# Scheduling : Proof of Correctness

**A Proof of Correctness for the Weighted Job Scheduling Algorithm**

**Theorem:** Scheduling jobs in decreasing order of their weight-to-processing-time ratio minimizes the weighted sum of completion times.

**Proof:**

We will use an exchange argument to prove the theorem. This involves showing that if any two jobs in the optimal schedule are swapped, the resulting schedule will have a higher weighted sum of completion times.

Let's assume that the optimal schedule is not in decreasing order of the weight-to-processing-time ratio (i.e. there are two jobs that exist in the greedy schedule that are swapped in this new optimal schedule where the optimal schedule is better than the greedy one). This means there are two adjacent jobs, say job i and job j, such that the ratio of job i is less than the ratio of job j.

Let's denote the processing time of job i as pi, its weight as wi, and its completion time in the optimal schedule as Ci. Similarly, for job j, we have pj, wj, and Cj.

Now, consider swapping jobs i and j in the optimal schedule (i.e. this swap will bring the supposed optimal schedule closer to the greedy schedule we started with). The completion time of job i will increase by pj, and the completion time of job j will decrease by pi.

The change in the weighted sum of completion times due to the swap is:

Δ = (wi \* (Ci + pj)) + (wj \* (Cj - pi)) - (wi \* Ci) - (wj \* Cj)

Simplifying this expression, we get:

Δ = wi \* pj - wj \* pi

Since the ratio of job i is less than the ratio of job j, we have:

wi / pi < wj / pj

Multiplying both sides by pi \* pj, we get:

wi \* pj < wj \* pi

Therefore, Δ is negative, which means the weighted sum of completion times decreases after the swap. This contradicts our assumption that the original schedule was optimal (i.e. This new schedule that is more like the greedy schedule is better than the supposed optimal schedule we started with).

Hence, the optimal schedule must be in decreasing order of the weight-to-processing-time ratio.

**Note:** This proof is a standard approach for proving the correctness of greedy algorithms in scheduling problems. You can find similar proofs in many algorithms textbooks and online resources.

**Recommended Online Resources:**

While I couldn't find a specific PDF that directly addresses this proof, you might find relevant discussions or proofs in these textbooks or online resources:

\* Classic Algorithms Textbooks:

\* "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein

\* "Algorithms" by Sedgewick and Wayne

\* Online Courses and Lecture Notes:

\* Coursera, edX, or MIT OpenCourseWare courses on algorithms or data structures

\* **Research Papers and Articles:**

\* Search academic databases like Google Scholar or ACM Digital Library using keywords like "weighted job scheduling," "greedy algorithm," or "proof of correctness."

These resources should provide you with a more in-depth understanding of the proof and its underlying concepts.

**Understanding the Iterative Improvement Argument**

The iterative improvement argument is a common technique used to prove the optimality of greedy algorithms. It involves starting with an assumed better solution and then showing that we can iteratively improve this solution until it matches the greedy solution.

**Here's how it works:**

\* Assume a better solution: Suppose there's an optimal solution (Opt) that's different from the solution produced by the greedy algorithm (Greedy).

\* Identify a difference: Find a point where the two solutions differ.

\* Improve the assumed solution: Show that by modifying the assumed solution at this point, we can create a new solution that's even better than Opt.

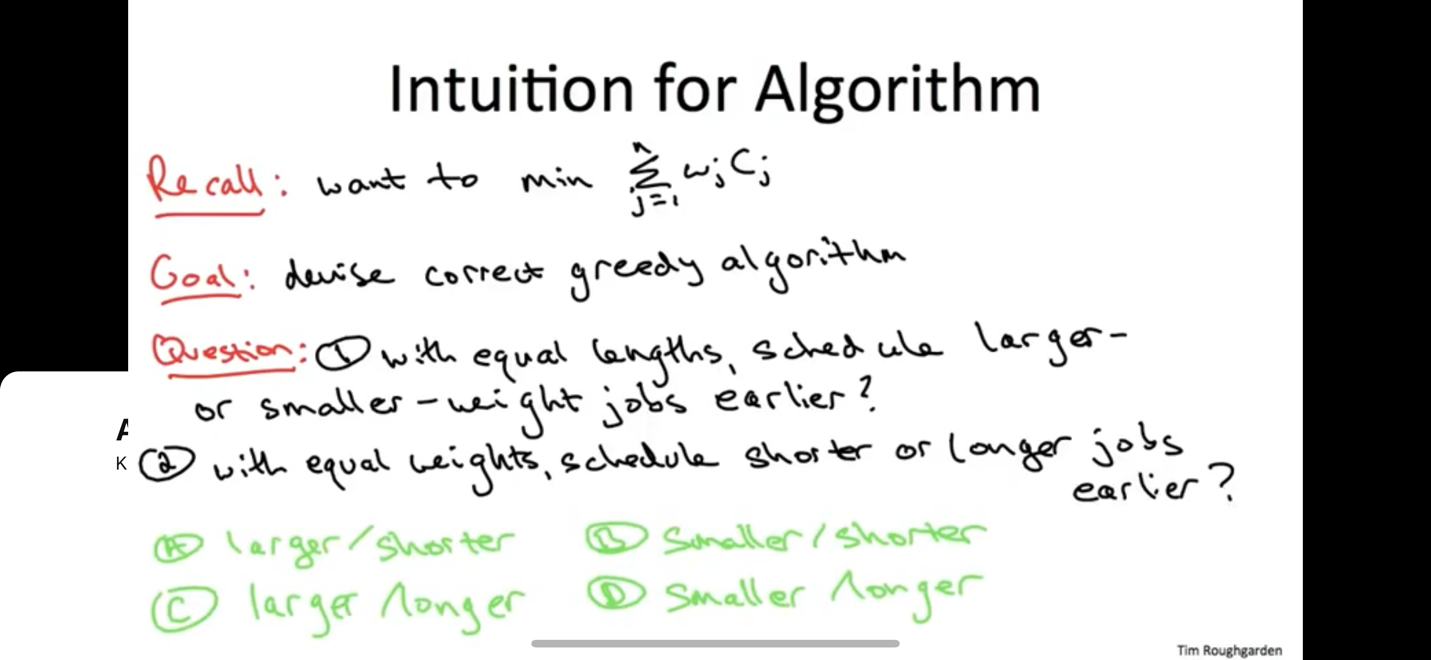
## \*\*Why does this prove that the greedy algorithm is correct?\*\*

\* Contradiction: If we can always improve an assumed better solution, it means that the assumed solution cannot be truly optimal.

\* Greedy solution is optimal: Since the greedy algorithm is the only solution that cannot be further improved, it must be the optimal solution.

In essence, the iterative improvement argument is a proof by contradiction. It assumes the opposite of what we want to prove (i.e., that there's a better solution than the greedy solution) and then shows that this assumption leads to a contradiction. This implies that the greedy solution must be optimal.

Note: This argument is often used in conjunction with other techniques, such as the exchange argument or induction, to provide a more comprehensive proof.

1. With equal lengths and varying weights, larger weight jobs should go first to minimize the product between the next weight and the running total.
2. With varying lengths and equal weights, the shorter jobs should be prioritized to minimize the product between the next weight and the running total.