Announcements

- Homework 1 Due Wednesday 11:59 pm
- Quiz 1 Due Tomorrow 11:59 pm
- Discussion on Friday

Plan

- Recap: Graphs, Traversal, etc.
- Greedy algorithms

Graphs Recap

- Simple definitions: vertex/node, edge, path, cycle, tree, path, components
- Algorithms: breadth first search, depth first search, and applications
- More complex: Bipartite, DAG, Topological ordering, Find CCs
- Also: Pseudocode, implementations, running time analysis.

Problem 1: Interval Scheduling

- In the 80s, your only opportunity to watch a specific TV show was the time it was broadcast. Unfortunately, on a given night there might be multiple shows that you want to watch and some of the broadcast times overlap.
- You want to watch the highest number of shows. Which subset of shows do you pick?
- Notation: \( n \) shows, let show \( j \) start at time \( s_j \) and finish at time \( f_j \) and we say two shows are compatible if they don’t overlap.
- Assume you like all shows equally, you only have one TV, and you need to watch shows in their entirety.

Interval Scheduling

- Notation: \( n \) shows, let show \( j \) start at time \( s_j \) and finish at time \( f_j \) and we say two shows are compatible if they don’t overlap.
- How do we find the maximum subset of shows that are all compatible? (e.g., How do we watch the most shows?)
- Example: \([1, 4], [2, 3], [2, 7], [4, 7], [3, 6], [6, 10], [5, 7]\)
Greedy Algorithms

- Main idea in greedy algorithms is to sort the shows in some “natural order”. Then consider the shows in this order and add a show to your list if it’s compatible with the shows already chosen.
- What’s a “natural order”?
  - Start Time: Consider shows in ascending order of $s_j$.
  - Finish Time: Consider shows in ascending order of $f_j$.
  - Shortest Time: Consider shows in ascending order of $f_j - s_j$.
  - Fewest Conflicts: Let $c_j$ be number of shows which overlap with show $j$. Consider shows in ascending order of $c_j$.
- Unfortunately, not all of these approaches are going to maximize the number of shows you could watch.
- But, we’ll show that considering the shows in order of the earliest finish time, maximizes the number of shows.

Ordering by Finish Time gives an optimal answer: Part 1

- To simplify the notation assume $f_1 < f_2 < f_3$...
- Suppose the earliest-finish-time-ordering approach picks shows $A = \{i_1, i_2, \ldots, i_k\}$ where $i_1 < i_2 < \ldots$.
- For the sake of contradiction suppose there’s a set of $k' > k$ compatible shows $B = \{j_1, j_2, j_3, \ldots, j_{k'}\}$ where $j_1 < j_2 < \ldots$.
- If there’s more than one subset of $k'$ compatible shows, pick the subset with $i_1 = j_1, \ldots, i_r = j_r$ for the max value of $r$.
- Note that $i_{r+1} \neq j_{r+1}$ and $k' \geq r + 1$ since the greedy algorithm could have picked show $j_{r+1}$ after show $i_r$.

Ordering by Finish Time is gives an optimal answer: Part 2

- But consider the schedule formed from $B$ by switching $i_{r+1}$ with $j_{r+1}$:
  $$C = \{j_1, j_2, \ldots, j_r, i_{r+1}, j_{r+2}, \ldots, j_{k'}\}$$
- $C$ is also compatible:
  - $i_{r+1}$ doesn’t overlap with $\{j_1, \ldots, j_r\}$
  - Because $i_{r+1}$ finishes before $j_{r+1}$, we know $i_{r+1}$ doesn’t overlap with $\{j_{r+1}, \ldots, j_{k'}\}$.
- But $C$ shares more than the first $r$ shows in common with $A$. This contradicts the assumption that $B$ was a subset of $k'$ compatible shows with the most initial shows in common with $A$.

Greedy Algorithms and Analysis

- Choose natural ordering, process items according to this ordering, avoiding conflicts as needed.
- How to choose the ordering?
  - Try to build counter-examples
  - Try to maintain some useful invariant
- Analysis: Today, greedy algorithm “stays ahead.”
  - Among all compatible sets $(i_1, \ldots, i_k)$ of size $k$, greedy guarantees $f_k$ as small as possible.

Problem 2: Interval Partitioning

- Suppose you are in charge of UMass classrooms.
- There are $n$ classes to be scheduled on a Monday where class $j$ starts at time $s_j$ and finishes at time $f_j$.
- Your goal is to schedule all the classes such that the minimum number of classrooms get used throughout the day. Obviously two classes that overlap can’t use the same room.

Possible Greedy Approaches

- Suppose the available classrooms are numbered 1, 2, 3, \ldots
- We could run a greedy algorithm… consider the lectures in some natural order, and assign the lecture to the classroom with the smallest numbered that is available.
- Continued next time...