Greedy Algorithms
Part One

Announcements

- Problem Set Three due right now if using a late period.
- Solutions will be released at end of lecture.

Outline for Today

- **Greedy Algorithms**
  - Can myopic, shortsighted decisions lead to an optimal solution?
- **Lilypad Jumping**
  - Helping our amphibious friends home!
- **Activity Selection**
  - Planning your weekend!

Frog Jumping

- The frog begins at position 0 in the river. Its goal is to get to position n.
- There are lilypads at various positions. There is always a lilypad at position 0 and position n.
- The frog can jump at most r units at a time.
- **Goal:** Find the path the frog should take to minimize jumps, assuming a solution exists.

Formalizing the Algorithm

- Let J be an empty series of jumps.
- Let our current position $x = 0$.
- While $x < n$:
  - Find the furthest lilypad l reachable from x that is not after position n.
  - Add a jump to J from x to l's location.
  - Set x to l's location.
- Return J.

Greedy Algorithms

- A **greedy algorithm** is an algorithm that constructs an object X one step at a time, at each step choosing the locally best option.
- In some cases, greedy algorithms construct the globally best object by repeatedly choosing the locally best option.

Greedy Advantages

- Greedy algorithms have several advantages over other algorithmic approaches:
    - **Simplicity**: Greedy algorithms are often easier to describe and code up than other algorithms.
    - **Efficiency**: Greedy algorithms can often be implemented more efficiently than other algorithms.

Greedy Challenges

- Greedy algorithms have several drawbacks:
  - **Hard to design**: Once you have found the right greedy approach, designing greedy algorithms can be easy. However, finding the right approach can be hard.
  - **Hard to verify**: Showing a greedy algorithm is correct often requires a nuanced argument.

Back to Frog Jumping

- We now have a simple greedy algorithm for routing the frog home: jump as far forward as possible at each step.
- We need to prove two properties:
  - The algorithm will find a legal series of jumps (i.e. it doesn't “get stuck”).
  - The algorithm finds an *optimal* series of jumps (i.e. there isn't a better path available).

Lemma 1: The greedy algorithm always finds a path from the start lilypad to the destination lilypad.

Proof: By contradiction; suppose it did not. Let the positions of the lilypads be $x_1 < x_2 < \dots < x_m$. Since our algorithm didn't find a path, it must have stopped at some lilypad x_k and not been able to jump to a future lilypad. In particular, this means it could not jump to lilypad $k + 1$, so $x_k + r < x_{k+1}$.

Since there is a path from lilypad 1 to the lilypad m, there must be some jump in that path that starts before lilypad $k + 1$ and ends at or after lilypad $k + 1$. This jump can't be made from lilypad k, so it must have been made from lilypad s for some $s < k$. But then we have $x_s + r < x_k + r < x_{k+1}$, so this jump is illegal.

We have reached a contradiction, so our assumption was wrong and our algorithm always finds a path. ■

Proving Optimality

- How can we prove this algorithm finds an optimal series of jumps?
- **Key Proof Idea**: Consider an arbitrary optimal series of jumps J^*, then show that our greedy algorithm produces a series of jumps no worse than J^*.
  - We don't know what J^* is or that our algorithm is necessarily optimal. However, we can still use the existence of J^* in our proof.

Some Notation

- Let J be the series of jumps produced by our algorithm and let J^* be an optimal series of jumps.
  - Note that there might be multiple different optimal jump patterns.
- Let $|J|$ and $|J^*|$ denote the number of jumps in J and J^*, respectively.
- Note that $|J| \geq |J^*|$. *(Why?)*

The Key Lemma

- Let $p(i, J)$ denote the frog's position after taking the first i jumps from jump series J.
- ***Lemma:*** For any i in $0 \leq i \leq |J^*|$, we have $p(i, J) \geq p(i, J^*)$.
    - After taking i jumps according to the greedy algorithm, the frog will be at least as far forward as if she took i jumps according to the optimal solution.
- We can formalize this using induction.

Lemma 2: For all $0 \leq i \leq |J^*|$, we have $p(i, J) \geq p(i, J^*)$.

Proof: By induction. As a base case, if $i = 0$, then $p(0, J) = 0 \geq 0 = p(0, J^*)$ since the frog hasn't moved.

For the inductive step, assume that the claim holds for some $0 \leq i < |J^*|$. We will prove the claim holds for $i + 1$ by considering two cases:

Case 1: $p(i, J) \geq p(i + 1, J^*)$. Since each jump moves forward, we have $p(i + 1, J) \geq p(i, J)$, so we have $p(i + 1, J) \geq p(i + 1, J^*)$.

Case 2: $p(i, J) < p(i + 1, J^*)$. Each jump is of size at most r, so $p(i + 1, J^*) \leq p(i, J^*) + r$. By our IH, we know $p(i, J) \geq p(i, J^*)$, so $p(i + 1, J^*) \leq p(i, J) + r$. Therefore, the greedy algorithm can jump to position at least $p(i + 1, J^*)$. Therefore, $p(i + 1, J) \geq p(i + 1, J^*)$.

So $p(i+1, J) \geq p(i+1, J^*)$, completing the induction. ■

Theorem: Let J be the series of jumps produced by the greedy algorithm and J^* be any optimal series of jumps. Then $|J| = |J^*|$.

Proof: Since J^* is an optimal solution, we know that $|J^*| \leq |J|$. We will prove $|J^*| \geq |J|$.

Suppose for contradiction that $|J^*| < |J|$. Let $k = |J^*|$. By Lemma 2, we have $p(k, J^*) \leq p(k, J)$. Because the frog arrives at position n after k jumps along series J^*, we know $n \leq p(k, J)$. Because the greedy algorithm never jumps past position n, we know $p(k, J) \leq n$, so $n = p(k, J)$. Since $|J^*| < |J|$, the greedy algorithm must have taken another jump after its kth jump, contradicting that the algorithm stops after reaching position n.

We have reached a contradiction, so our assumption was wrong and $|J^*| = |J|$, so the greedy algorithm produces an optimal solution. ■

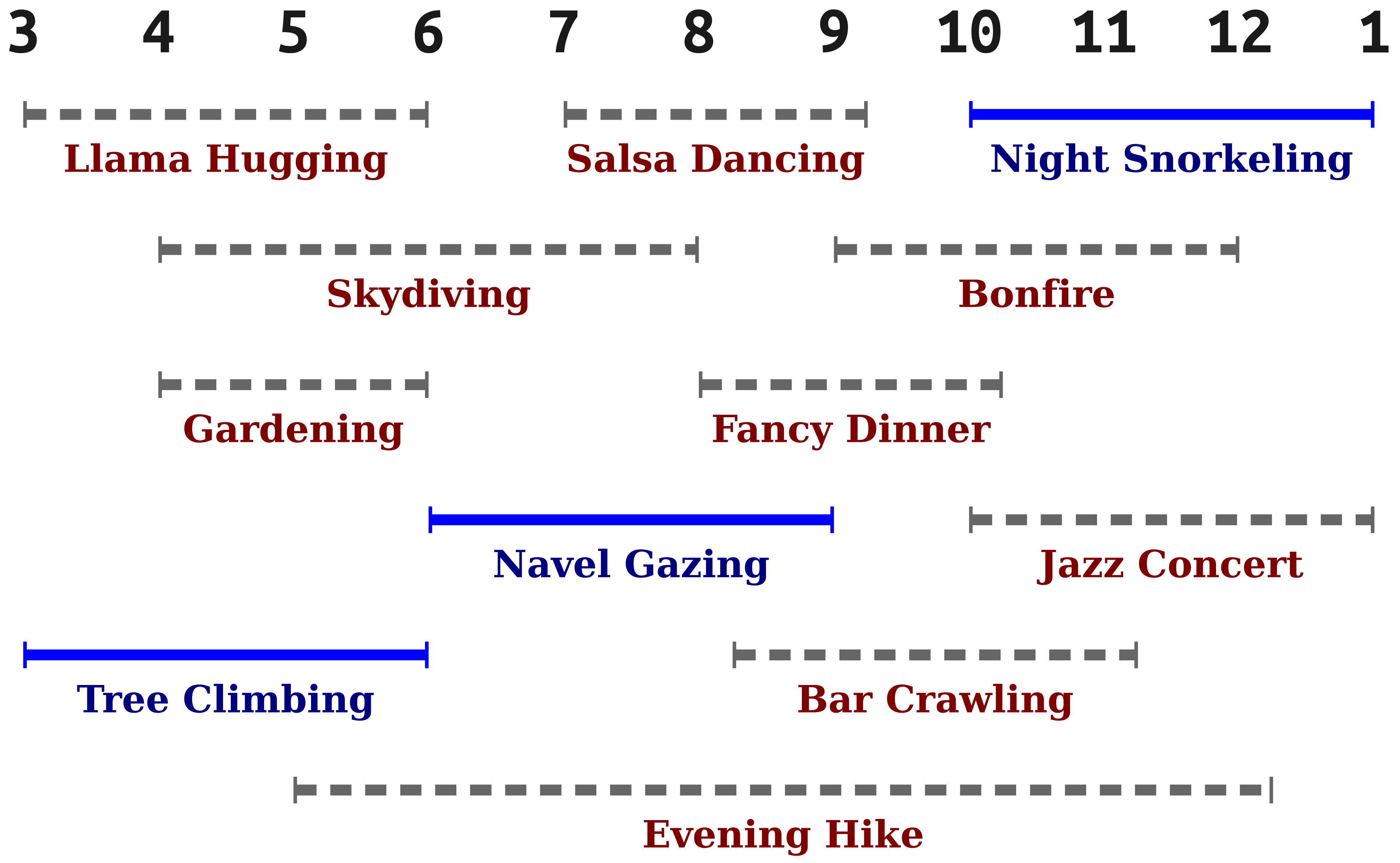
Greedy Stays Ahead

- 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, …, 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. (This usually involves induction.)
  - Prove that because the greedy solution's measures are at least as good as any solution's measures, the greedy solution must be optimal. (This is usually a proof by contradiction.)

Another Problem:

Activity Scheduling

Activity Scheduling



Activity Scheduling

- You are given a list of activities (s_1, e_1), (s_2, e_2), …, (s_n, e_n) denoted by their start and end times.
- All activities are equally attractive to you, and you want to maximize the number of activities you do.
- Goal: Choose the largest number of non-overlapping activities possible.

Thinking Greedily

- If we want to try solving this using a greedy approach, we should think about different ways of picking activities greedily.
- A few options:
  - **Be Impulsive:** Choose activities in ascending order of start times.
  - **Avoid Commitment:** Choose activities in ascending order of length.
  - **Finish Fast:** Choose activities in ascending order of end times.

Thinking Greedily

- Of the three options we saw, only the third one seems to work:

  Choose activities in ascending order of finishing times.

- More formally:
  - 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 S.

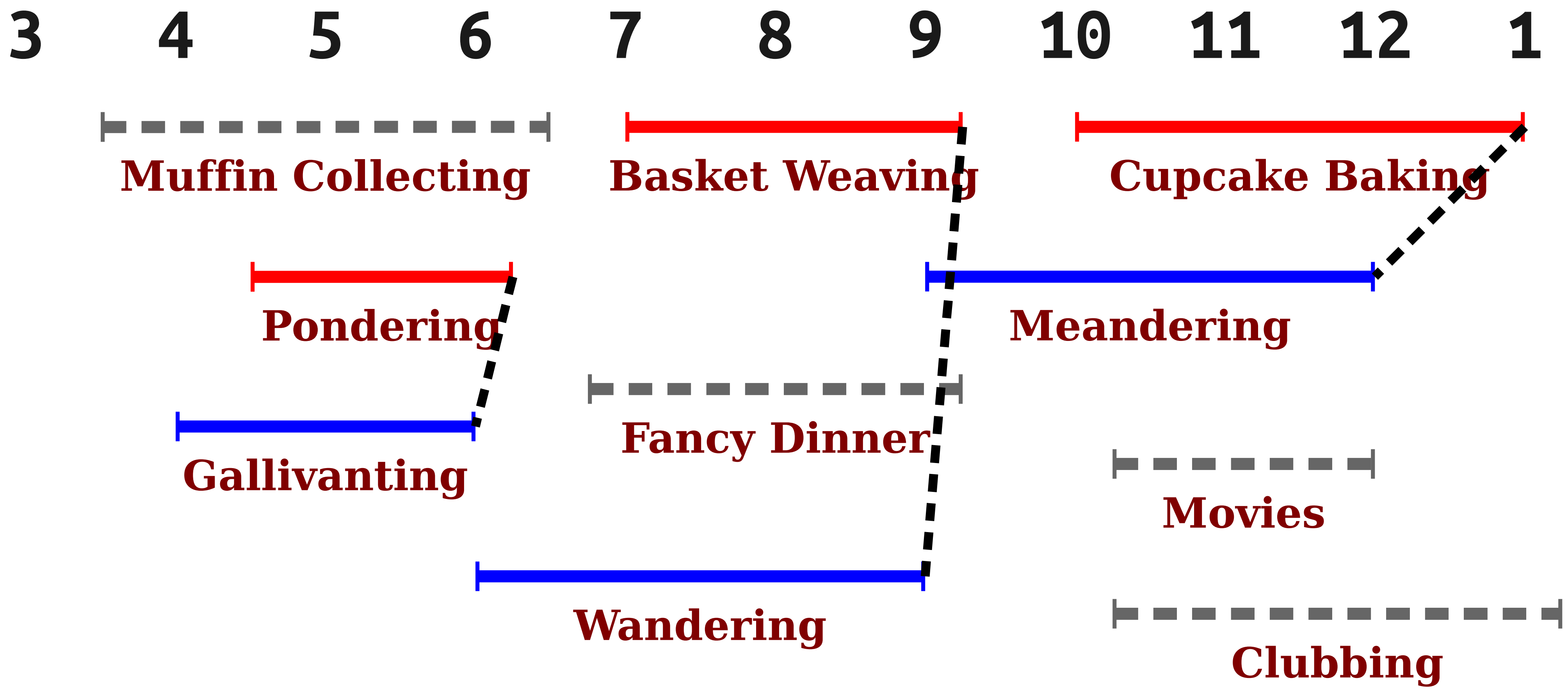
Proving Legality

- ***Lemma:*** The schedule produced this way is a legal schedule.
- ***Proof Idea:*** Use induction to show that at each step, the set U only contains activities that don't conflict with activities picked from S.

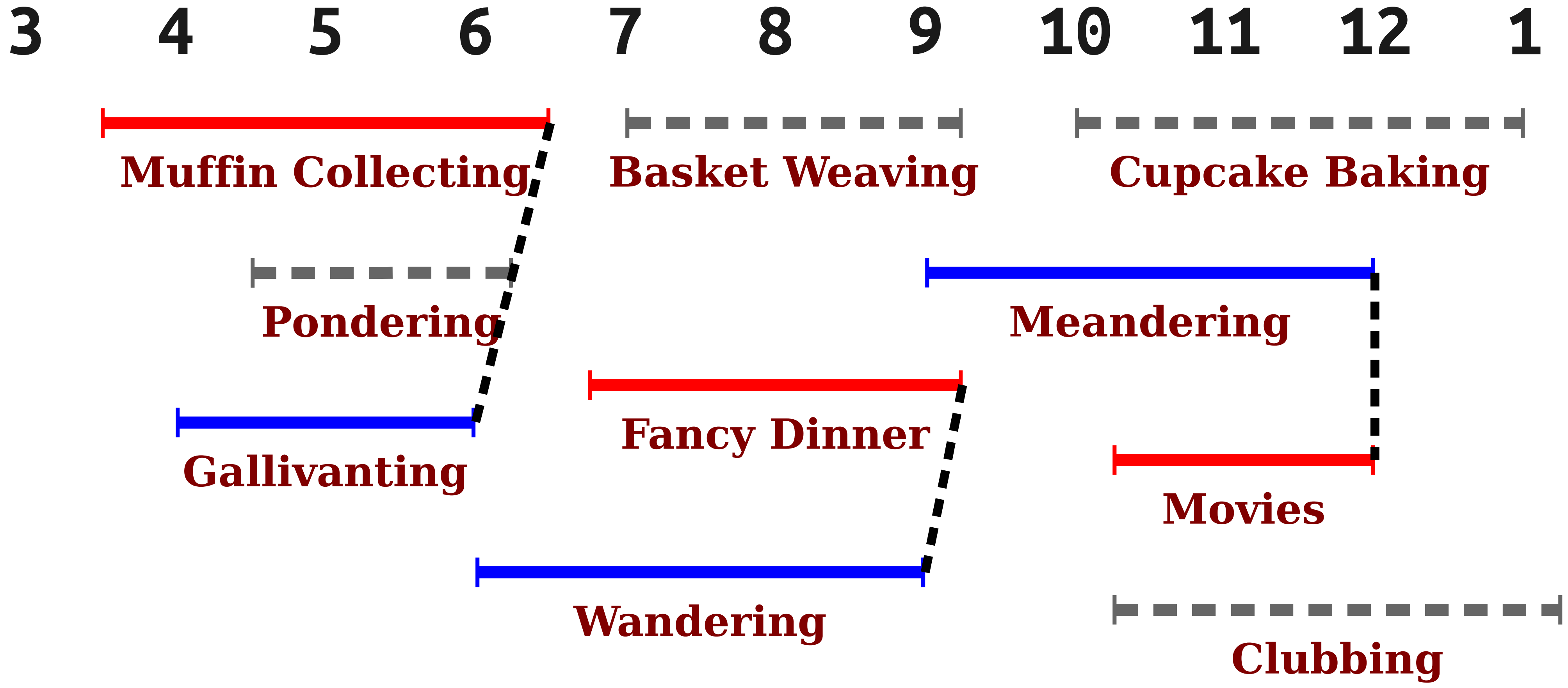
Proving Optimality

- To prove that the schedule *S* produced by the algorithm is optimal, we will use another "greedy stays ahead" argument:
  - Find some measures by which the algorithm is at least as good as any other solution.
  - Show that those measures mean that the algorithm must produce an optimal solution.

Comparing Solutions



Comparing Solutions



Greedy Stays Ahead

- **Observation:** The kth activity chosen by the greedy algorithm finishes no later than the kth activity chosen in any legal schedule.
- We need to
  - Prove that this is actually true, and
  - Show that, if it's true, the algorithm is optimal.
- We'll do this out of order.

Some Notation

- Let S be the schedule our algorithm produces and S^* be any optimal schedule.
- Note that $|S| \leq |S^*|$.
- Let $f(i, S)$ denote the time that the ith activity finishes in schedule S.
- ***Lemma:*** For any $1 \leq i \leq |S|$, we have $f(i, S) \leq f(i, S^*)$.

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

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

Assume for contradiction that $|S| < |S^*|$. Let $k = |S|$. By our lemma, we know $f(k, S) \leq f(k, S^*)$, so the kth activity in S finishes no later than the kth 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 kth 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. ■

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 $f(1, S) \leq f(1, S^*)$. The first activity 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 ith activity in S finishes before the ith activity in S^*. Since the $(i+1)$st activity in S^* must start after the ith activity in S^* ends, the $(i + 1)$st activity in S^* must start after the ith 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. ■

Summary

- Greedy algorithms aim for global optimality by iteratively making a locally optimal decision.
- To show correctness, typically need to show
  - The algorithm produces a *legal* answer, and
  - The algorithm produces an *optimal* answer.
- Often use "greedy stays ahead" to show optimality.

Next Time

- Minimum Spanning Trees
- Prim's Algorithm
- Exchange Arguments