CSC373: Lecture 1

Introduction and Motivation for "new CSC 373":
Design and Analysis of Algorithms
(with a brief introduction to complexity theory)

WHY CSC 373 remains a required course

The Computational Lens

- The world is undergoing a phase change, impelled by constant improvements in communication and computation. Think of the effect of cell phones on the politics of the Middle East, or the power of search engines on instant information accessibility, or the effect of DNA analysis on medicine. This phase change would not have been possible without advances in theoretical computer science (TCS).
- We are now at the dawn of an exciting era for TCS. Its core questions have gained prominence in both the intellectual and popular arenas. Recent years have witnessed breakthroughs in faster algorithms and scalable parallelizable data structures, complexity lower bounds, cryptography, approximate combinatorial optimization, pseudo-randomness, coding theory, … The field of TCS has expanded its frontiers. Its intellectual tools have turned to far broader applications than had once been envisioned. The mathematical and experimental sciences 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 the sciences. Computational biology and algorithmic game theory are two such examples.

Administration

- Lectures M,W,F 10-11; tutorials R 2-3 in rooms SS 2106, BA 3012, BA 3116
- Grading scheme: 3 assignments at 5% each (no late assignments), 3 term tests at 15% each, one final at 40%
- Office hours T 1:30-2:30 and W 11:30-12:30 and by appointment (dropping in also welcome); SF 2303B, bor@cs.toronto.edu
- Text: Kleinberg and Tardos: ignore at your peril

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:
- Interval Scheduling vs Job Interval Scheduling
- MST vs Bounded degree MST
- MST vs Steiner tree
- Shortest paths vs Longest (simple) paths
- 2 Colourability vs 3-Colourability

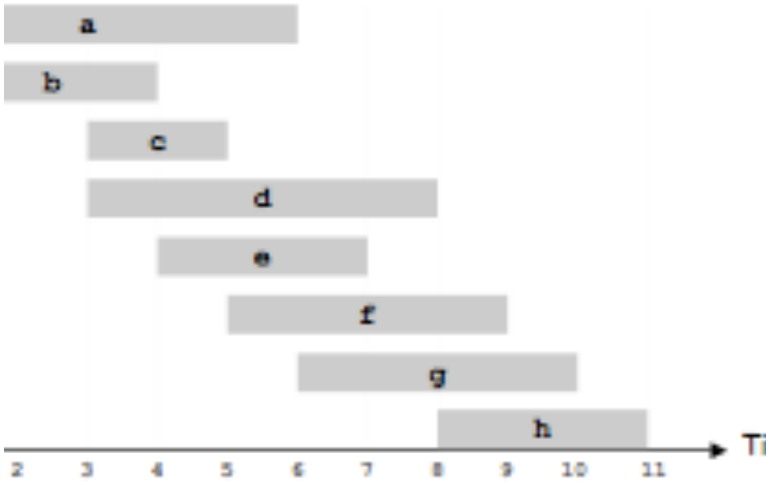
Tentative set of topics (very approximate)

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

Begin Greedy Algorithms

Interval Scheduling

uling.
: at s_j and finishes at f_j.
mpatible if they don't overlap.
aximum subset of mutually compatible jobs.



Interval Scheduling: Greedy Alg

Greedy template. Consider jobs in some natural o
Take each job provided it's compatible with the o

- [Earliest start time] Consider jobs in ascendir
- [Earliest finish time] Consider jobs in ascendi
- [Shortest interval] Consider jobs in ascending
- [Fewest conflicts] For each job j, count the n
  conflicting jobs c_j. Schedule in ascending orde

Interval Scheduling: Greedy Algorithms

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