§1 Introduction

Qin Zhang
Brief self-introduction:

My name: Qin[Chin] Zhang

I’ve worked on theoretical CS for 10+ years;

My main interest is **Algorithms for Big Data**, in particular: *data stream algorithms, sublinear algorithms, algorithms on distributed data, data structures*

I’ve published extensively in all top conferences/journals in theoretical CS

I write experimental papers too, and have published in all top databases, data mining and machine learning venues
Today’s plan

1. An introduction of the course

2. A $\approx 45$-min touch base exam
Why study algorithms?

Algorithms are used everywhere, any time

**Internet.** Web search, packet routing ...

**Biology.** DNA similarity search, protein folding ...

**Multimedia.** MP3, JPG, face recognition ...

**Social networks.** Recommendations, advertisements ...

**Daily life helpers:** Google maps, auto translation ...

This course tries to introduce some basic concepts, techniques and tools for algorithm design. These may serve as building blocks for solving real-world problems.
We want to *design* algorithms that are *time*, *space* and *communication* efficient
C343 vs B403

- **C343:** Implementation and make use of classic data structures and algorithms.

```java
private static void sort(double[] a, int lo, int hi)
{
    if (hi <= lo) return;
    int lt = lo, gt = hi;
    int i = lo;
    while (i <= gt)
    {
        if (a[i] < a[lo]) exch(a, lt++, i+++);
        else if (a[i] > a[lo]) exch(a, i, gt--);
        else i++;
    }
    sort(a, lo, lt - 1);
    sort(a, gt + 1, hi);
}
```

- **B403:** Design and analysis of algorithms.

\[
\sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{2}{j - i + 1} = 2 \sum_{i=1}^{N} \sum_{j=2}^{N-i} \frac{1}{j} \\
\leq 2N \sum_{j=1}^{N} \frac{1}{j} \\
\sim 2N \int_{x=1}^{N} \frac{1}{x} \, dx \\
= 2N \ln N
\]
Course schedule

0 : Introduction
   – Big-O notations, common running times

1 : Graph
   – BFS, DFS, DAG, topological sorting

2 : Greedy Algorithms
   – Interval scheduling, MST, shortest path

3 : Divide & Conquer
   – Mergesort, counting inversions, closest pair

4 : Dynamic Programming
   – Weighted interval scheduling, subset-sum, edit distance

5 : NP and Intractability
   – Polynomial reduction, NP-completeness, hard problems
Textbooks

- Required textbook
  - **Algorithm Design**
    by J. Kleinberg and E. Tardos
    Pearson Education

  The book comes with slides:
  http://www.cs.princeton.edu/~wayne/kleinberg-tardos/
  (or Google “Algorithm Design slides”)

- Supplementary Textbook
  - **Introduction to algorithms**
    by T. Cormen, C. Leiserson, R. Rivest, C. Stein
    3rd edition. MIT
Resources

- **Course website**
  
  http://homes.sice.indiana.edu/qzhangcs/B403-20-spring-algorithm/
  
  - Various information (e.g., office hours, exam dates)

- **Canvas**
  
  1. Homework assignments and solutions (submission in class)
  2. Announcements
  3. **Course lecture notes**
Instructors

- Instructor: Qin Zhang
  Email: qzhangcs@indiana.edu
  Office hours: Wed. 4-5pm @ Luddy 3128

- Als:
  - Mohsen Sayyadiharikandeh
  - Harshad Badiyani
  - ??

Helper (not official AI):
- Yan Song

Emails, office hours: posted on course website
Grading

• Assignments 30%: 6 written assignments
  Due before class on the due date, in hard copy.
  If you can, typeset in your favorite software and
  bring printed hard copy to class.
  If you are handwriting, make sure it is legible.
  No extensions or late homeworks will be granted.

• Exams 70%: Mid-term 1 (20%), Mid-term 2 (20%),
  and Final (30%).

• The final grades will be curved.
More practice

**Practice** is very important to master algorithm design.

1. Subsections in the textbook that we do not cover in class
2. Solved exercises in the textbook
3. Other exercises in the textbook (do not appear in homeworks). Feel free to ask us questions if you meet any difficulty (email us the question number first so that we can get prepared).
4. Any other questions that you can find online – there are tons of algorithm design questions online. Again, feel free to ask us questions if you meet any difficulty.
Participants must have a background in **discrete math** and **data structures**, and have taken

1. C241 Discrete Structures for Computer Science
2. C343 Data Structures
   
   https://iu.instructure.com/courses/1560867/pages/schedule?module_item_id=14976212

3. MATH-M 216 ”Analytic Geometry and Calculus II”
   (or MATH-M 212 CALCULUS II)
Thank you!

Questions?