Introduction

- Questionnaire
- Prayer
- Intro. Appointments: send me email

Objectives for Today

- Introduce course objectives
- Quickly cover course info.
- Define Problems, Solutions, and Algorithms
- Introduce 3 Questions
- Motivate the need for Analysis
- Begin to understand algorithm Efficiency

High-Level Course Objectives

- Algorithm Design and Analysis
  - Develop your general problem solving skills!
  - Become capable with several families of algorithms suitable for solving many kinds of problems
  - Learn to perform theoretical analysis of algorithms
  - Learn to perform empirical analysis of algorithms
  - Compare theoretical and empirical analysis
- Design and implement several useful algorithms
  - Use Visual Studio and C#

Course Info.

- Public web site: http://wiki.cs.byu.edu/cs-312/start
  - Syllabus – read today if you haven’t done so already
  - Quick quiz about syllabus on Wednesday
  - Projects
  - Homework
- Private web site: http://learningsuite.byu.edu/
- Office hours: See course website
  - Always available by appointment
- Email: ringger@cs.byu.edu
- TA Email: byu312ta@gmail.com
More Course Info.

- Learning Suite:
  - Schedule
  - Grades
  - Project report submission
  - Homework Keys
  - Quizzes

- Announcements:
  - Google Group: "byu-ca-312-ringger-announce"
  - For announcements
  - Mandatory

- Forum for discussion:
  - Google Group: "byu-ca-312-ringger"
  - Highly recommended

Assignments

- 6 projects
- Daily reading and homework
- About 1x per week:
  - Screencasts with quizzes
  - Flip homework and lecture

Course Policies

- Grades
- Early
- Late
- Other

See syllabus for details.

Another Thought

“Computer Science is no more about computers than Astronomy is about telescopes.”

-- Michael R. Fellows and Ian Parberry *

* often misattributed to Edsger Dijkstra

Definitions

- What is a problem?
- What is a solution?

Problems and Their Solutions
The Solution is the Algorithm

Properties of an Algorithm
- What is an algorithm?
  - "A finite sequence of well-defined steps for solving a problem."
- Required Properties:
  - Finite number of steps
  - Steps consist of Elementary Operations
  - On a given Computation Device
  - Terminate
- Properties:
  - Be correct
    - At times: willing to approximate
  - Be deterministic
    - At times: willing to allow randomness / stochasticity
- Desired properties:
  - Efficiency, both space and time
  - Structure, ease of interpretability and maintainability

Teaching a Computer

"A person well trained in computer science knows how to deal with algorithms: how to construct them, manipulate them, understand them, analyze them. This knowledge is preparation for much more than writing good computer programs; it is a general-purpose mental tool that will be a definite aid to the understanding of other subjects, whether they be chemistry, linguistics, or music, etc. The reason for this may be understood in the following way: It has often been said that a person does not really understand something until after teaching it to someone else. Actually, a person does not really understand something until after teaching it to a computer, i.e. expressing it as an algorithm… An attempt to formalize things as algorithms leads to a much deeper understanding than if we simply try to comprehend things in the traditional way."

Donald Knuth from Selected Papers on Computer Science, 1996

Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
Problem: compute the \( n \)-th Fibonacci number, described here as mathematical recurrence relation

\[
F_n = \begin{cases} 
F_{n-1} + F_{n-2} & \text{if } n > 1 \\
1 & \text{if } n = 1 \\
0 & \text{if } n = 0 
\end{cases}
\]

Solution: Algorithm

Function (first draft) for computing the \( n \)-th Fibonacci number \( F_n \) (assume \( n \geq 0 \))

```python
function fib1(n):
    if n==0: return 0
    if n==1: return 1
    return fib1(n-1) + fib1(n-2)
```

A Little History

Abu Jafar Muhammad ibn Mūsā
- ca. 780 – ca. 850
- "al-Khwārizmī"
  - "the (man) of Khwarizm", his place of origin
  - Khiva in present-day Uzbekistan and Turkmenistan
- Persian astronomer and mathematician
- Worked in Baghdad

Leonardo of Pisa, "Fibonacci"?
- ca. 1170 – ca. 1250
- Coined "algorithms" in his honor
- Also imported Hindu-Arabic numeral system
Three Questions
1. Is it correct?
2. How much time does it take, as a function of \( n \)?
3. Can we do better?

Is it Correct?

```python
function fib1(n)
if n=0: return 0
if n=1: return 1
return fib1(n-1) + fib1(n-2)
```

How much time does it take as a function of \( n \)?

- \( C(n) \leq 2 \) for \( n \leq 1 \)
- For \( n > 1 \), two recursive invocations of `fib1()`
  - Requiring \( C(n - 1) \) time
  - And \( C(n - 2) \) time
- Thus, \( C(n) = C(n - 1) + C(n - 2) + 3 \)
  - for \( n > 1 \)
- Notice that \( C(n) \geq F_n \approx (2^{0.694})^n \)
- Not good!
- \( C(n) \) is exponential in \( n \)

Can we do better?

- Key idea: Store the intermediate results

```python
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 \( f[n] \)
```

Revisit: Cost

- Cost of computing a solution to a problem of a particular size:
  - Space
  - Time
- How should these be measured?

Focus: Execution Time

- How to measure execution time?
- Want to measure the algorithm, not the implementation …
- … independently of:
  - computer speed
  - programming language
  - compiler quality, etc.
Elementary Operation

- Define elementary operation
  - Depends on the problem
  - Examples:
    - Element Comparison for sorting algorithms
    - Scalar Multiplication for matrix multiplication
- \( c_{op} \): time for a particular implementation to execute an elementary operation
- \( C \): number of elementary operations
- Execution time: \( T = c_{op} \cdot C \)

Efficiency

- Efficiency is how cost grows with the difficulty \( n \) of the instance
- “Difficulty” means “size” of the instance
  - i.e., \( n \) = the number of parameters needed to completely characterize the instance
  - e.g., \( n \) = size of a list, matrix, etc.
- Example: we expect the cost of sorting a list of 100 numbers to be worse than sorting 10 numbers
  \[ T(n) \approx c_{op}C(n) \]

Orders of Growth

- Suppose \( C(n) = \frac{1}{2}n^2 \)
- How much longer will the algorithm run if we double the input size?
  \[ \frac{T(2n)}{T(n)} \approx \frac{c_{op}C(2n)}{c_{op}C(n)} \approx \frac{\frac{1}{2}(2n)^2}{\frac{1}{2}n^2} = 4 \]

Orders of Growth

- Implementation constant \( c_{op} \) is irrelevant
- Also, the coefficient \( \frac{1}{2} \) on \( C(n) \) cancelled
- Order of growth describes the functional form of \( C(n) \) up to a constant multiple as \( n \) goes to infinity

Our Problem Solving Strategy

- For a given problem,
  - Pick a computational platform
  - Write the elementary operations for that platform and their costs
  - Write an algorithm to solve the problem
  - Ask the three questions:
    - Make sure that it is correct
    - Analyze the efficiency of the algorithm
    - Look for opportunities for improvement

Assignment

- HW #0: Visual Studio
  - Install Visual Studio (follow directions in syllabus) or use an open lab machine
  - Try out C# using one of the tutorials (see links in the assignment)
  - C# Programming Guide