CMPSCI 311: Introduction to Algorithms

Lecture 6: More Greedy Algorithms

Akshay Krishnamurthy

University of Massachusetts

Last Compiled: February 13, 2018

Announcements

- Quiz 3 due tonight!
- Homework 2 out (Due next wednesday 2/21)
- Homework 1 solutions posted
- Quiz 2 solutions

Recap: 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?)
- Answer: Order by finish time and choose shows greedly!
- Proof idea: Show that greedy "stays ahead" of other solutions.

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.
- Example: $[1, 4], [2, 3], [2, 7], [4, 7], [3, 6], [6, 10], [5, 7]$

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 number that is available.
- What's a "natural order" for this problem?
  - *Start Time*: Consider lectures in ascending order of s_j.
  - *Finish Time*: Consider lectures in ascending order of f_j.
  - *Shortest Time*: Consider lectures 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.
- Not all of these orderings will result in the best solution. But we'll show that ordering by start-time gives an optimal result.

Order by start time

- Number rooms $1, 2, \ldots,$
- Sort lectures by their start time s_j (assume $s_1 \leq s_2 \leq \ldots \leq s_n$)
- For $j = 1, \ldots, n$
  - Assign lecture j to available room with the smallest index.
  - For all occupied rooms r_t
    - If lecture i is in r_t and $f_i \leq s_{j+1}$, make r_t available.
- Running time: $O(n \log n)$. (But need to merge the f_js into the sorted list)

Ordering by Start Time gives an optimal answer

- A key observation:
  - Let the *depth* be the maximum number of lectures that are in progress at exactly the same time.
  - The number of class rooms needed by any schedule is $\geq$ depth.
- If d is the number of classrooms used by the greedy algorithm that considers classes in order of start time. We'll show $d \leq$ depth. Hence, $d =$ depth and there can't be a better schedule.
  - Suppose lecture j is the first lecture that the greedy algorithm assigns to classroom d.
  - At time s_j, there must be at least d lectures that are occurring. Hence, $d \leq$ depth.

Problem 3: Scheduling to Minimize Lateness

- Suppose an overworked UMass student has n different assignments due on the same day and each assignment has a deadline. Suppose that assignment j will take the student t_j minutes and has deadine d_j.
- If a student starts the assignment at s_j, she finishes the assignment at $f_j = s_j + t_j$ and let $\ell_j = \max\{0, f_j - d_j\}$ be the number of minutes she is late.
- Problem: In what order should she do the assignments if she wants to minimize the maximum lateness $L = \max_j \ell_j$.
- Example: $(t, d) = (5, 10), (1, 3), (3, 4), (2, 5)$

Possible Greedy Approaches

- We could do the assignments in order of:
  - *Shortest Time*: Consider in ascending order of t_j.
  - *Earliest Deadline*: Consider in ascending order of d_j.
  - *Smallest Slack*: Consider in ascending order of $d_j - t_j$.
- Not all of these orderings will result in the best solution. But we'll show that ordering by earliest deadline gives an optimal result.

Ordering by earliest deadline minimizes lateness: Part 1

- To simplify the notation assume $d_1 \leq d_2 \leq d_3 \leq \ldots$
- Given a schedule S, we say there's an *inversion* for jobs i and j if $d_i < d_j$ but job j is scheduled before i. The schedule generated by the greedy algorithm is the unique schedule in which there are no inversions.
- Some important observations:
  - There exists an optimal schedule with no idle time.
  - If there are any inversions in a schedule, there is an inversion involving two jobs that are scheduled consecutively.

Ordering by earliest deadline minimizes lateness: Part 2

- Claim: Given a schedule, swapping two adjacent, inverted jobs i and j (where $i < j$) reduces the number of inversions by one and does not increase the maximum lateness.
  - Let ℓ_k be the lateness of job k before the swap and let ℓ'_k be the lateness afterwards.
  - Note that $\ell'_k = \ell_k$ for all k other than $k \neq i$ and $k \neq j$.
  - Since i is finished earlier after the swap, $\ell'_i \leq \ell_i$
  - If job j is now late,
    $$\ell'_j = f'_j - d_j = f_i - d_j \leq f_i - d_i \leq \max\{0, f_i - d_i\} = \ell_i$$
  - Hence $\max\{\ell'_i, \ell'_j\} \leq \max\{\ell_i, \ell_j\}$
- Lemma: Ordering by the earliest deadline minimizes lateness.
  - Suppose there's a different schedule with inversions that has lateness L.
  - We can repeatedly use the above claim to transform it into a schedule with no inversions that has lateness at most L.

Summary

- Greedy algorithms for scheduling
  - Different "objectives" require different strategies
- On designing algorithms
  - Attack from both sides, try to build counter examples
- On proof strategies
  - Greedy "stays ahead"
  - Exchange arguments