(a)

(a) cannot guarantee the optimal solution. Here is a counterexample: $t_1 = 1, w_1 = 1$; $t_2 = 3, w_2 = 4$. According to this algorithm, we should place the first job first. However, when doing so, the result will be $1 \times 1 + (1 + 3) \times 4 = 17$; When place the second job first, the result will be $3 \times 4 + (3 + 1) \times 1 = 16$, which is clearly better. Hence, this algorithm is not the case.

(b)

(b) cannot guarantee the optimal solution. Here is a counterexample: $t_1 = 3, w_1 = 3$; $t_2 = 1, w_2 = 2$. According to this algorithm, we should place the first job first. However, when doing so, the result will be $3 \times 3 + (3 + 1) \times 2 = 17$; When place the second job first, the result will be $1 \times 2 + (1 + 3) \times 3 = 14$, which is clearly better. Hence, this algorithm is not the case.

(c)

(c) is the case. **Proof:** Use the induction to prove its correctness.

Base case: number of jobs $n = 1$: There is only one solution, which has to be the optimal one.

Induction: When we add a job $n + 1$ to the n jobs that are already arranged, according to the algorithm, $w_{n+1}/t_{n+1} \leq w_i/t_i$ for i in $1, 2, \cdots n$, and $(t_1 + \cdots + t_n + t_{n+1}) \times w_{n+1}$ will be added to the result. Consider an alternative optimal solution j where exist k in $1, 2, \cdots n$ that $w_j/t_j > w_k/t_k$, and $(t_1 + \cdots + t_n + t_j) \times w_j$ will be added to the result this time. If we replace $n + 1$ by j, denote $t_1 + t_2 + \cdots + t_n$ as a, and the result will be changed by:

$$\begin{aligned} & (a + t_j)w_j - (a + t_{n+1})w_{n+1} \\ = \; & t_j t_{n+1} \left(\tfrac{a(w_j - w_{n+1})}{t_j t_{n+1}} + \tfrac{w_j}{t_{n+1}} - \tfrac{w_{n+1}}{t_j} \right) \\ = \; & t_j t_{n+1}(a + 1) \left(\tfrac{w_j}{t_{n+1}} - \tfrac{w_{n+1}}{t_j} \right) \\ = \; & (a + 1) \left(\tfrac{w_j}{t_j} - \tfrac{w_{n+1}}{t_{n+1}} \right) \end{aligned} \tag{1}$$

Known that $w_{n+1}/t_{n+1} < w_j/t_j$, Equation (1) must be greater than 0, which means replacing $n + 1$ by j makes the result worse (we need it to be as small as possible). So job $n + 1$ should be the best one to be added. Hence, this algorithm us the case.