An optimal algorithm is to schedule the jobs in decreasing order of w_i/t_i. We prove the optimality of this algorithm by an exchange argument.

Thus, consider any other schedule. As is standard in exchange arguments, we observe that this schedule must contain an *inversion* — a pair of jobs i, j for which i comes before j in the alternate solution, and j comes before i in the greedy solution. Further, in fact, there must be an adjacent such pair i, j. Note that for this pair, we have $w_j/t_j \geq w_i/t_i$, by the definition of the greedy schedule. If we can show that swapping this pair i, j does not increase the weighted sum of completion times, then we can iteratively do this until there are no more inversions, arriving at the greedy schedule without having increased the function we're trying to minimize. It will then follow that the greedy algorithm is optimal.

So consider the effect of swapping i and j. The completion times of all other jobs remain the same. Suppose the completion time of the job before i and j is C. Then before the swap, the contribution of i and j to the total sum was $w_i(C + t_i) + w_j(C + t_i + t_j)$, while after the sum it is $w_j(C + t_j) + w_i(C + t_i + t_j)$. The difference between the value after the swap, compared to the value before the swap is (canceling terms in common between the two expressions) $w_i t_j - w_j t_i$. Since $w_j/t_j \geq w_i/t_i$, this difference is bounded above by 0, and so the total weighted sum of completion times does not increase due to the swap, as desired.

[1] ex948.540.252