CSC 373: Algorithm Design and Analysis
Lecture 1

Allan Borodin

January 7, 2013

Some materials are from Kevin Wayne's slides.

Lecture 1: Outline

- Introduction and motivation for

CSC 373: Design and Analysis of Algorithms

- Plus now in CSC373: A brief introduction to complexity theory
- Start of greedy algorithms; interval scheduling

The growing importance of TCS

- Its core questions (e.g. P vs NP) have gained prominence in both the intellectual and popular arenas.
- Recent breakthroughs in faster algorithms and scalable parallelizable data structures, complexity based cryptography, approximate combinatorial optimization, pseudo-randomness, coding theory,...
- TCS has expanded its frontiers.
  Many fields rely increasingly on the algorithms and abstractions of TCS, creating new areas of inquiry within theory and new fields at the boundaries between TCS and disciplines such as:
  - computational biology
  - algorithmic game theory
  - algorithmic aspects of social networks

Course Organization:

- **Lectures:** "Normally" M,W,F 11
- **Tutorials:** "Normally" M4 (but not today)
- **Exceptions:** When I have to swtich F11 lecture and M4 tutorial
- **Grading**:
  - 3 assignments at 5% each (**no late assignments**)
  - 3 term tests at 15% each (**similar to assignment)**
  - one final at 40%
- **Office hours**:
  - Scheduled hours TBA
  - by appointment
  - dropping in is also (usually) welcome
- **My office:** SF 2303B
- **My email:** bor@cs.toronto.edu

Materials

- The required text is Algorithms by dasGupta, Papadimitrou and Vazirani
- Many other excellent texts: Kleinberg and Tardos, Cormen et al
- Many existing course notes on the web
- My lecture slides are basically only outlines of the lectures and are *not* a substitute for a text. In particular, I will generally not include proofs in the posted slides.

The dividing line between efficient and NP hardnesss

- Many closely related problems are such that:

One problem has an efficient algorithm while a variant becomes (according to well accepted conjectures) difficult to compute (e.g. requiring exponential time complexity).

- For example (to be explained as we proceed):
  - Interval Scheduling vs Job Interval Scheduling
  - Minimum Spanning Tree (MST) vs Bounded degree MST
  - MST vs Steiner tree
  - Shortest paths vs Longest (simple) paths
  - 2-Colourability vs 3-Colourability

- Our focus is worst case analysis vs "peformance in practice"

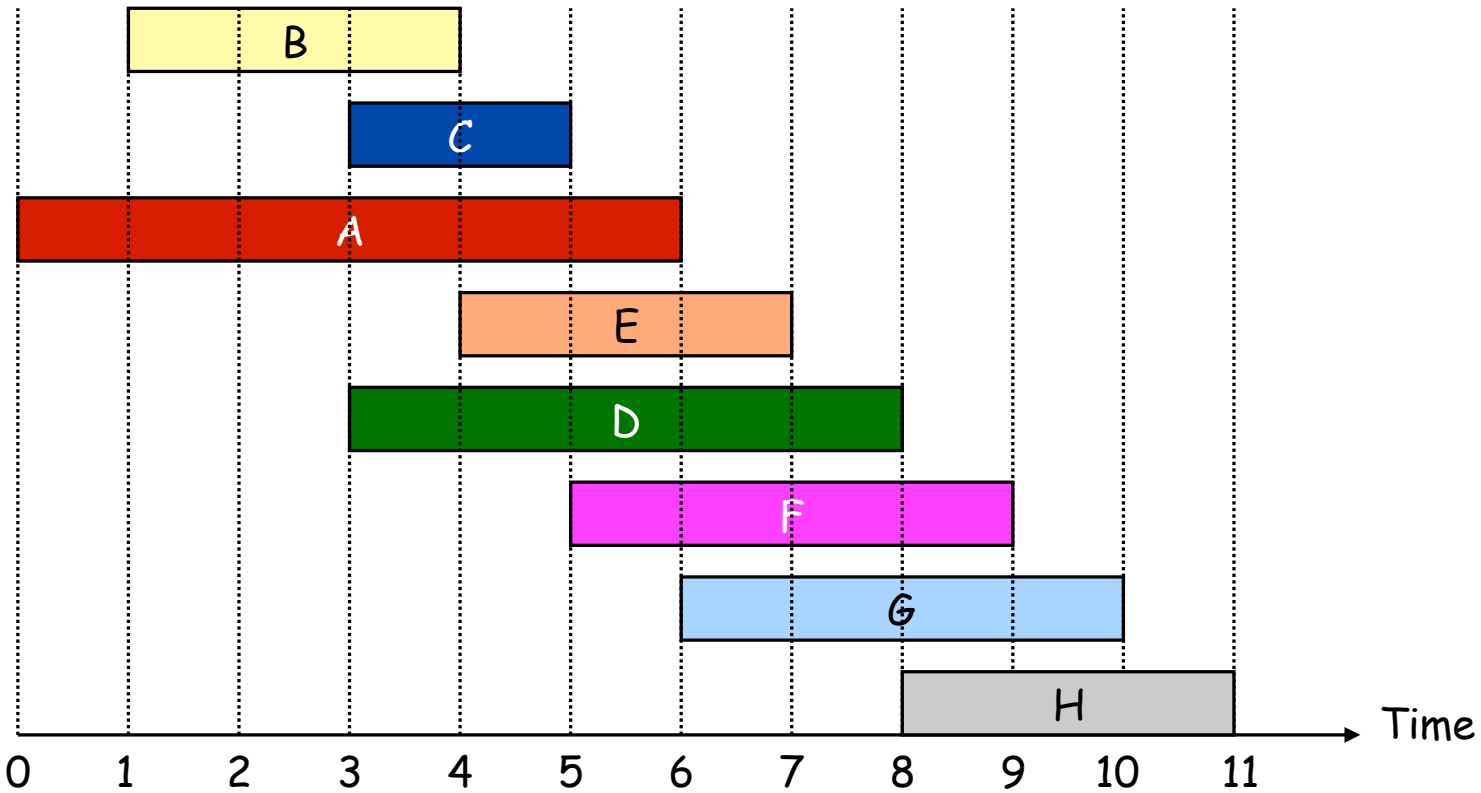
Tentative set of topics (very approximate)

- Easy vs Hard Problems (discussed throughout term)
- Greedy algorithms (5 Lectures)
- Dynamic Programming (5 Lectures)
- Network flows; matching (4 Lectures)
- NP and NP-completeness; self reduction (7 Lectures)
- Linear Programming; IP/LP rounding (5 Lectures)
- Local search (3 Lectures)
- Randomization (4 Lectures)
- Approximation algorithms (discussed throughout term)

Begin greedy algorithms

Interval Scheduling Problem

- Job j starts at s_j and finishes at f_j.
- Two jobs are compatible if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.

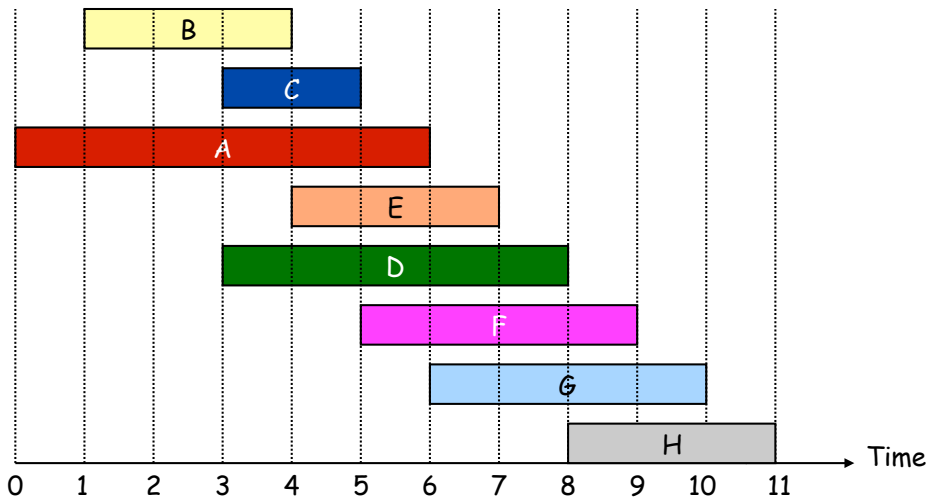


Interval Scheduling: Greedy Algorithm

Greedy template

- Consider jobs in some "natural" order.
- Take each job provided it's compatible with the ones already taken.

1. Earliest start time: Consider jobs in ascending order of s_j.
2. Earliest finish time: Consider jobs in ascending order of f_j.
3. Shortest interval: Consider jobs in ascending order of $f_j - s_j$.
4. Fewest conflicts: For each job j, count the remaining number of conflicting jobs c_j. Schedule in ascending order of c_j.



Interval Scheduling: Greedy Algorithm

Greedy template

- Consider jobs in some "natural" order.
- Take each job provided it's compatible with the ones already taken.

counterexample for earliest start time

counterexample for shortest interval

counterexample for fewest conflicts

A more general (greedy) myopic template

- Consider input items in some "reasonable" order.
- Consider each input item and make an irrevocable (greedy) decision regarding that input item.

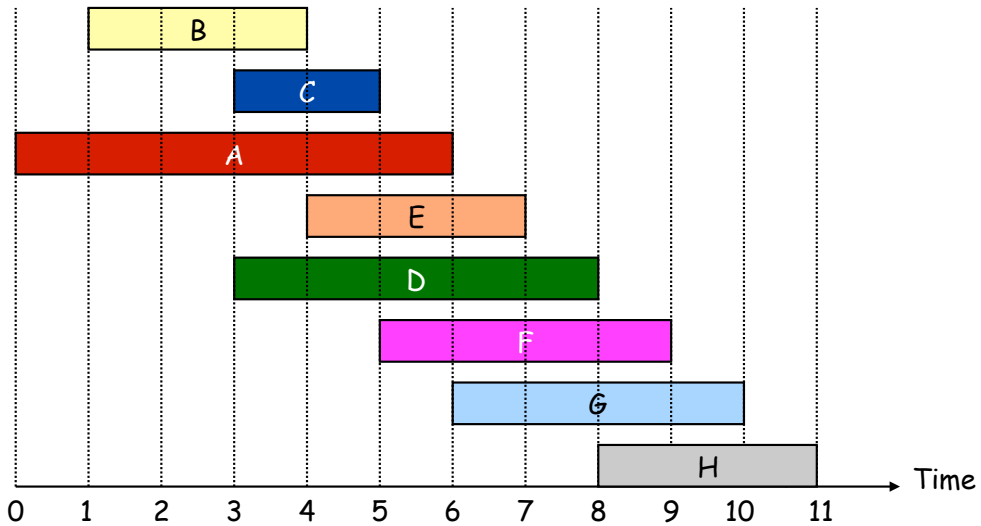
Terminology and templates

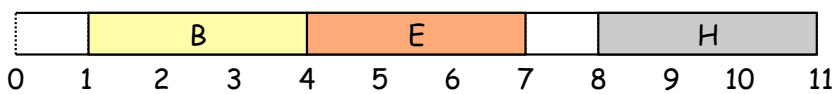
- We will follow the more common "greedy algorithms" terminology.
- However, as mentioned in text (and following older terminology), it would be better to use instead the suggestively broader class of myopic algorithms.
- My informal template for greedy algorithms differs somewhat from that given in the DPV text. I view the "greedy aspect" as it relates to the ("live for today") nature of the irrevocable decision rather than the choice of the next input item to consider.
- Neither the above template nor that in DPV is a precise definition.

Optimality of EFT Greedy algorithm

Earliest Finish Time (EFT) Algorithm

- Consider jobs in ascending order of finishing time f_j.
- Take each job provided it's compatible with the ones already taken.





Given the fact that some other reasonable greedy algorithms for the interval scheduling problem do not yield optimal solutions, how can we be convinced that EFT is optimal?

Comments on the optimality of EFT

- The proof outline shows that

The partial solution $S(i)$ at the end of the ith iteration is promising in that it can be extended to an optimal solution (using intervals not yet considered).

- This is not the only possible proof of this result. But before giving another type of proof (a charging argument), you might rightfully ask

"why bother proving this?"