CS310 – Advanced Data Structures and Algorithms

Class 14: Our first Online Class

Online Classes

- I’ll make slides, and videos to present them
- You can access them any time.
- Office hours are available by Zoom meetings—see links to click on our home page.
- Slide groups will be by topic, so some classes will have more than on slide set, and video for it.
- See class home page for announced online exam and practice exam exercise.
- Please let me know of your thoughts, concerns.

Today’s class and near future

- Today we have two slide sets:
  - Another greedy example
  - Intro to games, for project 3 and dynamic programming
- Thursday:
  - Finish up on games intro and algorithms
  - Next Tuesday: review for exam, practice exam submission
  - Next Thursday: online exam, no new slides

CS310 – Advanced Data Structures and Algorithms

Class 14: Another Greedy Example

Back to Algorithm Techniques

There are patterns in algorithms worth studying

We’re covering:

- Divide and conquer: we already saw examples
- Greedy algorithms: follow what appears to be best at each step, another example coming up
- Dynamic programming: save partial results as you go, then reuse them

Another Greedy Example: Interval scheduling

- Input: set of requests for periods of time (intervals) to use a special resource, like a supercomputer or a piano
- Task: Find maximum number of compatible time intervals out of this set. Two intervals are compatible if they don’t overlap.

Example (activities in each line are compatible):

```
[ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ]
```
Greedy, but how? (discussion from Wikipedia)

- Several algorithms, that may look promising at first sight, actually do not find the optimal solution:
  - Selecting the intervals that start earliest is not an optimal solution, because if the earliest interval happens to be very long, accepting it would make us reject many other shorter requests.
  - Selecting the shortest intervals or selecting intervals with the fewest conflicts is also not optimal.

The right Greediness...

- The following greedy algorithm does find the optimal solution:
  - Select the interval, \( x \), with the earliest finishing time.
  - Remove \( x \), and all intervals intersecting \( x \), from the set of candidate intervals.
  - Repeat until the set of candidate intervals is empty.

Why this greedy algorithm works (proof)

- Whenever we select an interval at step 1 (green one in figure), we may have to remove many intervals in step 2 (red).
- However, all these intervals necessarily cross the finishing time of \( x \), and thus they all cross each other (see figure).
- Hence, at most 1 of these intervals (green or red) can be in the optimal solution.
- Hence, for every interval in the optimal solution, there is an interval in the greedy solution. This proves that the greedy algorithm indeed finds an optimal solution.

Implementation for Interval Scheduling

- The straightforward algorithm is
  - Put all the intervals in a set \( S \) \( O(N) \)
  - Loop until done: \( O(N) \) passes
    - Find the interval \( I \) with earliest endtime \( O(N) \)
    - Drop \( I \) and the intervals that overlap with \( I \) from the set \( O(N) \)
- So \( T(N) = O(N^2) \)
- Can we do better?

Faster Implementation for Interval Scheduling (from K&T)

- Since the finishing time is crucial here, we sort the original set of intervals by end-time, at cost \( T(N) = N \log N \), well below the \( O(N^2) \) we're trying to beat. We can renumber them too, so the first-finishing appears first, etc.
- Then we iterate through the sequence:
  - Find the first interval, of end-time \( E \), and select it into result set
  - Iterate through following intervals, all of which have end-time \( \geq E \), skipping those with start-time \( \leq E \), until see one with start-time > \( E \), which we leave for the next pass.
- \( T(N) = O(N) \) for this second part, so \( T(N) = N \log N \) overall

K&T (Kleinberg & Tardos) Discussion

- Practical scheduling: we don't always know all the requests when we have to make a choice among the ones we have. This is "online" scheduling.
- The requests can have weights (profits, say), and we want to maximize the total weight for the selected requests.
  - This problem is known as Weighted Interval Scheduling.
  - No greedy algorithm is known for this problem.
- This is revisited in K&T Sec.6.1, where recursive search and dynamic programming are used, and the resulting algorithm is \( T(N) = O(N) \) after the input is sorted by end-time.