Administrative Matters

- **Time/location:** 1CD3A@SC204
- **Instructor:** Iris Hui-Ru Jiang
  - Email: huiru.jiang@gmail.com
  - Office: ED540, ext. 31211
  - Office hours: 1X (made by appointment)
- **Teaching assistants:** 鄭安哲/李承睿
  - Email: nctuee.alg@gmail.com
  - Lab: ED413, ext. 54226
  - Office hours: 2EF
- **Prerequisite:** two out of the following courses
  - Data structures
  - Discrete mathematics
  - Computer programming in C
  - Computer programming in C++

Course Objectives

- **Study unifying principles and concepts of algorithm design**
  - Algorithmic problems form the heart of computer science
- **Polish your critical thinking and problem-solving technique**
  - Algorithmic problems tend to come bundled together with lots of messy, application-specific detail, some of it essential, some of it extraneous
  - Two fundamental components
    - Get to the mathematically clean core of a problem
    - Identify the appropriate algorithm design techniques based on the structure of the problem
- **Intended audience:**
  - Who are interested in computer science
  - Who are computing something
  - Who are learning problem-solving techniques

Reading Materials

- **Course webpage:**
  - e3, NCTU open course ware
- **Required text:**
  - Jon Kleinberg, 20 Best Brains under 40, Discover Magazine, 2008
  - Cornell
- **Reference:**
  - UC Berkeley
  - Bible! MIT
### Course Content (1/3)

#### Introduction
- An opening problem: stable marriages
- Range of problems we will consider

#### Background:
- Basics of algorithm analysis
- Graphs

#### General algorithmic techniques
- Greedy algorithms
  - Finding optimal solutions with greedy methods
  - Scheduling time intervals
  - The minimum spanning tree problem
- The divide and conquer method
  - Some basic primitives in computational geometry

### Course Content (2/3)

#### Dynamic programming with many applications
- Weighted interval scheduling
- Knapsack problems
- Shortest paths
- Sequence alignment
  - Including efficient implementation via divide and conquer
- Flows and cuts in networks
  - The basic flow and cut problems
  - Basic methods: augmenting paths
  - Application to matching
  - Polynomial time methods
  - Extensions to more general models
  - Applications to resource allocation, sequencing, and segmentation

### Course Content (3/3)

#### Computational Intractability (Optional)
- NP-completeness
  - Hardness of problems in optimization and constraint satisfaction
  - How to show NP-completeness: reducibility
  - Examples including the traveling salesman problem, 3-dimensional matching, covering, packing, partitioning problems, and subset sum
- PSPACE completeness
  - Hardness of problems in artificial intelligence and game-playing

### Optional Course Content

#### Algorithms for hard problems
- Improved exponential methods
- Approximation algorithms
  - Greedy algorithms (load balancing, facility location)
  - The application of linear programming
- Local search techniques
  - The Metropolis Algorithm
  - Simulated annealing
  - Applications to graph partitioning and neural networks

#### Randomized algorithms
- Including contention-resolution protocols and satisfiability heuristics
- Algorithms that run forever
Grading Policy

- **Grading:**
  - Homework: 15%
  - Programming: (two mini-projects: 10% + project: 25%)
  - Exams: (Midterm: 25% + Final: 25%)

- **Homework & programming:**
  - Discussions are encouraged. However, solutions should be written individually and separately (no credits for plagiarism)
  - **No late submission** (partial solutions may get partial credits)
  - Homework is usually due in two weeks
  - Programming assignments are usually due in two or three weeks

- **Exams**
  - No discussions, solutions should be written individually

What is an Algorithm?

- **Definition:** An **algorithm** is
  - A finite, definite, effective procedure, with some output
  - **Input:** may have problem
  - **Output:** must have solution
  - **Definiteness:** must be clear and unambiguous
  - **Finiteness:** terminate after a finite number of steps
  - **Effectiveness:** must be basic and feasible with pencil and paper
  - **Procedure:** the sequence of specific steps in a logical order

- **Cf. An algorithm is**
  - A well-defined procedure for transforming some input to a desired output [Cormen et al. Introduction to Algorithms, 2nd Ed.]