Olympic Scheduling

September 25, 2016

You are in charge of a live-streaming YouTube channel for the Olympics that promises never to interrupt an event. (So, once you start playing an event, you must play only that event from the time it starts to the time it finishes.) You have a list of the events, where each event includes its: **start time, finish time (which must be after its start time), and expected audience value**. Your goal is to **make a schedule to broadcast the most valuable complete events**. **The best schedule is the one with the highest-valued event**; **in case of ties, compare second-highest valued events, and so on**. (So, for example, you obviously **will** include the single highest-valued event in the Olympics—presumably the hockey gold medal game—no matter what else it blocks you from showing.)

(Times when you're not broadcasting events will be filled with "human interest stories" that have zero value; so, they're irrelevant.)

ASSUME: all event values are distinct and all event times are distinct. I.e., for any two values v_i and v_j with $i \neq j$, $v_i \neq v_j$. The same holds for start and end times (e.g., for any two start times s_i and s_j with $i \neq j$, $s_i \neq s_j$). Further, for any two start and finish times s_i and f_j, whether $i = j$ or not, $s_i \neq f_j$.

1 Naïve Algorithm

Consider the following algorithm. Assume that deleting an event from a list of events takes constant time.

```
Naive(E):
  result = new empty list of events
  while E is not empty:
    bestEvent = E[0]
    for each e in E:
      if value(e) > value(bestEvent):
        bestEvent = e
    delete bestEvent from E
    for each e in E:
      if start(e) < finish(bestEvent) and finish(e) > start(bestEvent):
        delete e from E
    add bestEvent to result
  return result
```

1.1 Finiteness

Briefly sketch a proof that the `while` loop in the algorithm above terminates. You need not give a formal proof, but you should include all key insights in the proof.

1.2 Efficiency

Give and briefly justify a good asymptotic bound on the runtime of the algorithm.

1.3 Correctness

Briefly sketch a proof that the algorithm is correct. You need not give a formal proof, but you should include all key insights in the proof.

2 Reduction on Simplified Problem

To make the Olympic Broadcasting problem simpler, we completely remove start time and finish times from the problem. So, now events only have values (not times), and a "schedule" is just a set of selected events. To make it slightly harder again, you are not allowed to select two events i and j if their values are within 10 units of each other: $|v_i - v_j| \leq 10$.

Give a correct reduction from this simplified Olympic Broadcasting problem to the sorting problem (where you provide both a list of items and a function to compare two items). Your reduction should take $O(n \lg n)$ time.

NOTE: You will likely find that (a) you can solve this with a single call to the sorting problem's solution algorithm and (b) producing the sorting instance is the easier part and transforming the solution to sorting into a solution to this simplified Olympic Broadcasting problem is the harder part. Don't forget to do both!

3 Olympic Reduction, BONUS ONLY

This was significantly harder than we intended it to be! So, we removed it from the quiz/assignment. It's a bonus problem worth two CPSC 320 bonus points for extremely clear, correct, and efficient responses. (Extremely clear reductions that take $O(n)$ time—not counting an $O(1)$ number of calls to a sorting algorithm—may receive 3 bonus points, but we don't know if such reductions are possible.)

Give a correct and efficient reduction from the Olympic broadcasting problem to the sorting problem (where you provide both a list of items and a function to compare two items). Your reduction—combined with an $O(n \lg n)$ sorting algorithm—should be asymptotically more efficient than the naïve algorithm above.

3.1 Correctness

Briefly sketch a proof that your algorithm is correct. You need not give a formal proof, but you should include all key insights in the proof.

3.2 Efficiency

Give and briefly justify a good asymptotic bound on the runtime of **just** your reduction, **not** including the call to the sorting algorithm. So, for the purposes of this asymptotic bound, you can imagine that we somehow solve sorting in constant time. (Note: it's possible to give a reduction that takes $O(n)$ time.)