Lab Worksheet 5: Greedy Algorithms (Solution)

Interval Scheduling Problem

Imagine you are in charge of a single shared resource, like a lecture hall. You have received n requests to book this hall for various events. Each request i is for a specific time interval, starting at s_i and finishing at f_i. Because there is only one hall, you can only approve requests for intervals that do not overlap. We say two intervals, i and j, are **compatible** if the time period for i is completely over before the period for j begins, or vice-versa. Formally, they are compatible if $f_i \leq s_j$ or $f_j \leq s_i$.

Our Goal: To maximize the use of the hall, you want to select the largest possible subset of these requests that are all mutually compatible. Here is an example:

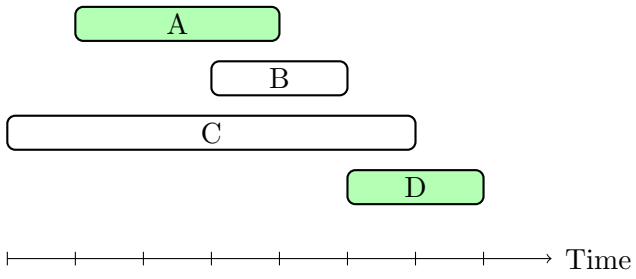


Figure 1: A set of 4 interval requests. The optimal solution is the set {A, D}, with size 2.

Designing a Greedy Algorithm

We will solve this problem using a **greedy algorithm**. The idea is to build the solution piece by piece. At each step, we will make a choice that seems best at that moment, without worrying about future consequences.

Before we explore specific strategies, let's think about the structure of our greedy algorithm more formally. Let's identify the following pieces:

- **State:** What defines the current state of the problem we are trying to solve?
- **Action:** What is the single action our algorithm takes at each step?
- **New State:** After we take an action, how does the state of the problem change? In other words, how do we define the remaining subproblem?

Well... here are the answers: in this context, the state of the problem is the set of available intervals, and our action is to select one. This leads to the following general process:

1. Start with the set of all n requested intervals.
2. Select one interval based on some "greedy" strategy. Add it to our solution set.
3. Remove the selected interval and all intervals that conflict with it from the set of requests.
4. Repeat until no more intervals are left to consider.

The crucial part is the "greedy" strategy for selecting an interval. A good strategy leads to an optimal solution, while a bad one may not. The state of our problem at any point is simply the set of intervals that are still available to be chosen.

Exploring some (incorrect) greedy strategies

Let's explore a few greedy strategies that seem reasonable at first glance. For each one, your task is to find a **counterexample**. That is provide a small set of intervals where the strategy fails to produce the largest possible set of compatible intervals. Show which intervals the greedy algorithm selects and which intervals form the true optimal solution.

Earliest Start Time: The greedy strategy is that at each step, pick the available interval with the earliest start time.

Your counter example:

The greedy algorithm selects A first since it begins at time 0. However, this choice excludes both B and C, resulting in the solution $\{A\}$ of size 1. In contrast, the optimal solution is $\{B, C\}$, which has size 2.

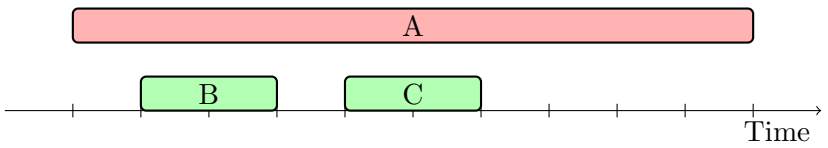


Figure 2: Counterexample for the *Earliest Start Time* strategy.

Shortest Duration: The greedy strategy is that at each step, pick the available interval with the smallest duration $(f_i - s_i)$.

Your counter example:

The greedy algorithm selects C, since it is the shortest interval. However, this choice excludes both A and B, resulting in the solution $\{C\}$ of size 1.

In contrast, the optimal solution is $\{A, B\}$, which has size 2.

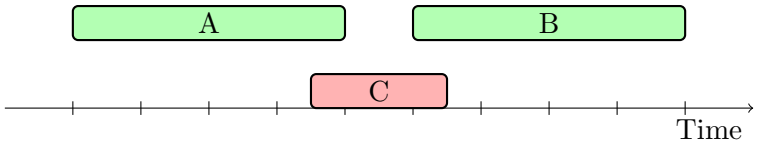


Figure 3: Counterexample for the *Shortest Duration* strategy.

A Correct Greedy Strategy: Earliest Finish Time

Here is a strategy that does work. At each step, pick the available interval with the earliest finish time (smallest f_i).

Let's trace the algorithm. Consider the set of intervals shown below. First, list the intervals (by their letter) in sorted order of their finish times. Then, trace the *Earliest Finish Time* algorithm: state which interval is chosen at each step and which intervals are removed. What is the final set of intervals in your solution?

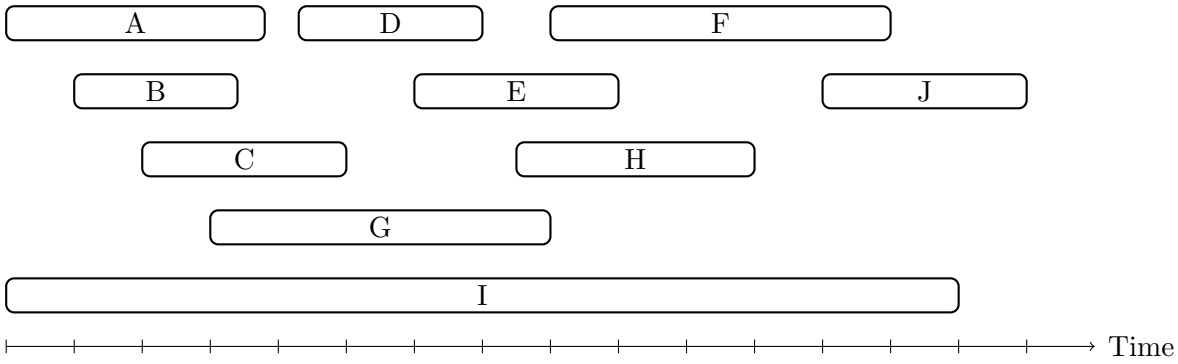


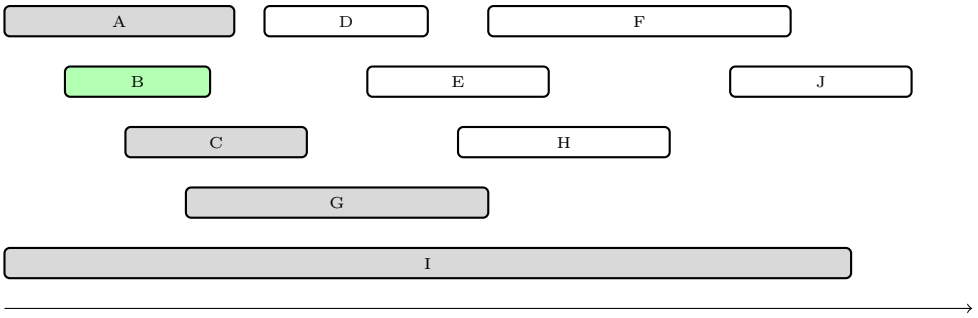
Figure 4: An example to trace the *Earliest Finish Time* algorithm.

If we sort the intervals based on their finish time, we get:

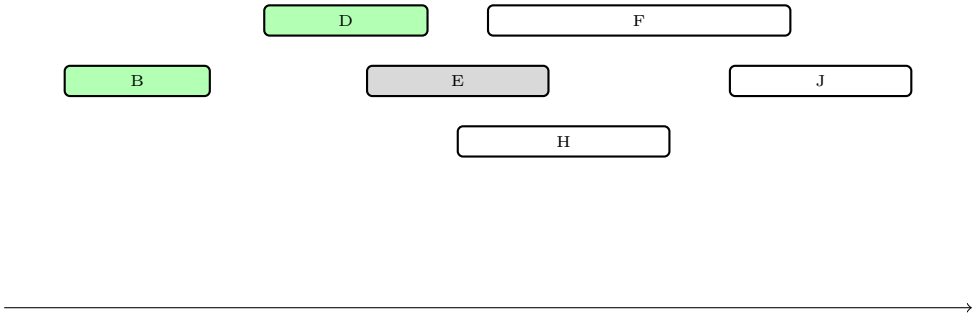
B, A, C, D, G, E, H, F, I, J .

The following is a visual trace for the algorithm. The greedy algorithm selects intervals in four steps. Green intervals are chosen, gray intervals are eliminated as conflicts, and white intervals remain as candidates for the next step.

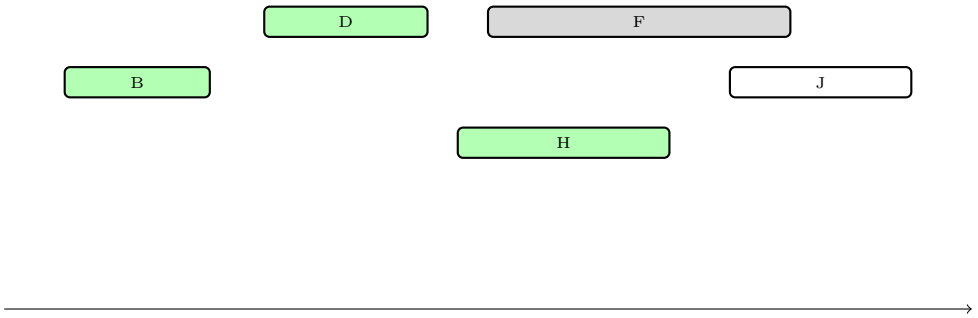
Step 1: Choose B. Eliminate conflicting intervals A, C, G, I.



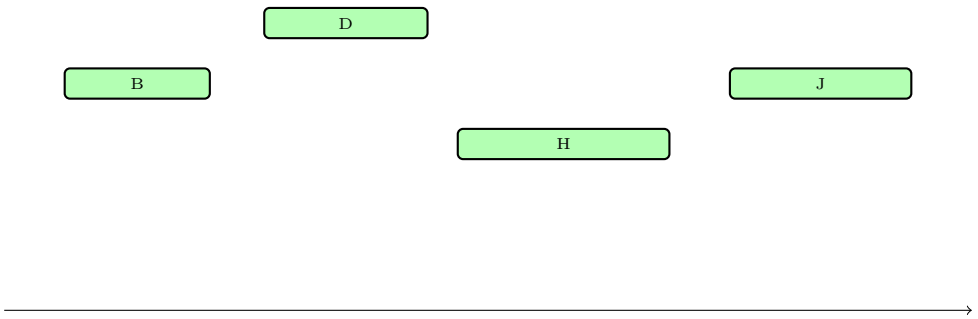
Step 2: Choose D. Eliminate conflicting interval E.



Step 3: Choose H. Eliminate conflicting interval F.



Step 4: Choose J. No remaining candidates.



Final Greedy Solution: {B, D, H, J}. Size = 4.

Proof of Optimality

Now, let's focus on proving that this strategy is always optimal. Here is the statement we want to prove:

Theorem 1. *The greedy algorithm that, at each step, selects an available interval with the earliest finish time produces an optimal solution for the Interval Scheduling problem.*

To prove this, we will use an **exchange argument**. The idea is to show that the greedy algorithm's choices are always *safe* or *correct*. We will demonstrate that for any optimal solution, we can transform it step-by-step into the greedy solution without decreasing its size. Formally, we will build an **inductive argument** on the size of the problem. It is natural to assume that the number of available intervals is a good measure of size.

Write the predicate (or statement) you would like to prove via induction:

P(n): Given n intervals (scheduling requests), the greedy algorithm that, at each step, selects an available interval with the earliest finish time produces an optimal solution for the Interval Scheduling problem.

Base case: Show the statement for $n = 1$. Clearly, our algorithm selects the single available interval. This is optimal, since it represents the maximum number of intervals we could possibly choose.

Induction hypothesis: For the rest of this proof, we assume that $P(n')$ holds for any $n' < n$. [1]

Induction step: We will show that if $P(n')$ holds for all $n' < n$, then $P(n)$ also holds. In the following, we have an sketch of the argument. Complete the missing parts of the proof.

We start off by showing that we can bring the greedy solution closer to the optimal one by proving that at least one of the intervals selected by the greedy algorithm must appear in some optimal solution. This is the *exchange* part of the argument.

Lemma 1. Let g_1 denote the interval with the earliest finish time among all available intervals. Let $O = \{o_1, o_2, \ldots, o_m\}$ be an optimal solution, sorted by finish times. There exists an optimal solution O' that includes g_1.

Hint: Consider a good candidate to remove from O and replace with g_1.

Proof. If g_1 is in O, we are done, as O is an optimal solution containing g_1. Suppose $g_1 \notin O$. Then $g_1 \neq o_1$. By definition of g_1, it has the earliest finish time of any interval, so we have $f_{g_1} \leq f_{o_1}$. Consider a new set of intervals $O' = \{g_1, o_2, \ldots, o_m\}$. We will show that this is an

[1] This is called **strong induction**. Instead of assuming only that $P(n-1)$ holds, we assume that $P(1), P(2), \ldots, P(n-1)$ all hold. This approach sometimes makes the proof easier.

optimal solution as well. This set has the same size, m, as O. We just need to show it is a valid, compatible set. Since $f_{g_1} \leq f_{o_1}$ and the intervals in O were compatible, we know that $s_{o_2} \geq f_{o_1}$, meaning o_2 starts after o_1. This implies $s_{o_2} \geq f_{g_1}$. Therefore, g_1 is compatible with o_2 and, by the same reasoning, with all subsequent intervals in O'. Thus, O' is a set of m mutually compatible intervals. It is a valid, optimal solution that contains the first greedy choice, g_1. □

Lemma 2. Suppose we are given n intervals. Assume that the greedy algorithm produces an optimal solution for any set of intervals of size smaller than n. Let $G = \{g_1, \ldots, g_k\}$ be the solution produced by the greedy algorithm, and let $O' = \{g_1, o_2, \ldots, o_m\}$ be an optimal solution that contains the first greedy choice g_1. Then, G is an optimal solution.

Proof. To prove optimality, we aim to show that $|G| = |O'|$. Consider the subproblem S_1 consisting of all intervals that are *compatible* with g_1, i.e.,

$$S_1 \coloneqq \{\, I \mid s_I \geq f_{g_1} \,\}.$$

Note that $G' = \{g_2, \ldots, g_k\}$ is exactly the set the greedy algorithm selects when run on S_1 (i.e., the intervals chosen after committing to g_1). This modularity yields the identity

$$|G| = 1 + (\text{optimal solution size for } S_1).$$

Define O'' to be $\{o_2, \ldots, o_m\}$. We claim that O'' is a feasible and optimal solution for S_1, thereby linking $|O'|$ and $|G|$.

Feasibility. Since O' is feasible and contains g_1, every o_i with $i \geq 2$ must be compatible with g_1, hence belongs to S_1. Moreover, the o_i remain mutually compatible because O' is feasible. Therefore O'' is a feasible solution for S_1.

Optimality. Suppose, for contradiction, there exists a feasible solution X for S_1 with $|X| > |O''|$. Then

$$\{g_1\} \cup X$$

would be a feasible solution for the original instance of size

$$1 + |X| > 1 + |O''| = |O'|,$$

contradicting the optimality of O'. Hence O'' is optimal for S_1, and so

$$|O'| - 1 = |O''| = \text{optimal solution size for } S_1.$$

Conclusion. Putting this together,

$$|G| = |O'|\,.$$

Thus, the greedy algorithm produces a solution of optimal size. □

Greedy Algorithms: The Interval Stabbing Problem

Imagine you are a biologist studying migratory birds. You know the time intervals during which different species of birds visit a particular watering hole. You want to place automated cameras to record these visits. Each camera can only be placed at a single point in time, but it will record any bird species whose visiting interval includes that point in time. Your cameras are expensive, so you want to use as few as possible to ensure that every species is observed.

This is an instance of the **Interval Stabbing Problem**. You have a set of n intervals on a line, and you want to find the smallest set of points such that every interval is "stabbed" – that is, every interval contains at least one of the points.

Our Goal: Given n intervals, find a minimum-sized set of points that stabs all of them. Here is an example:

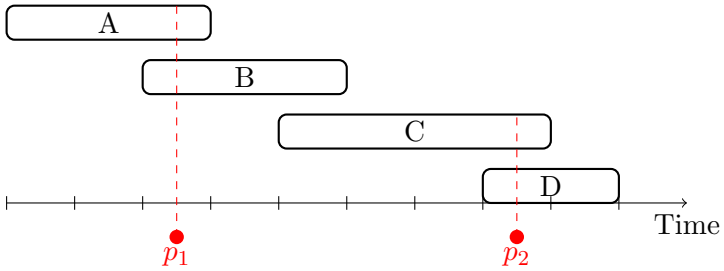


Figure 5: A set of 4 intervals. The optimal solution is to place two points, for example at times 2.5 and 7.5. Point p_1 stabs intervals A and B. Point p_2 stabs intervals C and D.

Designing a Greedy Algorithm

We will solve this with a greedy algorithm. The idea is to repeatedly place points until all intervals are covered. The "greedy" part is deciding *where* to place the next point.

Here is the general process:

1. Start with the set of all n intervals, none of which are stabbed yet.
2. While there are still unstabbed intervals:
   - 2.1 Choose a location to place a point based on some "greedy" strategy. Add this point to our solution set.
   - 2.2 Remove all intervals stabbed by this new point from the set of unstabbed intervals.
3. Return the set of points.

The crucial part is the strategy for placing a point. A good strategy will lead to an optimal (minimum size) set of points.

Exploring an incorrect greedy strategies

Let's explore a greedy strategies that seems plausible. Your task is to find a **counterexample**: a small set of intervals where the strategy fails to produce the smallest possible set of stabbing points.

Most Stabbing Point: The greedy strategy is to find a point on the line that stabs the maximum number of currently unstabbed intervals. Place a point there.

Your counterexample: The greedy strategy picks a point placed anywhere that stabs the maximum number of intervals: B, C, D, and E (a total of four). Then, this choice leaves intervals A and F unstabbed. Since A and F do not overlap, we need two more points to stab them. The greedy solution has a total size of three.

The optimal solution is to place two points, stabbing A, B, C, and stabbing D, E, F. The optimal size is 2.

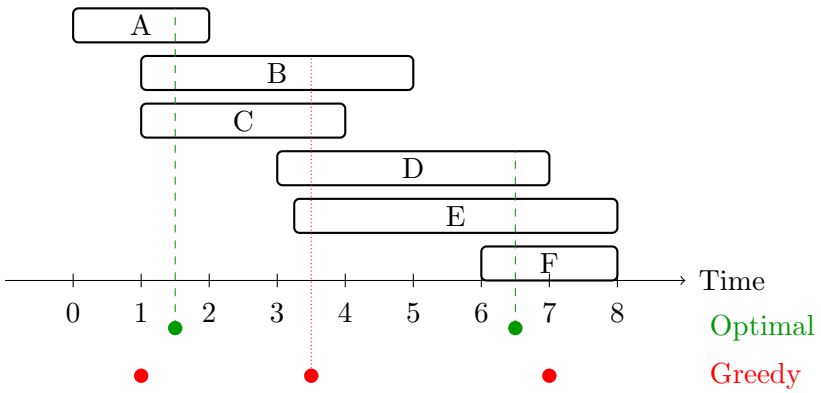


Figure 6: Counterexample for the *Most Stabbing Point* strategy. The greedy choice (red) leads to a 3-point solution. The optimal solution (green) uses only two points.

A Correct Greedy Strategy: Earliest Finish Time

Here is a strategy that does work. Find the unstabbed interval with the **earliest finish time**. Place a point at its finish time.

Let's trace this algorithm. First, sort the intervals below by their finish times. Then, trace the algorithm: state which point is chosen at each step and which intervals are stabbed. What is the final set of points?

Sorted by finish time: A (3), B (5), C (7), D (8), E (9), F (11), G (13).

In the diagrams below, the chosen interval with the earliest finish time is blue. Other intervals stabbed in the current step are dark gray, intervals stabbed in previous steps are light gray, and unstabbed intervals are white.

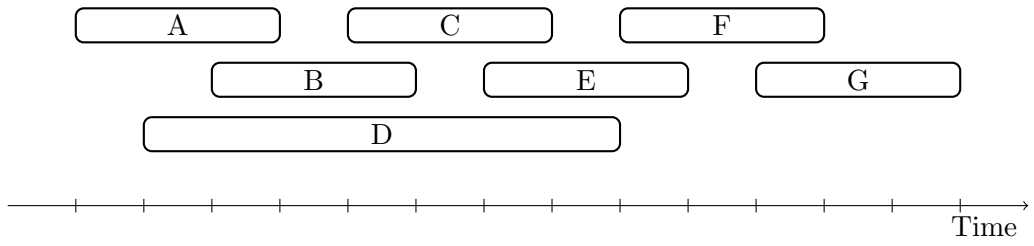
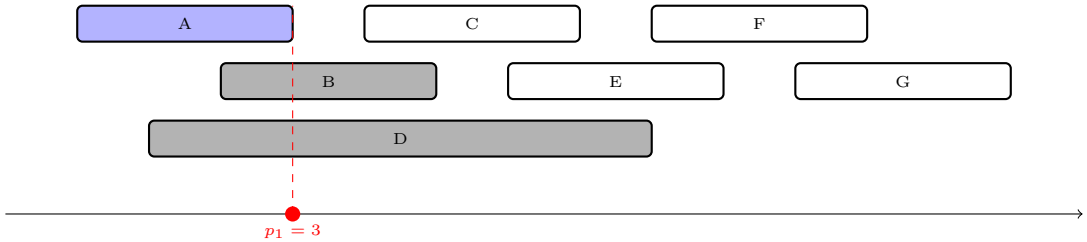


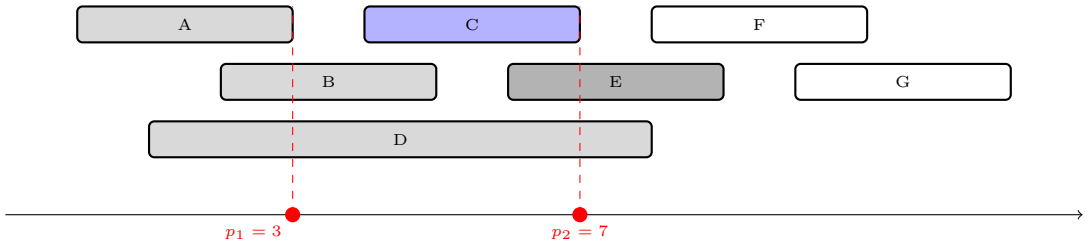
Figure 7: An example to trace the *Earliest Finish Time* algorithm.

Step 1: The interval with the earliest finish time is A (ends at 3). Place a point $p_1 = 3$. This stabs intervals A, B, and D.



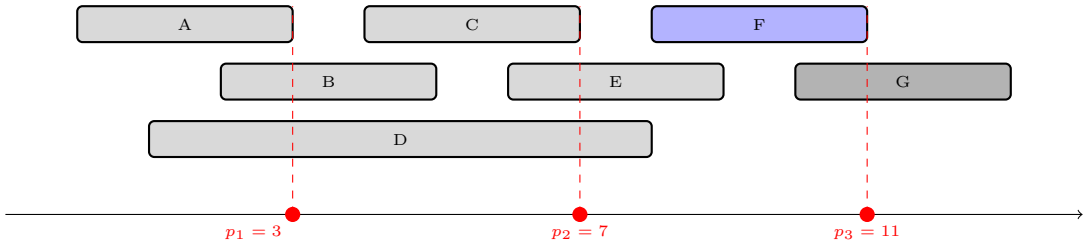
Unstabbed intervals: C, E, F, G.

Step 2: Among the unstabbed intervals, C has the earliest finish time (7). Place a point $p_2 = 7$. This stabs C and E.



Unstabbed intervals: F, G.

Step 3: Among the unstabbed intervals, F has the earliest finish time (11). Place a point $p_3 = 11$. This stabs F and G.



Unstabbed intervals: None. All intervals are stabbed.

Final Greedy Solution: $\{3, 7, 11\}$. Size = 3.

Proof of Optimality

Let's prove this strategy is always optimal using an **exchange argument**. The core idea is to show that the greedy choice is always "safe." We will show that we can transform any optimal solution into the greedy solution without increasing its size.

Theorem 2. *The greedy algorithm that repeatedly finds the unstabbed interval with the earliest finish time and places a point at that finish time produces an optimal solution for the Interval Stabbing problem.*

Let $G = \{p_1, p_2, \ldots, p_k\}$ be the solution produced by our greedy algorithm, sorted by position. Let $O = \{q_1, q_2, \ldots, q_m\}$ be any optimal solution, also sorted by position. We want to prove that $k = m$.

Our proof will show that the greedy algorithm "stays ahead." Specifically, we will prove by induction that for all $i \geq 1$, the point p_i is at least as far to the right as q_i.

Claim: For all $i = 1, \ldots, m$, we have $p_i \geq q_i$.

Base case ($i = 1$): Show that $p_1 \geq q_1$.

Hint: Let i_1 be the first interval chosen by the greedy algorithm (the one with the earliest finish time). Where does the greedy algorithm place p_1? Where must an optimal solution place a point q_j to stab i_1? How do these positions relate?

Proof. Let i_1 be the interval with the earliest finish time overall. The greedy algorithm selects $p_1 = f_{i_1}$ (the finish time of i_1). Now, consider the optimal solution O. Some point in O, say q_j, must stab the interval i_1. For q_j to stab i_1, it must be that $s_{i_1} \leq q_j \leq f_{i_1}$. Since the points in O are sorted, q_1 is the leftmost point. So, $q_1 \leq q_j$. Combining these, we have $q_1 \leq q_j \leq f_{i_1} = p_1$. Therefore, $p_1 \geq q_1$. The base case holds. □

Inductive Step: Assume that $p_{i-1} \geq q_{i-1}$ for some $i > 1$. We want to show that $p_i \geq q_i$.

Before we prove the inductive step, let's first show the following lemma:

Lemma 3. Suppose the greedy algorithm, at step j, selects i^*, the unstabbed interval with the earliest finish time, and places a point $p_j = f_{i^*}$ at that finish time of i^*. If an interval i' remains unstabbed after placing p_j, then i' must start strictly after p_j; that is, $p_j < s_{i'}$.

Proof. We proceed by contradiction. Assume that i' is an unstabbed interval after placing the point $p_j = f_{i^*}$, but it starts before or at p_j: $s_{i'} \leq p_j$

Since i' is unstabbed, the point p_j must not cover it. Therefore, since i' is not covered by p_j and we have assumed $s_{i'} \leq p_j$, it must be that p_j is strictly greater than the finish time of i':

$$p_j > f_{i'}$$

Substituting $p_j = f_{i^*}$, we have:

$$f_{i^*} > f_{i'}$$

This implies that i' has an earlier finish time than i^* (i.e., $f_{i'} < f_{i^*}$). Furthermore, since i' was unstabbed *before* step j (as it remains unstabbed *after* placing p_j), and it has an earlier finish time than i^*, this contradicts the greedy choice of i^*. The greedy algorithm specifically chose i^* because it was the unstabbed interval with the earliest finish time. Thus, our initial assumption ($s_{i'} \leq p_j$) must be false.

Therefore, any interval i' that remains unstabbed must satisfy $p_j < s_{i'}$. □

Using this lemma, complete the argument for the inductive step.

Proof. By the induction hypothesis, we assume $p_{i-1} \geq q_{i-1}$.

Let S_{i-1} be the set of all intervals stabbed by the first $i-1$ greedy points $\{p_1, \ldots, p_{i-1}\}$. The greedy algorithm's i-th step is to consider all intervals not in S_{i-1}. Let's call this set of remaining intervals R_i. The algorithm then finds the interval i^* in R_i with the earliest finish time and sets $p_i = f_{i^*}$. This means its start time must be after all of those points, i.e., $s_{i^*} > p_j$ for all $j < i$ via Lemma 3.

Now, consider the optimal solution's point q_i. By our claim $p_{i-1} \geq q_{i-1}$ and the sorted nature of the points, any interval stabbed by $\{q_1, \ldots, q_{i-1}\}$ must start before q_{i-1}, and thus must also start before p_{i-1}.

Since we proved $p_j \geq q_j$ for all $j < i$, it must be that $s_{i^*} > q_j$ for all $j < i$. This means that i^* could not have been stabbed by any of the first $i-1$ points of the optimal solution. Therefore, some point q_j with $j \geq i$ must stab i^*. So, we must have $s_{i^*} \leq q_j \leq f_{i^*}$. Since the points are sorted, $q_i \leq q_j$. This gives us $q_i \leq q_j \leq f_{i^*} = p_i$. So, $p_i \geq q_i$. The inductive step holds. □

Conclusion of the Proof: We have shown that $p_i \geq q_i$ for all $i = 1, \ldots, m$. How does this prove that the greedy solution is optimal (i.e., that $k = m$)?

Hint: Assume for contradiction that the greedy algorithm is not optimal. What would that imply about the relationship between k and m? Can you use the claim to find a contradiction?

Proof. Assume for contradiction that the greedy solution is not optimal. This would mean it uses more points than the optimal solution, so $k > m$. If $k > m$, then the greedy algorithm produced at least $m+1$ points: $\{p_1, \ldots, p_m, p_{m+1}\}$. After the m-th step of the greedy algorithm, there must have been at least one unstabbed interval, let's call it $i_{unstabbed}$, which caused the algorithm to place the point p_{m+1}. For $i_{unstabbed}$ to be unstabbed by $\{p_1, \ldots, p_m\}$, its start time must be greater than all of those points: $s_{i_{unstabbed}} > p_j$ for all $j \leq m$.

From our claim, we know that $p_j \geq q_j$ for all $j \leq m$. Therefore, $s_{i_{unstabbed}} > p_m \geq q_m$. This implies $s_{i_{unstabbed}} > q_j$ for all $j = 1, \ldots, m$. This means that the interval $i_{unstabbed}$ cannot be stabbed by any of the points in the optimal solution $O = \{q_1, \ldots, q_m\}$. This

is a contradiction, because O is supposed to be a valid solution that stabs *all* intervals. Therefore, our assumption that $k > m$ must be false. Since the greedy algorithm produces a valid solution, we know $k \geq m$. The only possibility is that $k = m$. □