Introduction to Algorithms

Zhenlin Wang

About this course

- Text:
  - Introduction to Algorithms by T. H. Cormen at el., 2nd edition
- We meet TR 2:05 PM – 3:20 PM at Fisher 325
- Office hours: MW 1:30 PM – 3 PM

Course administration

- Homework
  - 6-7 assignments, 25%
  - Encouraged to form discussion groups, but you need to finish your work independently
- One in-class midterm (around Oct. 23), 25%
- Final (comprehensive), 35%
- Class participation & Quizzes 15%
  - Five to six quizzes
  - 50% participation credits

Recommendations

- This is a challenging course!!
  - My job is to make it easier and more enjoyable
  - Your job is to work hard
- Come to class
- Web page
  - Homework, examples, hints
- See your instructor for help
  - You are welcome when my door is open (almost always)
  - Topics not limited to this course

Major topics

- Algorithm design and analysis
  - Efficiency: worst case, best case, and average, amortized analysis
- Asymptotic notations
  - O, Ω, o, ω
- Data structures
  - Heap, binomial heap, disjoint sets, trees, graphs
- Design Paradigms
  - Greedy, divide-and-conquer, dynamic programming
- P and NP problems
  - Reduction, complexity

Topics and Exams

Topics	Reading
Introduction	1
Induction and loop invariants	2.1
Asymptotic Notation	3.1-3.2
Algorithm Analysis - Analyzing control structures - Worst-case and Average-case - Amortized analysis	2.2 5.1-5.3 17.1-17.3
Solving Recurrences	4.1-4.3
Heap and Heap Sort	6
Binomial Heaps	19

Topics and Exams

Topics	Reading
Splay Trees	Notes, handouts
Disjoint Set	21.1-21.3
Greedy Algorithms - Minimum Spanning Tree - Dijkstra's algorithm - Knapsack - Scheduling	16.1-16.2 16.5 23 24.3
In-class mid term (around Oct. 23)	
Divide-and-Conquer - Mergesort and Quicksort - Median - Closest pair	2.3 7.1-7.4 9

Topics and Exams

Topics	Reading
Dynamic Programming - Assembly line scheduling - Longest common subsequence - Floyd's algorithm - Matrix chain	15.1-15.4 25.1-25.2
Exploring Graphs - Graph Search - Topological sorting	22.1-22.4
Network Flow and Matching	26.1-26.3
Decision Trees and Low Bound Arguments	8.1
P and NP Problems	34-35
Final	

Algorithmics, Data Structures and Discrete Structures

- Algorithms
  - Thought and presented according to design techniques rather than by application domain
  - Focus on efficiency, complexity, and correctness
- Data structures
  - Focus on data organizations accompanied by related algorithms
- Discrete structures
  - Mathematical properties of data structures

Why this course?

- Algorithm
  - A sequence of computational steps that transform the input to the output
  - For example, sorting
    - Input: an array of n elements
    - Output: an sorted array that is a permutation of the input
- Algorithmics
  - The systematic study of the design and analysis of algorithms
- Our goal
  - Given a problem, be able to design an efficient algorithm (be able to analyze your algorithm) and prove the algorithm works

Pitfall: machine is so cheap, so the efficiency of an algorithm is not important

- Moore's law
  - Processor's speed doubles every 18 months
  - A typical 2006 machine is 64 times faster that one in 2000

2000	512 MHZ
2003	2-3GHZ
2006	Multicore 2GHZ * 4

Example: Fibonacci Sequence

- Definition

$$f_n = \begin{cases} n, & n = 0,1 \\ f_{n-1} + f_{n-2}, & n \geq 2 \end{cases}$$

- Algorithms to generate the n^{th} Fibonacci number
  - Recursive algorithm
  - Iterative algorithm

Iterative vs. Recursive Algorithm

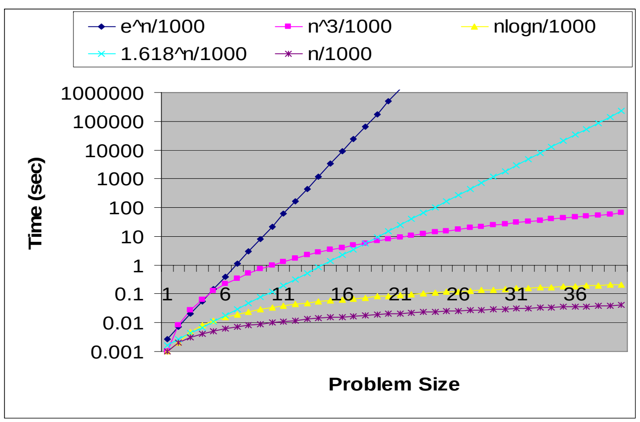
```
double fibIterative(int n)
{
  double Fn_1, Fn_2, Fn;
  int i;

  if (n<2)
    return ((double)n);

  Fn_2 = 0;
  Fn_1 = 1;
  for (i=2; i<=n ; i++) {
     Fn = Fn_1 + Fn_2;
     Fn_2 = Fn_1;
     Fn_1 = Fn;
  }
  return Fn;
}
```

```
double fibRecursive(int n)
{
  double ret;

  if (n<2)
    ret = (double)n;
  else
    ret =  fibRecursive(n-1) +
           fibRecursive(n-2);

  return ret;
}
```

Experimental comparison

n	Recursive	Iterative	Recursive/ Iterative
4	0.27 ms	0.03 ms	9
8	0.71 ms	0.05 ms	14
16	27 ms	0.14 ms	193
32	0.28 sec	0.32 ms	875
64	> 3300 mins	0.64 ms	> $3*10^{11}$

Execution time versus problem size



Fibonacci Sequence Algorithms

- Design
  - We saw two implementations (top-down/bottom-up methods are used here).
- Correctness
  - How can we prove the two algorithms compute f_n?
- Efficiency
  - How can we show that the execution time of the iterative algorithm is proportional to *n* while the recursive algorithm to $(\frac{1+\sqrt{5}}{2})^n$, when *n* is sufficiently large?