15 Feb 2021

# Greedy Interval Scheduling

Announcements:

1. TA office hours start this week. See schedule on website. Click a calendar entry for Zoom link and Google Doc link. Use Google Doc to indicate what you'd like to ask about. Some office hours will be in person. Rooms have limited capacity; maintain safe distance while waiting in line.

2. Homework partner matching: if you requested homework partners you should have gotten email from Ruqi Zhang (rz297) last night.

   If you didn't get a partner yet, e.g. because you recently enrolled, email Ruqi. rz297@cornell.edu.

   If you aren't on CMS or Canvas yet, email me directly: robert.kleinberg@cornell.edu

# REDUCTIONS.

To reduce problem A to problem B means:

to use an algorithm for B as a step in running an algorithms to solve A.

A common template for reducing A to B is to design two algorithms

"Encoding": takes the input to A and transforms into an input for B

"Decoding": takes solution to B and transforms into a solution to A.

YOU ARE ALWAYS ALLOWED (ENCOURAGED!) TO SOLVE 4820 PROBLEMS BY REDUCING THEM TO ALGS THAT WERE TAUGHT IN

- LECTURES
- ASSIGNED READINGS
- PREREQUISITE COURSES
- ALREADY SOLVED HOMEWORK (same or previous assignment)

When you reduce A to B...

1. You can incorporate algorithm B as a single line of pseudocode.
2. Remember to account for the time it takes to run algorithm B.
3. In the correctness proof for A you are allowed to assume B is correct.

## Interval Scheduling

Given a set of $n$ time intervals ("jobs") $[s_i, f_i]$

$s_i$ = start time

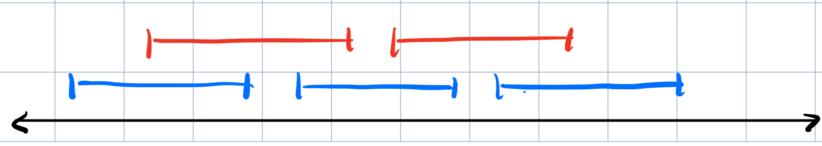
$f_i$ = finish time

$s_i \leq f_i \ \forall i$

find a non-conflicting subset with as many intervals as possible.

Def. $S$ is non-conflicting if $\forall i \neq j \in S$ either $f_i < s_j$ or $f_j < s_i$.

Example. Blue and red sets are non-conflicting. Blue set is optimal.



Examples of greedy algorithms

1. Shortest Job First:

   Pick the job that minimizes $f_i - s_i$.
   Add that to $S$ and remove all conflicting jobs.
   Recurse on remaining job set.

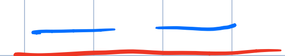
2. Earliest start time

3. Earliest finish time

   The only correct and efficient algo on this list.

4. Fewest conflicts

5. Latest finish time



6. Largest non-conflicting set of jobs.

   Correct, but exponential time.

Running time analysis of Earliest Finish Time:

1. Must sort all $n$ intervals by finish time.
   $O(n \log n)$

2. The remaining logic in EFT can be implemented in linear time.

$S = \{[s_1, f_1]\}$

$k = 1$ // $k$ = highest numbered job in $S$ so far.

for $i = 2, 3, \ldots, n$:

  if $s_i > f_k$: // $i$ doesn't conflict with any jobs in $S$

    $S = S \cup \{[s_i, f_i]\}$

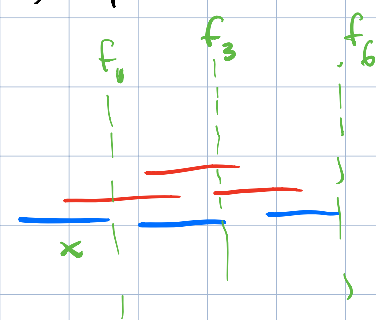
    $k = i$

  end if

endfor

output $S$

Adding an element to $S$ takes $O(1)$ if, for example, sets are implemented as linked lists.




Why is the algorithm correct? How can we prove its output is a maximum size non-conflicting set, for every possible input instance?

**Obs. 1.** $S$ is a non-conflicting set.

**Proof.** Induction on # loop iterations.

We only add an interval to $S$ if its start time is after latest finish time in $S$.

Proper proof would use the induct hyp:
after $m$ loop iterations, variable $k$ stores the index of the latest finishing job in $S$, <u>and</u> all jobs in $S$ are non-conflicting.

<u>Obs. 2.</u> Every interval in the input set contains one of the times

$$T = \{ f_i \mid \text{interval } i \text{ belongs to } S \}.$$

E.g. in the example above $T$ is a set of 3 time steps represented by 3 vertical dotted lines.

<u>Proof.</u> Intervals inserted into $S$ contain $f_i$.
Intervals that we skipped were of the form $[s_i, f_i]$ with $s_i \le f_k$ ← because we skipped $i$
and $f_i \ge f_k$. ← because we sorted by increasing finish time.

So that means $[s_i, f_i]$ contains $f_k$, which is in $T$.

Obs. 2 $\Longrightarrow$ size of $T$ is an upper bound on size of non-conflicting sets.
But $|S| = |T|$, so $|S|$ is as large as it could be.