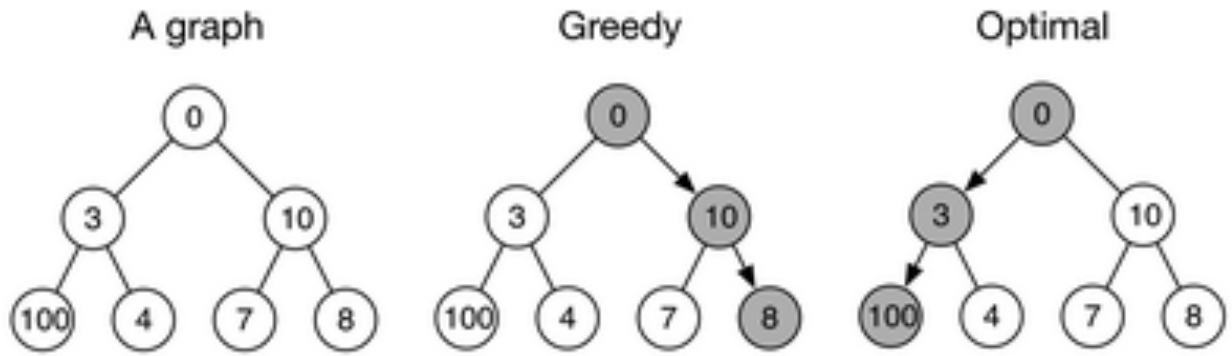
CSCI 3104: Algorithms

Lecture 10: Intro to Greedy Algorithms

Rachel Cox

Department of Computer Science



A greedy algorithm fails to maximise the sum of nodes along a path from the top to the bottom because it lacks the foresight to choose suboptimal solutions in the current iteration that will allow for better solutions later

Source

What will we learn today?

- ❑ Greedy Algorithms

Intro to Algorithms, CLRS:
Sections 16.1, 16.2

Greedy Algorithms

What is a Greedy Algorithm?

- An algorithm that seeks to produce an optimal global solution by making the best local choice.

- Sometimes greedy algos produce an optimal solution, and sometimes they don't

- In this lecture we examine the activity-selection problem which is also known as interval scheduling.

Greedy Algorithms

Greedy Choice Property: A global optimum can be arrived at by selecting a local optimum incrementally without referencing past or future decisions.

- greedy local choice, not considering any results from subproblems.

Optimal Sub-Structure Property: An optimal solution to the problem contains an optimal solution to the sub-problems

Activity / Interval Scheduling

ending time

- Suppose you are given a list of activities $(s_1, e_1), (s_2, e_2), \ldots, (s_n, e_n)$ denoted by their start and end times.

starting time

- All activities are equally attractive to you, and you want to maximize the number of activities you do.

* Here the optimal solution means having a schedule with the most activities

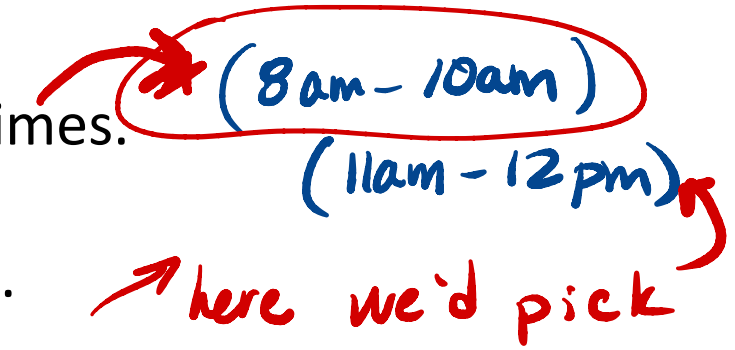
- Goal: Choose the largest number of non-overlapping activities possible.

e.g. booking a venue
- weddings
- birthday party
- concert

Activity / Interval Scheduling

- If we want to try solving this using a Greedy approach, we should think about different ways of picking activities greedily.

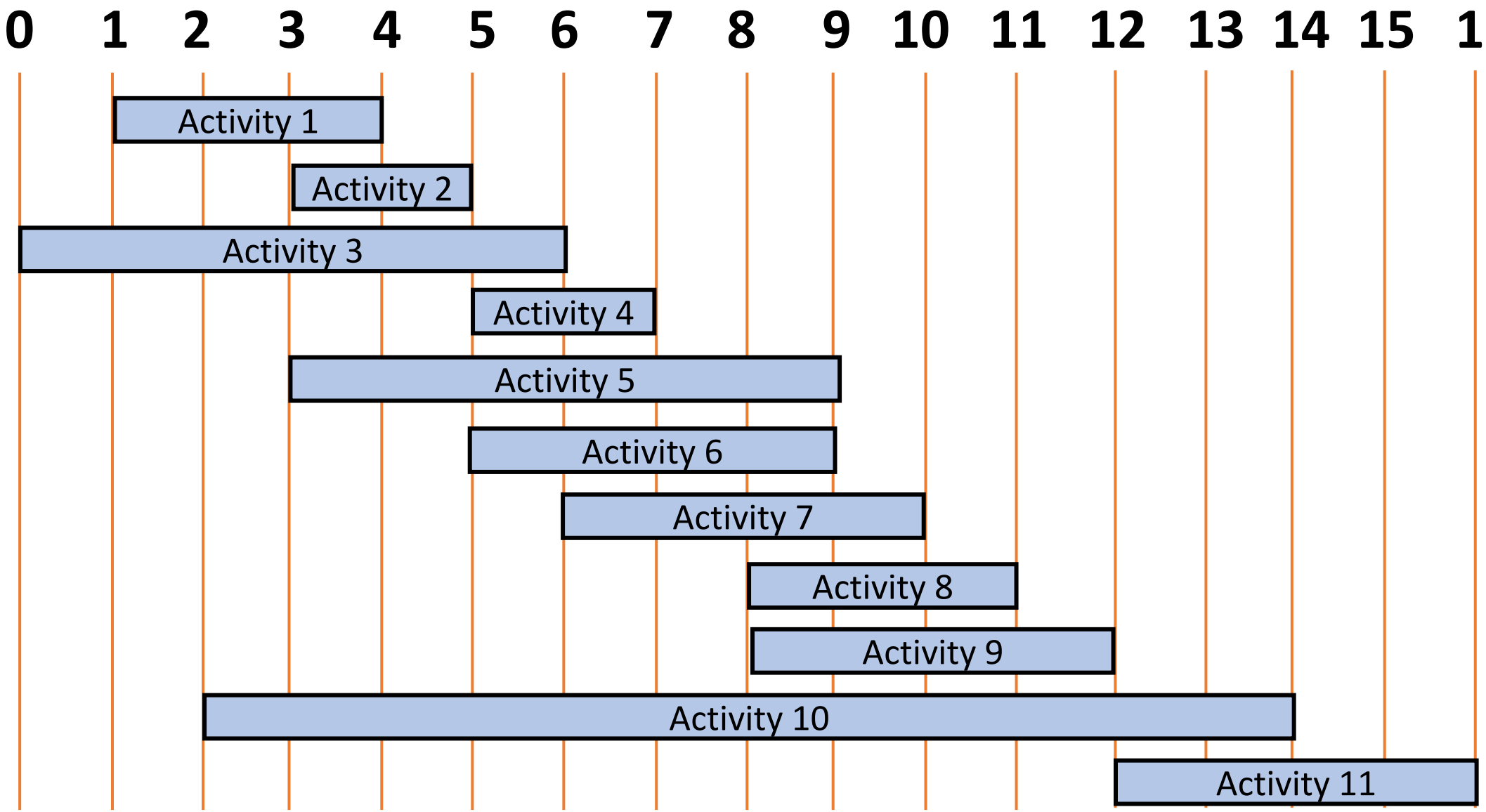
- Choose activities in ascending order of start times.



- Choose activities in ascending order of length.

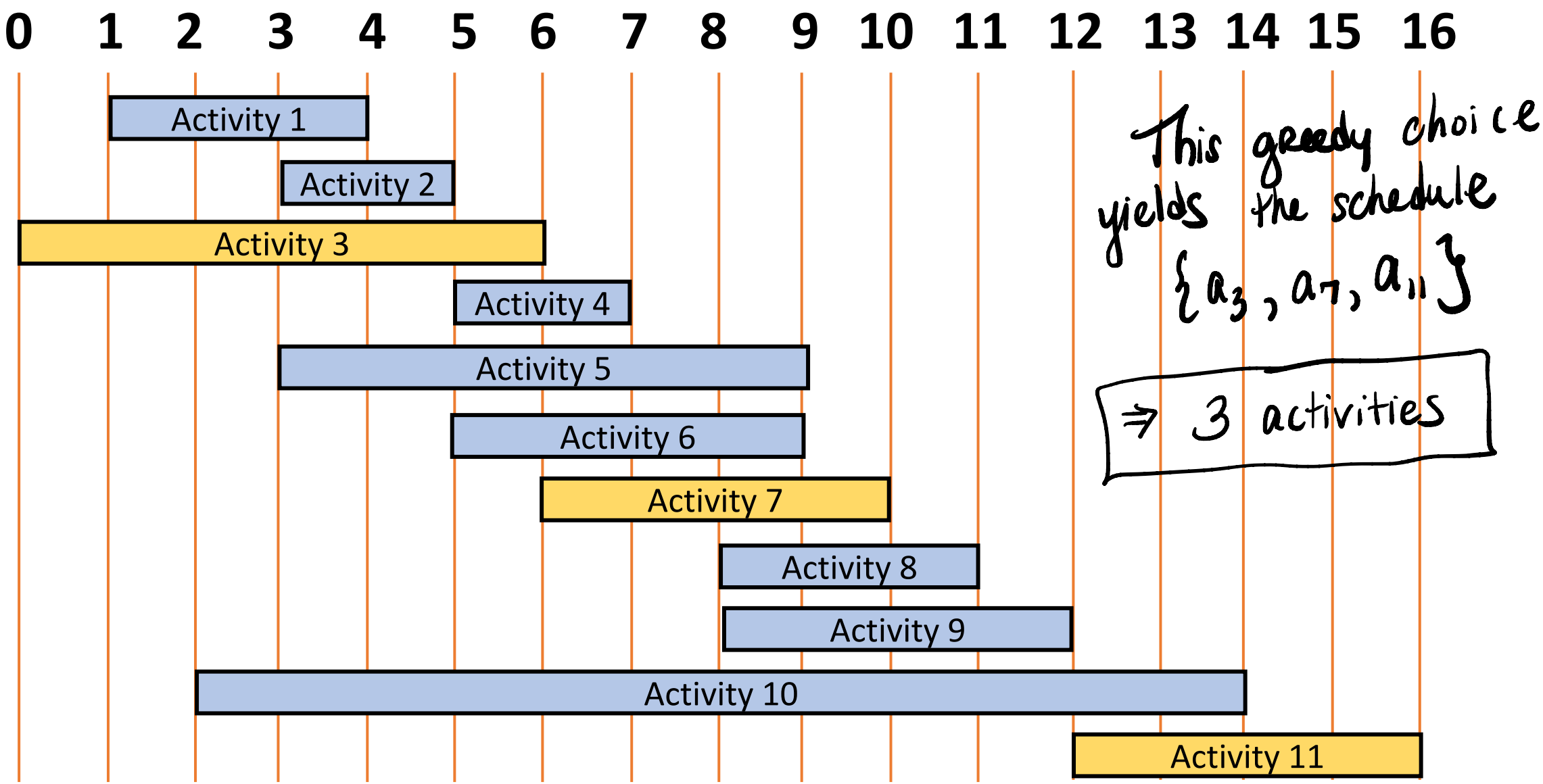
- Choose activities in ascending order of end times.

Activity / Interval Scheduling



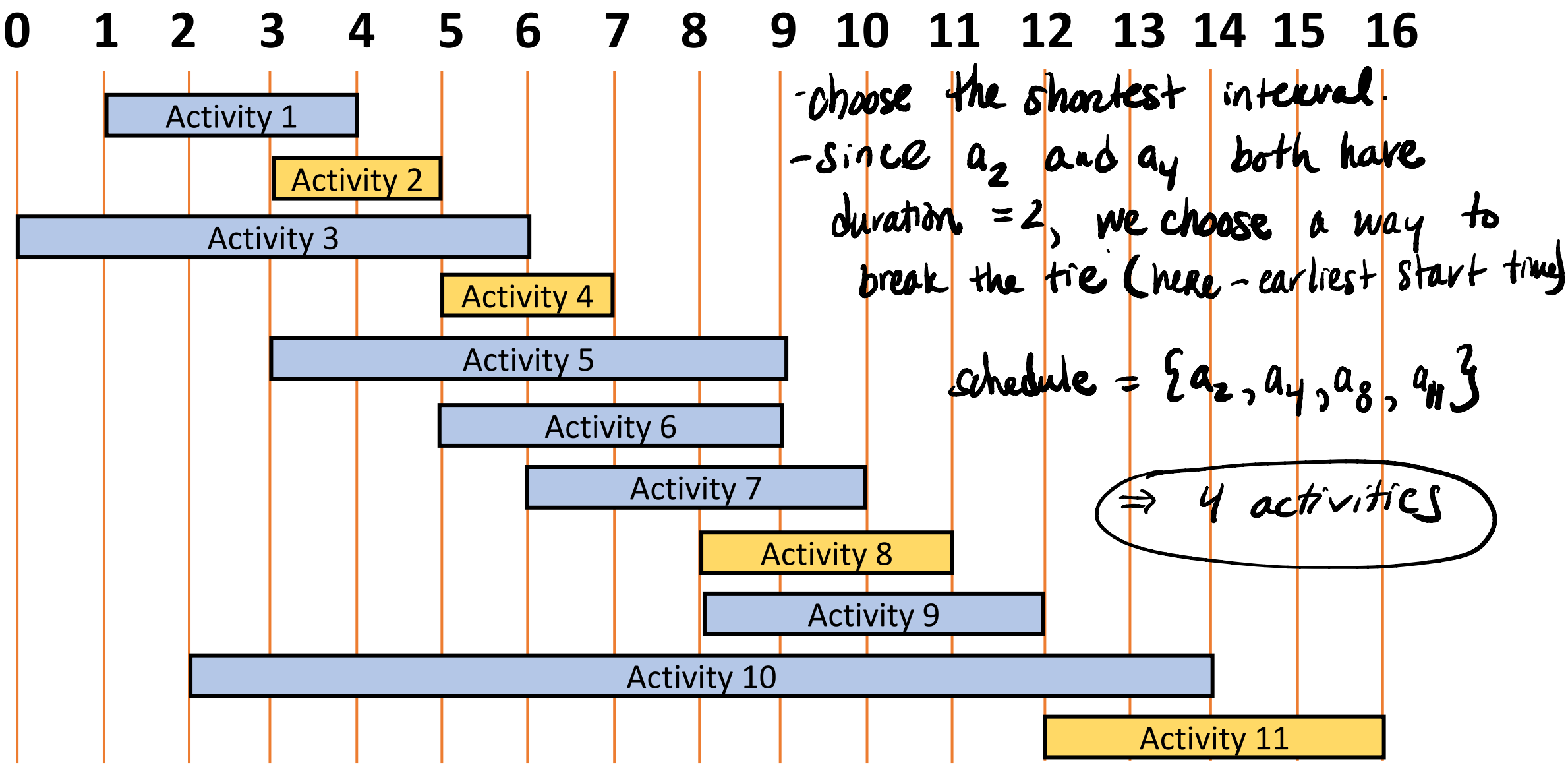
Activity / Interval Scheduling

Ascending order of start time



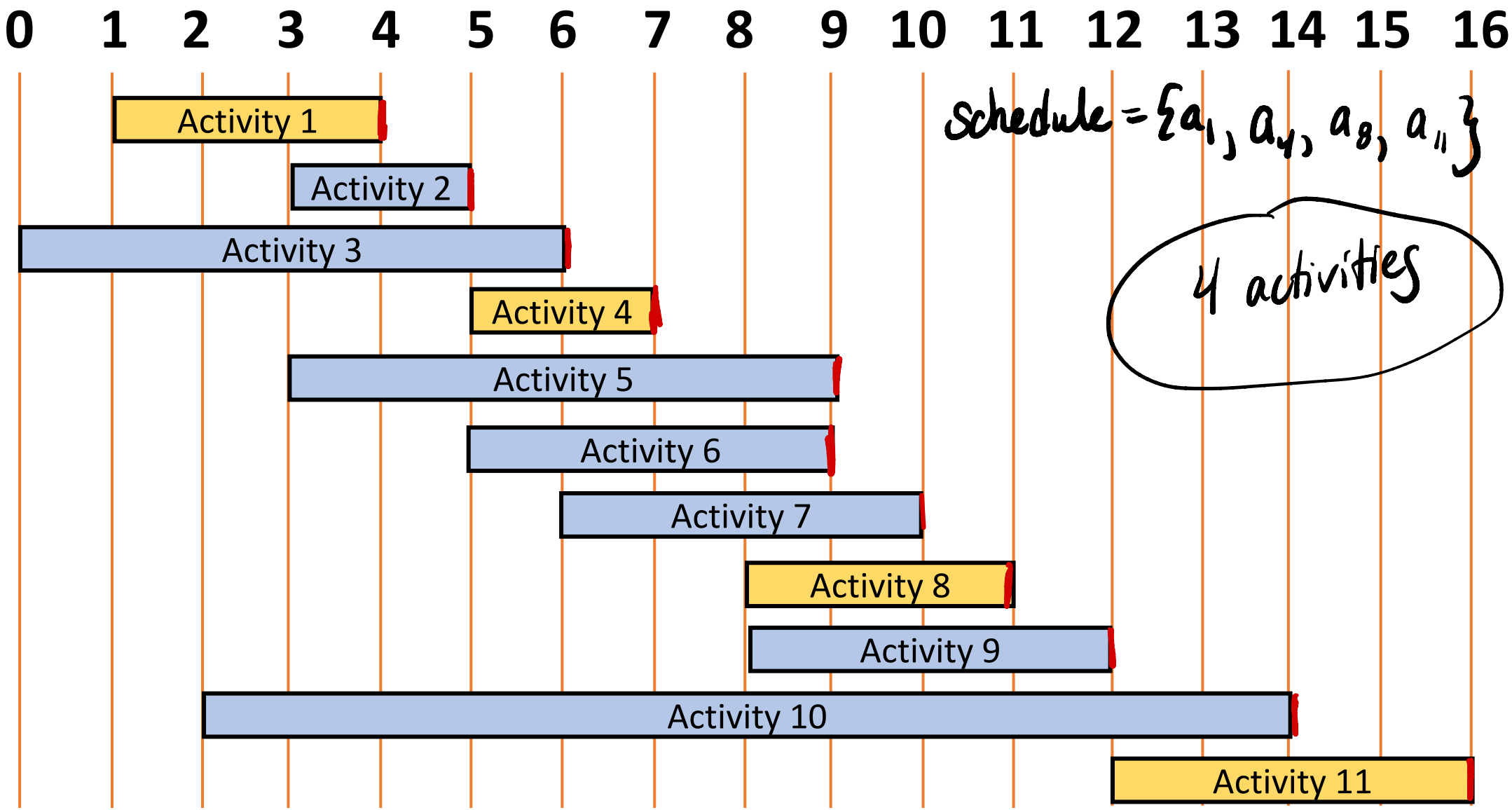
Activity / Interval Scheduling

Ascending order of length



Activity / Interval Scheduling

Ascending order of finishing time



Activity / Interval Scheduling

Of the three options we saw, the third one yields an optimal solution for any scenario:

- **Choose activities in ascending order of finishing times.**

counterexample

why doesn't earliest starting time work? consider [0,20], [1,5], [7,9]

Idea:

intuition for why: by choosing the activity with the earliest ending time, we maximize the amount of time left to choose other activities.

- Sort the activities into ascending order by finishing time and add them to a set U.

- While U is not empty:
  - Choose any activity with the earliest finishing time.
  - Add that activity to S.
  - Remove from U all activities that overlap with S.

U - universe (all available activities)

S - our schedule

Proving Optimality

Two Methods:

- Show that the greedy algorithm's measures are at least as good as any solution's measures. Because of that, show that the greedy solution must be optimal.

• greedy algorithm "stays ahead"

- Consider a possible optimal solution to the problem and transform it into a solution found by the greedy algorithm without hurting optimality. It follows that the greedy algorithm must have found a solution at least as good as any other solution. (exchange argument)

Proving Optimality

Observation: The kth activity chosen by the greedy algorithm finishes no later than the kth activity chosen in any legal schedule

legal - in this context indicates non-overlapping activities

- why?: the greedy algorithm is designed to choose intervals with the earliest finishing time

Proving Optimality

Notation:

- Let S be the schedule our algorithm produces and S^* be any optimal schedule.
- Note that $|S| \leq |S^*|$

This means that the number of activities in our schedule S is less than or equal to the number of activities in an optimal schedule.

- Let $f(i, S)$ denote the time that the i^{th} activity finishes in schedule S.

Lemma: For any $1 \leq i \leq |S|$, we have $f(i, S) \leq f(i, S^*)$

Proving Optimality

Lemma: If S is a schedule produced by the greedy algorithm and S^* is an optimal schedule, then for any $1 \leq i \leq |S|$, we have $f(i, S) \leq f(i, S^*)$.

(nicely typed proof on next page…we work through the idea here.)

Proof: By induction. We consider the finish times of the activities in the Greedy solution versus the finish times of the activities in the optimal solution.

Greedy: $f(1,S)$, $f(2,S)$ … $f(r,S)$

Optimal: $f(1,S^*)$, $f(2,S^*)$ … $f(r,S^*)$ … $f(m,S^*)$

where $m \geq r$ because the optimal solution has the max. number of activities.

Base case: $f(1,S) \leq f(1,S^*)$

Since the greedy algorithm selects the activity with the earliest finish time, this must be true.

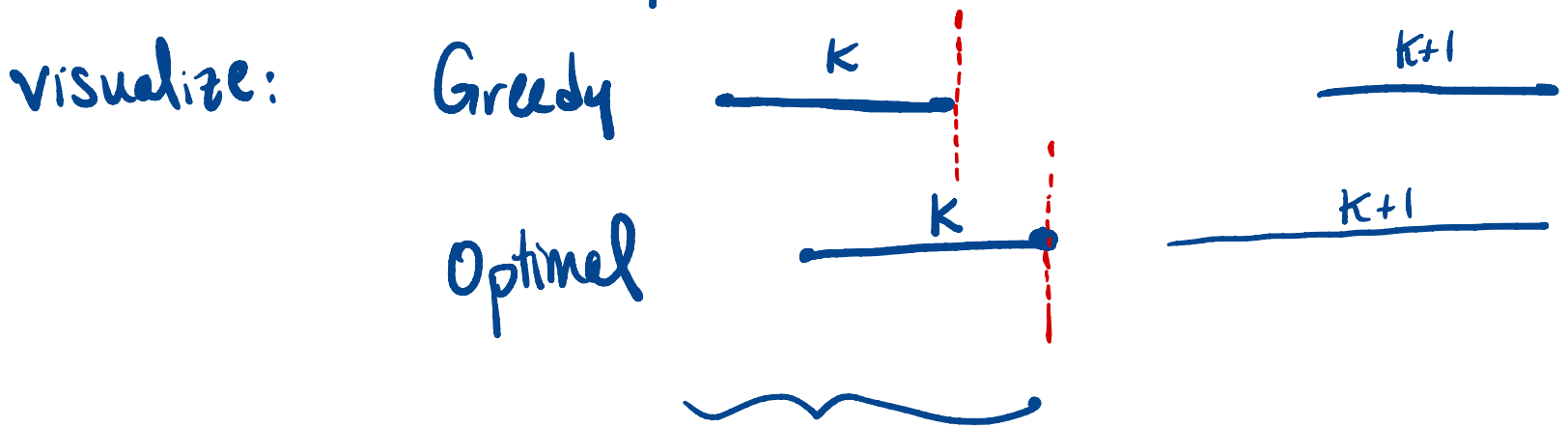
Proving Optimality

Lemma: If S is a schedule produced by the greedy algorithm and S^* is an optimal schedule, then for any $1 \leq i \leq |S|$, we have $f(i, S) \leq f(i, S^*)$.
(nicely typed proof on next page…we work through the idea here.)

Proof: By induction.

Induction Step: Inductive Hypothesis: Assume that $f(k, S) \leq f(k, S^*)$

visualize:



By the induction hypothesis, k^{th} choice of Greedy finishes before the k^{th} choice of the optimal.

For the $k+1$ activity, the Greedy algorithm will have a choice and it will always choose the $(k+1)^{st}$ activity with earliest finish time.

$\Rightarrow f(k+1, S) \leq f(k+1, S^*)$

Proving Optimality

Lemma: If S is a schedule produced by the greedy algorithm and S^* is an optimal schedule, then for any $1 \leq i \leq |S|$, we have $f(i, S) \leq f(i, S^*)$.

Proof: By induction. For our base case, we prove that $f(1, S) \leq f(1, S^*)$. The first activity that the greedy algorithm selects must be an activity that ends no later than any other activity, so $f(1, S) \leq f(1, S^*)$.

For the inductive step, assume the claim holds for some i in $1 \leq i < |S|$. Since $f(i, S) \leq f(i, S^*)$, the i^{th} activity in S finishes before the i^{th} activity in S^*.

Since the $(i+1)^{st}$ activity in S^* must start after the i^{th} activity in S^* ends. Therefore, the $(i+1)^{st}$ activity in S^* must be in U when the greedy algorithm selects its $(i+1)^{st}$ activity.

Since the greedy algorithm selects the activity in U with the lowest end time, we have $f(i+1, S) \leq f(i+1, S^*)$, completing the induction. ■

Proving Optimality

Theorem: The greedy algorithm for activity selection produces an optimal schedule.
(nicely typed proof on next page…we work through the idea here.)

Proof: We just proved that $f(i,S) \leq f(i,S^*)$ for any $1 \leq i \leq |S|$

Let's use a proof by contradiction.

Since S^* is optimal, $|S| \leq |S^*|$

- For the theorem, we want to show that $|S| \geq |S^*|$

For the contradiction, we assume that the greedy algorithm does not produce an optimal schedule.

$\Rightarrow |S| < |S^*|$

Let $k = |S|$

- By the lemma we just proved, $f(k,S) \leq f(k,S^*)$
  - $\Rightarrow$ the k^{th} activity in S finishes no later than the k^{th} activity in S^*

Proving Optimality

Theorem: The greedy algorithm for activity selection produces an optimal schedule.
(nicely typed proof on next page…we work through the idea here.)

Proof: Since our assumption say $|S| < |S^*|$, there must be at least one more activity in S^*

There must be a $(k+1)^{st}$ activity in S^*

Since this is a legal schedule, the $(k+1)^{st}$ activity has a start time after $f(k, S^*)$

$\Rightarrow$ the $(k+1)^{st}$ activity also has a start time after $f(k, S)$

So after the greedy algorithm added activity k to S, the $(k+1)$ activity from S^* would still belong to U.

But since the greedy algorithm ended after k activities, U must be empty.

$\Rightarrow$ can't be a $(k+1)^{st}$ activity is S^* $\Rightarrow$ not the case that $|S| < |S^*|$

Proving Optimality

Theorem: The greedy algorithm for activity selection produces an optimal schedule.

Proof: Let S be the schedule the algorithm produced and let S^* be any optimal schedule. Since S^* is optimal, we have $|S| \leq |S^*|$. We will prove that $|S| \geq |S^*|$.

Assume for a contradiction that $|S| < |S^*|$. Let $k = |S|$. By our lemma, we know that $f(k, S) \leq f(k, S^*)$, so the k^{th} activity in S finishes no later than the k^{th} activity in S^*. Since $|S| < |S^*|$, there is a $(k+1)^{st}$ activity in S^*, and its start time must be after $f(k, S^*)$ and therefore after $f(k, S)$.

Thus, after the greedy algorithm added its k^{th} activity to S, the $(k+1)^{st}$ activity from S^* would still belong to U. But the greedy algorithm ended after k activities, so U must have been empty.

We have reached a contradiction, so our assumption must have been wrong. Thus the greedy algorithm must be optimal. ■

Proving Optimality

The style of proof we just wrote is an example of a **greedy stays ahead** proof.

The general proof structure is the following:

- Find a series of measurements $M_1, M_2, \ldots, M_k$ you can apply to any solution.
- Show that the greedy algorithm's measures are at least as good as any solution's measures.
- Prove that because the greedy solution's measures are at least as good as any solution's measures, the greedy solution must be optimal.

Interval Scheduling

Example: Suppose we have a number of events m_i. Each event starts at time s_i and finishes at time e_i, where $0 \leq s_i < e_i$. We represent the event m_i with the closed interval $[s_i, e_i]$. Our goal is to construct a maximum size sett of events, where no two events in the set overlap.

Suppose the following intervals are provided:

Event Index	Interval
1	[1, 2]
2	[3, 4]
3	[5, 6]
4	[7, 8]
5	[0, 20]

Suppose we sort the intervals in ascending order by start time. Consider a greedy algorithm that selects the next event based on earliest start time, so long as the interval selected does not conflict with any previously selected interval. Using the intervals provided, show that this greedy algorithm fails to provide a maximum size sett of events, where no two events in the set overlap. That is, the solution returned by this greedy algorithm is not optimal.

S: Not optimal

The event with the earliest start time is a_5

$S = \{a_5\}$ (1 activity)

Optimal

consider ascending sort by end time

$S^* = \{[1,2], [3,4], [5,6], [7,8]\}$

4 activities

Interval Scheduling

Example: Using the same definition as the previous slide, suppose the following intervals are provided:

Event Index	Interval
1	[1, 10]
2	[11, 20]
3	[21, 30]
• 4	[9, 12]
• 5	[19, 22]

Suppose we sort the intervals in ascending order by interval length. For events with the same length, order by start time. Consider a greedy algorithm that selects the next interval based on the smallest interval length, so long as the interval selected does not conflict with any previously selected interval. Using the intervals provided, show that this greedy algorithm fails to provide a maximum size set of events, where no two events in the set overlap. That is, the solution returned by this greedy algorithm is not optimal.

Optimal (end time)

{[1,10], [11,20], [21,30]}

3 activities

Ascending order by interval length:

{[9,12], [19,22]}

2 activities

2 < 3 ⇒ not optimal

Interval Scheduling

Example: Using the same definition as the previous slides:

Event Index	Interval
1	[1, 3]
2	[4, 6]
3	[7, 9]
4	[10, 12]
5	[2, 5]
6	[2, 5]
7	[2, 5]
8	[5.5, 7.5]
9	[8, 11]
10	[8, 11]
11	[8, 11]

Let c_i denote the number of intervals on our list in which interval i conflicts. For example, interval 1 participates in 3 conflicts: with intervals 5, 6, and 7. So $c_1 = 3$.

Suppose we sort the intervals in ascending order based on the number of conflicts. So if $c_i < c_j$, then the interval i comes before the interval j. Consider a greedy algorithm that selects the next interval based on the smallest number of conflicts, so long as the interval selected does not conflict with any previously selected interval. Using the intervals provided, show that this greedy algorithm fails to provide a maximum size set of events, where no two events in the set overlap. That is, the solution returned by this greedy algorithm is not optimal.

Interval Scheduling

Example: Using the same definition as the previous slides:

Event Index	Interval
1	[1, 3]
2	[4, 6]
3	[7, 9]
4	[10, 12]
5	[2, 5]
6	[2, 5]
7	[2, 5]
8	[5.5, 7.5]
9	[8, 11]
10	[8, 11]
11	[8, 11]

$c_1 = 3$ (conflicts 5, 6, 7)

$c_2 = 4$ (conflicts 5, 6, 7, 8)

$c_3 = 4$ (conflicts 8, 9, 10, 11)

$c_4 = 3$

$c_5 = 4$

$c_6 = 4$

$c_7 = 4$

$c_8 = 2$

$c_9 = 4$

$c_{10} = 4$

$c_{11} = 4$

Sort by ascending order of conflicts:

$c_8 < c_1 = c_4 < c_2 = c_3 = c_5 = c_6 = c_7 = c_9 = c_{10} = c_{11}$

- We start selecting based on this criteria
  - → we pick 8 first.
  - → then we choose 1 and 4
  - ⇒ activities = 3

Optimal

1, 2, 3, 4

4 activities

Proving Optimality

To recap:

- Greedy algorithms aim for global optimality by iteratively making a locally optimal decision.

- To show correctness, we typically need to show:
  - The algorithm produces a legal answer. (induction)
  - The algorithm produces an optimal answer. ★

Next Time

- Huffman Encoding