CSC373 Notes

Max Xu

January 9, 2026

Contents

1 Day 1: Intro (Jan 06, 2026) **2**

2 Day 2: Intro Redux (Jan 08, 2026) **3**

§1 Day 1: Intro (Jan 06, 2026)

This was taken off the slides from past years.

1.1 About this Class

This class is about designing algorithms to solve problems.

We will be:

- Designing fast algorithms
  - Divide and conquer
  - Greedy algorithms
  - Dynamic programming
  - Network flow
  - Linear programming
- Proving no fast algorithms are likely possible
  - Reductions and NP-completeness
- What to do if no fast algorithms are possible
  - Approximation algorithms
  - Randomized algorithms

When we analyze an argument, we do correctness and running-time proofs.

§2 Day 2: Intro Redux (Jan 08, 2026)

This course is now about the thought process behind solutions of problems. We use the **RAM Computational Model**.

A proof is a convincing argument:

- Convince your TA for marks
- Convince employer that your program does what it claim it does

Sometimes, formal verification is used for mission-critical applications, where unit tests may not have sufficient coverage. We use semi-formal proofs in this course, to prove specific results (as opposed to more general ones, like in math).

2.1 Divide & Conquer

The general framework is to

- Break a problem into two smaller subproblems of the same type
- Solve each problem recursively and independently
- Quickly combine solutions from subproblems to form a solution to a bigger part of the problem

'Quick/cheap' means that the step count is in $O(f(n))$ where f is a polynomial.

Recurrence relations are often encountered while analyzing the running time of divide-and-conquer algorithms. We take the master theorem from (CLRS) for granted, a general result about the asymptotic behavior of certain types of recurrences.

The master theorem branches: branching dominated, balance, merging dominated.

Example 2.1

Algorithms considered divide-and-conquer include:
- Closest pair in $\mathbb{R}^2$, with non-degeneracy assumption
- Karatsuba's Algorithm
- Strassen's Algorithm