CS483 Design and Analysis of Algorithms

Lecture 1 Introduction and Prologue

---

1Course web-site:
http://www.cs.gmu.edu/~lifei/teaching/cs483_fall12
Figures unclaimed are from the textbook “Algorithm Design”.
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: Fridays 10:30am - 1:10pm
- Where: Robinson Hall B208
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: Fridays 2:00pm – 4:00pm or by appointments

1. Teaching Assistant: Ermo Wei
2. Email: ewei@masonlive.gmu.edu
3. Office: Room 4456, Engineering Building
4. Office hours: Tuesdays 1:00pm – 3:00pm
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 ~ 12 assignments; each assignment deserves 4 ~ 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 ($\geq 85$), 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.

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
<tr>
<td>Yancey</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
</tr>
<tr>
<td>Zeus</td>
<td>Amy</td>
<td>Bertha</td>
<td>Clare</td>
</tr>
</tbody>
</table>

*Men’s Preference Profile*

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Zeus</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
</tr>
</tbody>
</table>

*Women’s Preference Profile*
What Are We Going to Learn in this Course?
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, ..., 

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

```c
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

```cpp
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

```java
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

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 $n$. 

Figure 0.1 The proliferation of recursive calls in fib1.
Why Does it Matter?

<table>
<thead>
<tr>
<th>n</th>
<th>n</th>
<th>n log₂ n</th>
<th>n²</th>
<th>n³</th>
<th>1.5ⁿ</th>
<th>2ⁿ</th>
<th>n!</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>4 sec</td>
</tr>
<tr>
<td>n = 30</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>18 min</td>
<td>10²⁵ years</td>
</tr>
<tr>
<td>n = 50</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>11 min</td>
<td>36 years</td>
<td>very long</td>
</tr>
<tr>
<td>n = 100</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>1 sec</td>
<td>12,892 years</td>
<td>10¹⁷ years</td>
<td>very long</td>
</tr>
<tr>
<td>n = 1,000</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>1 sec</td>
<td>18 min</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>n = 10,000</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>2 min</td>
<td>12 days</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>n = 100,000</td>
<td>&lt; 1 sec</td>
<td>2 sec</td>
<td>3 hours</td>
<td>32 years</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>n = 1,000,000</td>
<td>1 sec</td>
<td>20 sec</td>
<td>12 days</td>
<td>31,710 years</td>
<td>very long</td>
<td>very long</td>
<td>very long</td>
</tr>
</tbody>
</table>

*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 10²⁵ years, we simply record the algorithm as taking a very long time.
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