

GEM 1501 Problem Solving With Computers

Lecture 2:

Algorithmics

Martin Henz

Overview of this Lecture

- History of algorithmics
- Motivation
- Control structures

Summary of Previous Lecture

- Computers can do simple steps very fast
- Computers can solve those problems that can be broken down into sequences of simple steps
- Some problems cannot be broken down into sequences of simple steps

Focus Today: How To Take Steps?

- What are the kinds of steps that we take?
- How to control the steps?
- How to describe the steps to be taken?

Focus Next Lecture: How To Execute These Steps?

- Programming languages
- Algorithmic methods
- Correctness and efficiency

Overview of this Lecture

- **History of algorithmics**
- Motivation
- Control structures

History of Algorithms

- First algorithm
- First description of algorithmics
- First machine to execute algorithms
- First computing machine
- First programming

First Algorithm

- Euclid of Alexandria (born about 325 BC, died about 265 BC)
- Most famous for his “Elements” (used as standard textbook in geometry in schools until early 20th century!)
- Follower of Plato
- Invented the first algorithm, for finding the Greatest Common Divisor (GCD) of two positive integers
- See <http://aleph0.clarku.edu/~djoyce/java/elements/bookVII/propVII2.html>

First Description of Algorithmics

- Mohammed al-Khowârizmî (born about 780 AD in Baghdad, died about 850)
- Founder of “school algebra” (as he saw it, theory of linear and quadratic equations with a single unknown, and the elementary arithmetic of relative binomials and trinomials)
- Described many algorithms for adding, dividing, equation solving etc.

First Machine to Execute Algorithms

- Invented by Joseph Jacquard in 1801
- Domain of application: Weaving
- Algorithms described “declaratively” by holes punched in stiff cards
- Holes determine weaving pattern
- Executed by a mechanical loom

First Computing Machine

- Invented by Charles Babbage (born in 1791, Died 1871)
- Precursor was “difference engine”
- General computing device called “analytical engine”
- Programmed by punch cards
- Never completed (parts were recently built by a team of historians and mechanics)

First Computing Machine (II)

- Invented by Herman Hollerith (born 1860, died 1929)
- Based on relays (electromechanical devices)
- Used by American Census Bureau
- Gave rise to company “Tabulating Machine Company” in 1896

First Computing Machine (III)

- In the late 40s, early 50s, several machines were built that allowed general computing
- Among them ENIAC (assembled late 1945), and the Z series of computers in Germany
- Programmed using punch cards
- General “von Neumann” architecture evolved from these machines in the late 1940s

First Programming

- Ada Lovelace “programmed” Babbage’s analytical engine
- Logicians Turing, Gödel, Markov, Church, Post, Kleene all worked on formalisms for general computing and thus “programmed”
- Zuse developed “Plankalkül in 1944
- Grace Hooper’s MATH-MATIC (1950s)
- FORTRAN (1957)

Overview of this Lecture

- History of algorithmics
- **Motivation for algorithms**
- Control structures

Algorithms are Recipes

Melt chocolate and 2 tablespoons water in double boiler.
When melted, stir in powdered sugar; add butter bit by bit.
Set aside. Beat egg yolks until thick and lemon-colored,
about 5 minutes...

Loops

Loops allow short descriptions of long processes. Example: compute the total sum of all salaries in a list of employees

1. make a note of the number 0
2. proceed through the list, adding each employee's salary to the noted number
3. having reached the end of the list, produce the noted number as output

Algorithmic Problems

- Characterize input/output behavior (algorithmic problem)
- Formulate an algorithm A
- A should be given input and produce output

Overview of this Lecture

- History of algorithmics
- Motivation for algorithms
- **Control structures**

Control Structures

- Direct sequencing
- Conditional branching
- Bounded iteration
- Conditional iteration
- Subroutines
- Recursion

Direct Sequencing

- “do A followed by B”
- “gently fold in chocolate; reheat slightly”
- In programming, often the ; symbol is enough
- A; B
- Sometimes, just juxtaposition does the job
- A B

Conditional Branching

- “if Q then do A otherwise do B”
- “if Q then do A” (otherwise do nothing)
- “reheat slightly to melt chocolate, if necessary”

Combining Sequencing and Branching

- `A; if Q then B; C else D; E`
- The precedence of the operators matters
- `if Q then A; B`
- `(if Q then A); B`

Bounded Iteration

- “do A exactly N times”
- N is a fixed (or computed) number
- allows to describe long computations with short programs

Conditional Iteration

- “repeat A until Q”
- “while Q do A”
- “beat egg whites until foamy”
- combines iteration with conditional branching

Example: Salary Computation

1. make a note of 0; point to the first salary on the list;
2. do the following $N - 1$ times
 - (a) add the salary pointed at to the noted number;
 - (b) point to the next salary;
3. add the salary pointed at to the noted number
4. produce the noted number as output

Another Example: Bubble Sort

- Given: A list of numbers
- Desired: Sorted list with the same numbers
- Algorithm: go “often enough” through the list and exchange neighboring numbers

Bubble Sort

1. do the following $N - 1$ times

(a) point to the first element

(b) do the following $N - 1$ times

i. compare the element pointed to with the next element

ii. if the compared elements are in the wrong order, exchange them

iii. point to the next element

Diagrams from Algorithms

- Describe algorithms in 2-D
- Use arrows to describe the flow of control
- Not all diagrams can be written in any programming language
- Arrows correspond to “goto” instructions on some languages

Subroutines: Example

Let us count the number of sentences in which the word “money” occurs.

Subroutine: search-for X

1. do the following until either the combination X is being pointed at, or the end of the text is reached:
 - (a) advance the pointer one symbol in the text;
2. if the end of the text is reached, output the counter's value and stop;
3. otherwise return to the main algorithm

Main Algorithm

1. set counter to 0; pointer to the start of the text
2. do the following until the end of the text is reached
 - (a) search-for “money”
 - (b) search-for “.”
 - (c) increment counter

Recursion: Towers of Hanoi

1. Move a tower of rings from A to B using C
2. No larger ring can sit on top a smaller ring
3. Idea: use subroutine for moving smaller tower

Subroutine: move N from X to Y using Z

1. if N is 1 then output “move X to Y ”

2. otherwise
 - (a) call move $N - 1$ from X to Z using Y
 - (b) output “move X to Y ”
 - (c) call move $N - 1$ from Z to Y using X

Main Algorithm

call move 3 from A to B using C

Next Week

- Programming languages
- Introduction to Oz