Intro to Algorithms: their Limitations

Saiil Vadhan '95 (يهـ) 🌟

Head TF: Sanket Shukla

Agenda
- Overview of learning objectives + context
- Course operation
- Q & A

Motivation
Algorithms solve problems on a massive scale, e.g.,
  web search, Google maps, online dating

How? fast, hardware? application-specific engineering?

CS120: efficient, general-purpose algorithm design

- Applies to many problems
- Independent of technology/hardware
- Mathematical abstraction

Common algorithmic techniques:
- Divide & conquer
- Greedy
- Dynamic programming

Analysis:
- Correctness
- Time
- Space
- Proofs

Examples:
- Graph coloring
- Shortest path

Reducing:
- Transform solutions from one problem to another
Algorithmic Problem Areas We'll Study

- Sorting, sorting, searching, datasets (6 classes)
  - ordered vs. unordered data
  - static vs. dynamic data
  - deterministic vs. randomized algos
    - ordered: natural scheduling
    - static: balanced binary search trees
      - collection of the window functions
      - res 1 res 2 res 3 res 4
        - res 5
          - res 6
  - matching - "independent" set of edges
    - find a edge-free (conflict-free)
      - set of nodes as large as possible

- Graph algorithms (n+4 classes)
  - graph search - BFS, DFS, random walks
  - graph coloring & independent sets
  - matching - "independent" set of edges
    - set of nodes

- Geometric algorithms (2 classes if time permits)
  - apps: graphics, robotics, ML
    - hyperplane separation (CP, SVM)
    - minimum spanning tree

- Logic algorithms
  - propositional satisfiability (SAT/CDP/SMT solvers)
  - apps: program verification

Limits of Algorithms

- Why don't we have:
  - perfect software debugging?
  - fast algorithms for under scheduling problems?
  - fast algorithms for predicting behavior of natural systems?

- Intractability: probably no polynomial-time algorithm
  - uncomputability: probably no general algorithm
    - uncomputability: uncomputable
    - intractability: probably no polynomial-time algorithm
      - reasoning: P ≠ NP

(Extended) Church-Turing Thesis: these are technology-independent limitations
Learning Outcomes

- mathematical abstraction
- design and implement algorithms
- recognize formalize limits
- rigorously analyze
- appreciate the theory

Course Overview ➔ Syllabus