

Today's topics

Designing and Implementing Algorithms

Problem solving

Pseudocode

Java

Syntax and Grammars

Upcoming

More Java

Acknowledgement

Marti Hearst, UC Berkeley

David Smith, Georgia tech

Reading

Computer Science, Chapter 5

Great Ideas, Chapter 2

Problem Solving

Programming is a strenuous exercise in problem solving

G. Polya. *How to Solve It*, 2nd ed., Princeton University Press, 1957.

- **Understand the problem**
 - What are its parts? unknown, data, condition
 - Does the problem make sense? Is it feasible?
 - Think about the problem, get a sense of what it needs
- **Make a plan**
 - Find the connection between givens and result
 - What kind of problem is it? Is it familiar?
 - Think about generalizations, specializations, variants
- **Carry out the plan**
 - Check each step
- **Examine the result**
 - Does it make sense?

Back of the envelope calculations

Jon Bentley. *Programming Pearls*. 2nd edition. A-W, 2000.

<http://www.vendian.org/envelope/>

- **Engineering technique to approximate and check answers**
 - **Two answers are better than one**
 - **Quick checks**
 - **Rules of thumb**
 - **Practice**
- **Ad claims that salesperson drove 100,000 miles in a year. True?**
- **Newspaper article states that a United States quarter dollar coin has “an average life of 30 years.” How can you check that claim?**

Why “back of the envelope” estimates?

- Often need to make rapid estimates
 - to eliminate candidate solutions
 - establish feasibility
 - sketch out potential trade-offs
- Most remember key numbers related to their field, not every detail
- Hence we need to estimate
 - *which* numbers are important
 - *values* of numbers needed
 - *how* to perform the calculation
- Emphasis is on “order of magnitude” estimates
 - to nearest factor of 10 (or 2)

Orders of Magnitude

- **How far away is home? Is it more like 1, or 10, or 100 miles?**
 - Probably do not know exactly
 - Is it approximately "a couple", or "a few", or "a lot"
 - Estimate based on powers rather than multiples of 10
- **How tall is your dorm? More like 1, 10, 100, 1000 feet?**
 - 1 foot tall is like a doll house, so that's out
 - What do we know that is about 10 feet big? Hmm... People
 - If building is a couple of people high, 10 sounds good.
 - But that means 1000, would be 100 people high, so that's out
 - So 10 or 100 depending on how many people tall the building is
- **Use orders of magnitude as brackets to find reasonable range**

Estimation General Principles

- **Recall Einstein's famous advice**
 - Everything should be made as simple as possible, but no simpler
- **Do not worry about constant factors of 2, π , etc.**
 - Round to “easy” number or nearest order of magnitude
- **Guess numbers you do not know**
 - Within bounds of common sense (accuracy increases with experience)
- **Adjust geometry, etc., to suit you**
 - Assume a cow is spherical if it helps
- **Extrapolate from what you do know**
 - Use ratios to assume unknown value is similar to known quantity
- **Apply a ‘plausibility’ filter**
 - If answer seems unbelievable, it probably is
 - Can usually set range of reasonable values that indicates major mistake (e.g., speed cannot be faster than light!)

What's wrong with this algorithm?

(From back of shampoo bottle)

Directions:

Wet Hair

Apply a small amount of shampoo

Lather

Rinse

Repeat

Properties of good algorithms

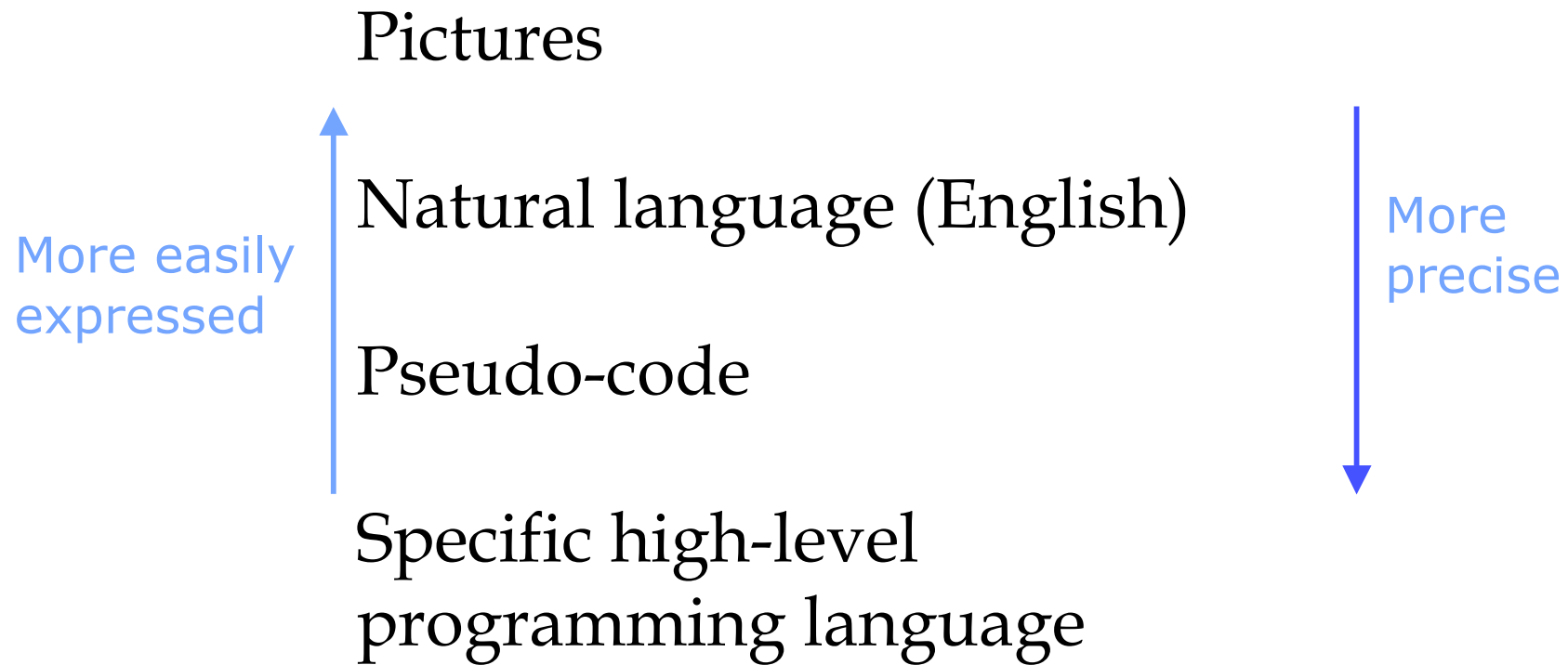
- **Good algorithms must be**
 - **Correct**
 - **Complete**
 - **Precise**
 - **Unambiguous**
- **And should be**
 - **Efficient**
 - **Simple**
 - **Contain levels of abstraction**

An algorithm is an ordered set of unambiguous, executable steps, defining a terminating process.

Algorithms

- Hand-waving not allowed!
- Specifying algorithms requires you to say what is really involved in making it work.
- Example:
 - How does a computer work?
 - Hand-wave: zeros & ones
 - Real answer: see later part of class.
- You learn to know when you don't know
 - *"I know nothing except the fact of my ignorance."*
 - Socrates, from Diogenes Laertius, Lives of Eminent Philosophers

Describing Algorithms



Pseudocode

- A shorthand for specifying algorithms
- Leaves out the implementation details
- Leaves in the essence of the algorithm

```
procedure Greetings  
Count ← 3;  
while (Count < 0) do  
    (print the message “Hello” and  
    Count ← Count + 1)
```

- What does this algorithm do?
- How many times does it print Hello?

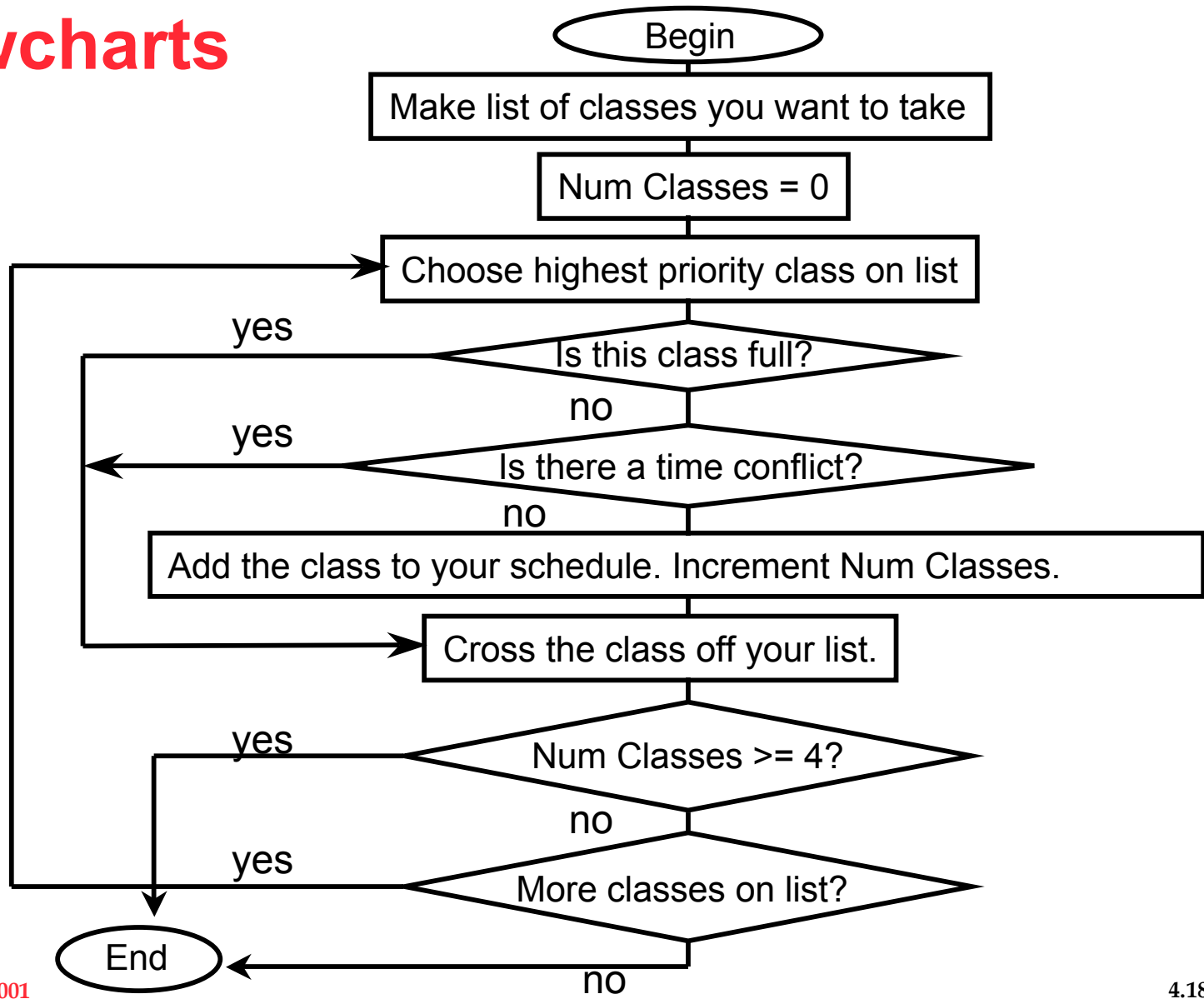
Sequential search

```
procedure Search (List, TargetValue)
if (List empty)
  then
    (Declare search a failure)
  else
    (Select the first entry in List to be TestEntry;
     while (TargetValue > TestEntry and
            there remain entries to be considered)
       do (Select the next entry in List as TestEntry.);
     if (TargetValue = TestEntry)
       then (Declare search a success.)
       else (Declare search a failure.)
    ) end if
```

Picking courses

1. Make a list of courses you want to register for, in order of priority
2. Start with empty schedule. Number of courses = 0.
3. Choose highest priority class on list.
4. If the chosen class is not full and its class time does not conflict with classes already scheduled, then register for the class (2 steps):
 1. Add the class to the schedule
 2. Increment the number of classes scheduled
5. Cross that class off of your list.
6. Repeat steps 3 through 5 until the number of classes scheduled is ≥ 4 , or until all classes have been crossed out.
7. **Stop.**

Flowcharts



Programming Primitive Operations

- Assign a value to a variable
- Call a method
- Arithmetic operation
- Comparing two numbers
- Indexing into an array
- Following an object reference
- Returning from a method

Components of Computing Algorithms

Any computing algorithm will have AT MOST five kinds of components:

- **Data structures** to hold data
- **Instructions** change data values
- **Conditional expressions** to make decisions
- **Control structures** to act on decisions
- **Modules** to make the algorithm manageable by abstraction, i.e., grouping related components

Java!

- Java is a buzzword-enabled language
- From Sun (the developers of Java),
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”
- What do all of those terms mean?

A Java Program

```
import java.awt.*;

public class HelloWorld extends
    java.applet.Applet
{
    TextField m1;
    public void init()
    {
        m1 = new TextField(60);
        m1.SetText("Hello World");
        add(m1);
    }
}
```

Definitions

- ***Algorithm***: ordered set of unambiguous executable steps, defining a terminating process
- ***Program***: instructions executed by a computer
- ***Applet***: Java program that is executed in a program such as `appletviewer` or a Java-enabled web browser
- ***Class***: family of components sharing common characteristics consisting of:
 - ***Data***: information
 - ***Method***: functionality
- ***Object***: instance of a class
- ***Variable***: represent value stored in computer memory. A variable must be defined or declared before being used
 - Sometimes synonymous with object

Grammar

- English and other natural languages have structure

<S> => <NOUN-PHRASE> <VERB-PHRASE>

=> <NOUN> | <ARTICLE> <NOUN> | <PP>

<VERB-PHRASE> => <VERB> | <VERB> <NOUN-PHRASE>

<NOUN> => DOG | FLEAS | PERSON | ...

<VERB> => RAN | BIT | ...

- Process of taking sentence and fitting it to grammar is called *parsing*

DOG BIT PERSON

<NOUN> <VERB> <NOUN>

<NOUN-PHRASE> <VERB-PHRASE>

<S>

- Parsing English is complex because of *context dependence*

Formal specifications

- Need a precise notation of syntax of a language
- Grammars can be used for generation and also can be used
- Context-free grammars

<name> => sequence of letters and/or digits that begins with a letter

<name> => guessB

<name> => msg42

- Substitute as many times as necessary. All legal statements can be generated this way
 - `Want person = firstn + " " + lastn;`
 - How do we get this from our grammar?

Random Sentence Generator

- Constructs sentences, paragraphs, and even papers that fit a prescribed format.
 - The format is specified by a set of rules called a *grammar*
 - A grammar consists of a set of definitions
 - Each definition is a set of productions
 - Examples of grammars
 - Extension request
 - College rejection
 - Poem
 - http://www.cs.duke.edu/courses/fall05/cps001/class/03_Grammars/
 - Natural languages have grammars
- <S> => <NP> <VP>

Poem Grammar

- All grammars begin with start rule

```
{  
  <start>  
  The <object> <verb> tonight. ;  
}
```

- Nonterminals are indicated by angle brackets

```
{  
  <object>  
  waves ;  
  big yellow flowers ;  
  slugs ;  
}
```

More on the poem grammar

- Nonterminals can refer to other nonterminals

```
{
  <verb>
  sigh <adverb> ;
  portend like <object> ;
}
{
  <adverb>
  warily ;
  grumpily and <adverb> ;
}
```

Generating a poem

- all sentences start with **<start>**

<start>

- There is only one production in the definition of **<start>**

The <object> <verb> tonight.

- Expand each grammar element from left to right
- "The" is a terminal, so it is simply printed –
- **<object>** is a non-terminal, so it must be expanded
- Choose one:
 - waves
 - big yellow flowers
 - **slugs**
- Suppose that '**slugs**' is chosen

Generating a poem

The slugs <verb> tonight.

- -<verb> is a non-terminal, so it must be expanded –
- Choose one:
 1. sigh <adverb>
 2. portend like <object>

The slugs sigh <adverb> tonight.

- <adverb> is a non-terminal, so it must be expanded
 - ☞ warily
 - ☞ grumpily

The slugs sigh grumpily tonight.

- "tonight." is a non-terminal so it is simply printed
- There are no more non-terminals to expand!
- The grammar has generated a complete poem

Game

- **10 coins**
 - You and a friend have a stack of 10 coins
 - On each person's turn, they remove either 1 or 2 coins from the stack
 - The person who removes the last coin wins.

 - Can you win?
- **10 coins with a twist**
 - 10 coins, can now ALSO place 1 or 2 coins back on the stack
 - Person who removes last coin wins

 - Should you go first or second, and what's your strategy