CS483 Design and Analysis of Algorithms

Lecture 1 Introduction and Prologue

Instructor: Fei Li

lifei@cs.gmu.edu with subject: CS483

Office hours:
Room 5326, Engineering Building, Thursday 4:30pm - 6:30pm or by appointments

Course web-site:
http://www.cs.gmu.edu/~lifei/teaching/cs483_fall11

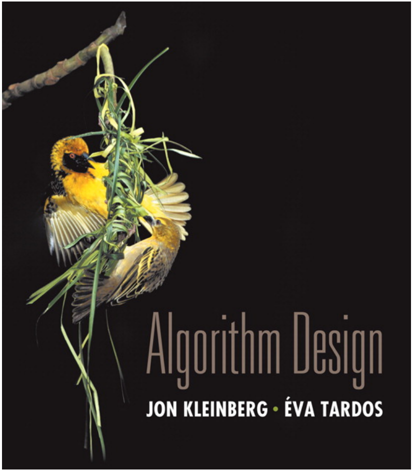
Figures unclaimed are from the textbook "Algorithm Design".

About this Course

- About this Course
  (From 2007-2008 University Catalog) Analyze computational resources for important problem types by alternative algorithms and their associated data structures, using mathematically rigorous techniques. Specific algorithms analyzed and improved
- Prerequisites
  CS310 (Data Structures) and CS330 (Formal Methods and Models) and MATH125 (Discrete Mathematics I)
- Weekly Schedule
  - When: Tuesday & Thursday 3:00pm - 4:15pm
  - Where: Krug Hall 242

Required Textbooks

1. **Algorithm Design** by Jon Kleinberg and Eva Tardos



How to Reach Me and the TA

1. Instructor: Fei Li
2. Email: lifei@cs.gmu.edu
3. Office: Room 5326, Engineering Building
4. Office hours: Thursday 4:30pm - 6:30pm or by appointments

1. Teaching Assistant: Chen Liang
2. Email: cliang1@gmu.edu
3. Office: Room 4456, Engineering Building
4. Office hours: Wednesday 11:00am - 1:00pm

Making the Grades

1. Your grade will be determined 45% by the take-home assignments, 20% by a midterm exam, and 35% by a final exam
2. Tentatively, there will be 9 assignments; each assignment deserves 5 points
3. Hand in hard copies of assignments in class. No grace days for late assignment. All course work is to be done independently. Plagiarizing the homework will be penalized by maximum negative credit and cheating on the exam will earn you an F in the course
4. Tentative grading system:
   A (≥ 85), B ($\in [70, 85)$), C ($\in [60, 70)$), D ($\in [50, 60)$), and F (< 50)

Any Questions?

What Are We Going to Learn from this Course?

Goal. Given n men and n women, find a "suitable" matching.

- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

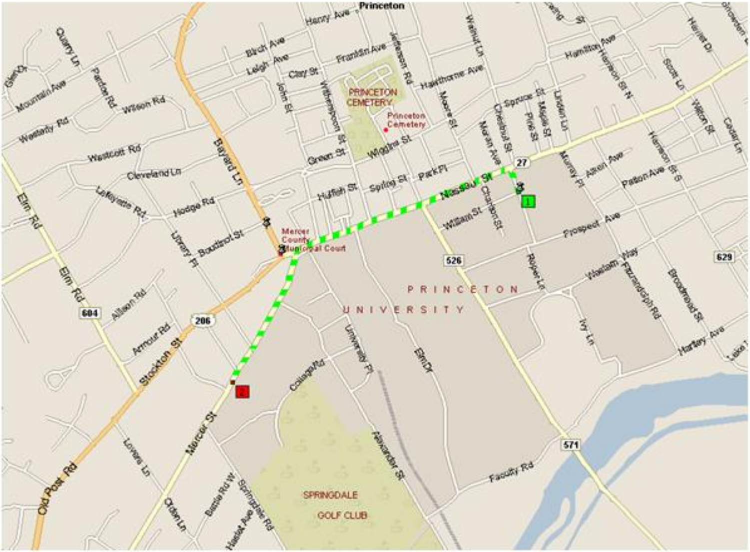
	favorite ↓ 1st	2nd	least favorite ↓ 3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	favorite ↓ 1st	2nd	least favorite ↓ 3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

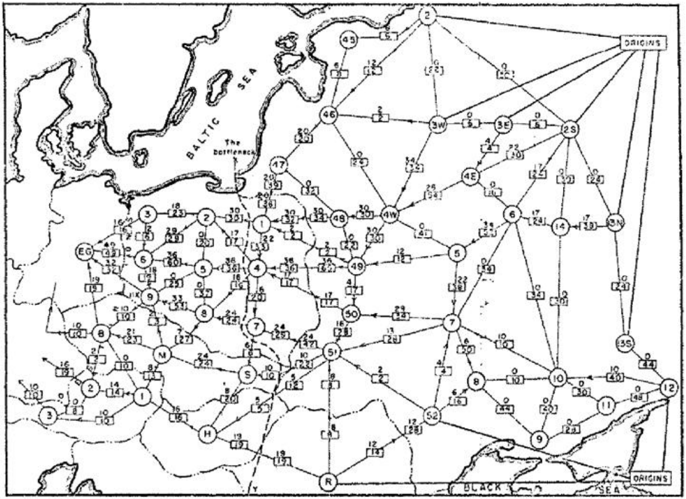
What Are We Going to Learn in this Course?



shortest path from Princeton CS department to Einstein's house

What Are We Going to Learn in this Course?

Soviet Rail Network, 1955

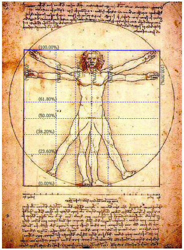


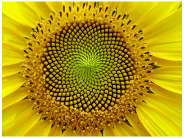
Reference: *On the history of the transportation and maximum flow problems.*

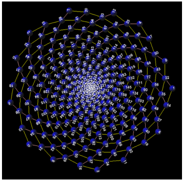
The Necessity and Benefits of Learning Algorithms

1. Algorithm example — **calculate Fibonacci Numbers**
2. Running time — asymptotic notation

Fibonacci Series and Numbers







$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ \ldots,$$

The Fibonacci numbers F_n is generated by

$$F_n = \begin{cases} F_{n-1} + F_{n-2}, & \text{if } n > 1, \\ 1, & \text{if } n = 1, \\ 0, & \text{if } n = 0. \end{cases}$$

The golden ratio $\phi = \frac{1+\sqrt{5}}{2} = 1 + \frac{1}{\phi} \approx 1.618 = \lim_{n\to\infty} \frac{F_{n+1}}{F_n}$

Calculate F_n — First Approach

From the recursive definition

```
function fib1(n)
{
    if (n = 0)
        return 0;
    if (n = 1)
        return 1;

    return fib1(n - 1) + fib1(n - 2);
}
```

Calculate F_n — First Approach

From the recursive definition

```
function fib1(n)
{
    if (n = 0)
        return 0;
    if (n = 1)
        return 1;

    return fib1(n - 1) + fib1(n - 2);
}
```

- Correctness

Calculate F_n — First Approach

From the recursive definition

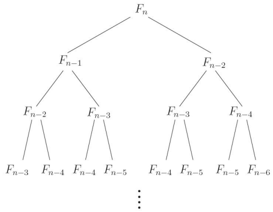
```
function fib1(n)
{
    if (n = 0)
        return 0;
    if (n = 1)
        return 1;

    return fib1(n - 1) + fib1(n - 2);
}
```

- Correctness
- Running time $T(n) = T(n-1) + T(n-2) + 3,\ n > 1$

$$T(200) \geq F_{200} \geq 2^{138}$$

Calculate F_n — Second Approach

Figure 0.1 The proliferation of recursive calls in fib1.



```
function fib2(n)
{
    if (n = 0)
        return 0;

    create an array f[0, ..., n];

    f[0] = 0; f[1] = 1;

    for (i = 2, ..., n)
        f[i] = f[i - 1] + f[i - 2];

    return f[n];
}
```

fib2(n) is linear in n.

Why Does it Matter?

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Course Outcomes

1. An understanding of classical problems in Computer Science
2. An understanding of classical algorithm design and analysis strategies
3. An ability to analyze the computability of a problem
4. Be able to design and analyze new algorithms to solve a computational problem
5. An ability to reason algorithmically