost of the optimal plan ended with SF we choose the minimum one and return it.

Time Complexity Analyze

There is one for loop with size n. The other processes takes constant time. So time complexity is O(n).