Name: HAVI BOHRA
Roll No.: 210429

Solution (1):

Idea: Sort the jobs in non-increasing order of $\frac{w_i}{t_i}$

Proof: Suppose there exists any other optimal scheduling order S_{other}, other than proposed above, then there exists consecutively scheduled jobs i and j s.t.

$$\frac{w_i}{t_i} < \frac{w_j}{t_j}$$

Now, note that if we swap order of these two jobs , then completion time of jobs other than these two won't be affected.

So the change in $\sum_{k=1}^{n} w_k C_k$ by swapping order of these two jobs is

$$\begin{aligned} Change &= (w_j(z + t_j) + w_i(z + t_j + t_i)) - (w_i(z + t_i) + w_j(z + t_i + t_j)) \\ &= w_i t_j - w_j t_i \\ &= t_i t_j \left(\frac{w_i}{t_i} - \frac{w_j}{t_j} \right) < 0 \end{aligned}$$

This implies that there exists a scheduling order different from S_{other} which has $\sum_{k=1}^{n} w_k C_k$ lesser than the S_{other}, a contradiction!!

Hence, the scheduling order with non-increasing order of $\frac{w_i}{t_i}$ is the required scheduling.

Time Complexity:

1. Sorting an array $\left(\frac{w_i}{t_i} \right)_{i=1,2,\ldots n}$ would take **O(nlogn)** time complexity.

Solution (2):
Idea: Run DFS, and store the minimum of *price(u)* at each vertex, as DFS traverses every reachable node and since G is a DAG, there won't be any cycle resulting into infinite loop.
Pseudo-Code:

```
G=(V,E)
price <- prices of node dor each u in V (+ve entries)
cost  <- prices of node dor each u in V (initially set -1)
DFS(u){
    cost[u] = price[u];
    for v in E[u] //assuming edges are present in adjacency list
    {
        if(cost[v]==-1) DFS(v)
        cost[u] = min(cost[u], cost[v]);
    }
}

for u in V{
    if(cost[u]==-1) DFS(u)
}
```

Time Complexity:
1. Time Complexity of DFS = **O(V+E)**