

GEM 1501 Problem Solving With Computers

Lecture 2:

Algorithmics

Frank Stephan

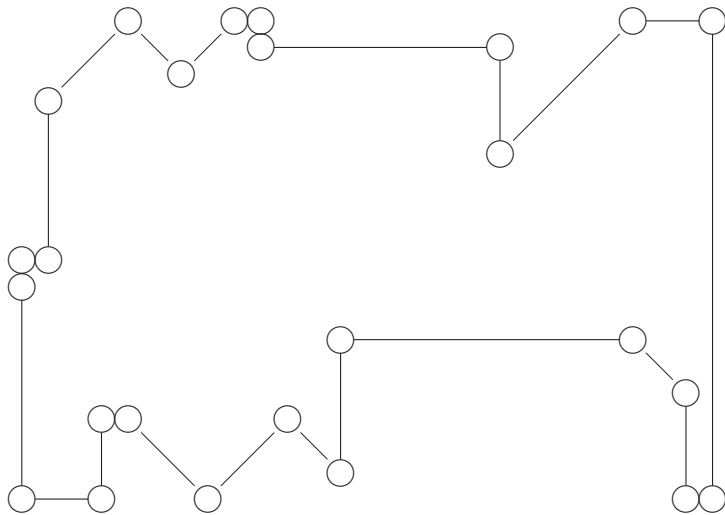
Repetition Lecture 1

- Overview of Course
 - Selected Chapters of Theoretical Computer Science
 - Practical Exercises (Java Script)
- Definition of a Problem
- Example of Transportation Problem
- Formal Approach
- Euclid's Algorithm
- History of π — Example for Computing Performance

Travelling Salesman Problem

A salesman wants to visit from his hometown 24 other places for doing business and then return home.

He draws the route into a map.



Is given round tour really the shortest possible?

If not, find the shortest.

Do not visit any place twice.

Formalization

- Predicates: $X(Y)$ says “The animal Y has the colour X ”.
- Implication: $\text{Brown}(\text{Cat}) \Rightarrow \text{White}(\text{Sheep})$.
- Disjunction (Or): $\text{Black}(\text{Sheep}) \vee \text{White}(\text{Sheep})$.
- Negation (Not): $\neg \text{Brown}(\text{Sheep})$.
- Conjunction (And): $\neg \text{Brown}(\text{Sheep}) \wedge \neg \text{Brown}(\text{Mouse})$.
- Existential Quantification: $\exists \text{ Animal } Y \ [\text{White}(Y) \vee \text{Black}(Y)]$.
- Universal Quantification: $\forall \text{ Animal } Y \ [\neg \text{Red}(Y) \wedge \neg \text{Blue}(Y)]$.
- Formal reasoning gave solution (last lecture):
 $\text{Black}(\text{Sheep}) \wedge \text{White}(\text{Cat}) \wedge \text{Grey}(\text{Mouse})$.

Overview of this Lecture

- History of algorithmics
- Motivation
- Control structures

How do computer solve problems?

- Computers can do a lot of simple steps very fast
- Computers solve problems which are broken down into sequences of simple steps
- Warning: Some problems cannot be broken down into sequences of simple steps (computers cannot solve such problems)

Overview Today and Next Week

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?

Next Lecture: How To Execute These Steps?

- Programming languages
- Algorithmic methods
- Correctness and efficiency

A bit of programming languages comes already up today.

Overview of this Lecture

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

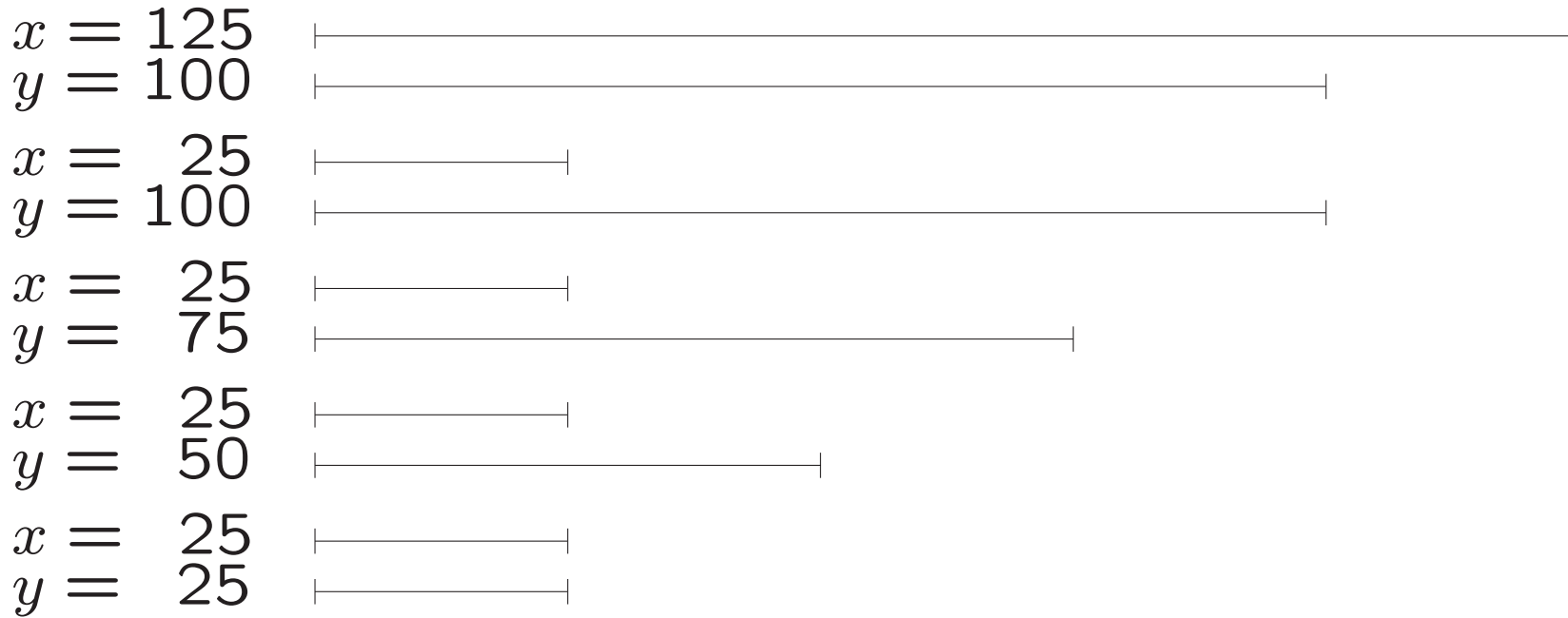
History of Algorithms

- 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 (= Factor) of two positive integers
- <http://aleph0.clarku.edu/~djoyce/java/elements/bookVII/propVII2.html>

Recall Lecture 1



Find longest line such that two given lines are multiples of it.

While lines are not of equal length, subtract shorter line from longer one.

First Description of Algorithmics

- Mohammed al-Khowârizmî (born about 780 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)

$$2 \cdot x^2 - 12 \cdot x + 10 = 0$$

$$x^2 - 6 \cdot x + 5 = 0$$

$$x^2 - 2 \cdot 3 \cdot x + 3^2 = 3^2 - 5 = 4$$

$$(x - 3)^2 = 4 = (-2)^2 = 2^2$$

$$x - 3 \in \{-2, 2\}$$

$$x \in \{1, 5\}$$

- Described many algorithms, for example to add, subtract, multiply and divide numbers and to solve equations.

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 (1791–1871)
- 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 (1860–1929)
- Based on relays (electromechanical devices)
Tasks like sorting and counting punch cards for statistical purposes
- Used by American Census Bureau
Hollerith constructed it to win a competition of the American Census Bureau how to tabulate census data
1880–1887: seven years of eleventh census
1890: six weeks with punch cards and Hollerith's machines
Methods used until approximately 1960
- Gave rise to company “Tabulating Machine Company” in 1896

First Computing Machine (III)

- In the late 1940s, early 1950s, 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)

FORTRAN EXAMPLE

- FORTRAN = “Formula Translator”;
Several Revisions like “FORTRAN 77” and “Fortran 90”
- This example tries to use old features

```
10 PROGRAM EUCLID
20 I = 125
30 J = 100
40 IF (I-J) 50, 90, 70
50 J = J-I
60 GO TO 40
70 I = I-J
80 GO TO 40
90 PRINT *, 10HResult is , I
```
- More about FORTRAN: a Fortran 77 Language Reference can be found on <http://www.star.le.ac.uk/~cgp/prof77.html>

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-coloured, about 5 minutes ...

Food Factories might even employ computers to cook according to certain recipes.

Loops

Loops allow short descriptions of long processes.

Example: Producing fruit salad

- Collect fruits on a pile
- Fetch a bowl
- For every fruit on the pile do
 - Remove uneatable parts like peel or cherrystone
 - Cut fruit into pieces
 - Put pieces into the bowl
- Put some sauce into the bowl
- Mix the fruit pieces in the bowl with the sauce
- Serve with some cream

Loops ...

A more mathematical example from real life bureaucracy:
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 behaviour (algorithmic problem)
- Formulate an algorithm A
- A should be given input and produce output

Ingredients of an Algorithm

- Information: stored in files and variables;
- Statements in program: manipulate variables;
- Control-Structures: organize the order in which statements are executed, jump from one point in the program to another one.

Information and Control

- 10 PROGRAM EUCLID
20 I = 125
30 J = 100
40 IF (I-J) 50, 90, 70
50 J = J-I
60 GO TO 40
70 I = I-J
80 GO TO 40
90 PRINT *, 10HResult is , I
- Information: here integer numbers in variables I and J
- Line Numbers: Positions in program
- Control-Flow: Program at each time in one line of program
- Statements tell which variable to update or which line to go

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

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

Statements in Java Script

- Statements always followed by semicolon
- Groups of statements could be put together by brackets, no extra semicolon;
- Examples:

```
document.write("The number is "+x); y = 5;
```

```
{x = 2; y = 8; z=z+1; }
```

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”

Conditional Branching in Java Script

- `if (x == 5) {y = 4; }`
- `if (y != x) {z = 3; } else {z = 4; }`
- Comparisons: `==` `!=` `<` `<=` `>` `>=`
- Logical connectives: `&&` `||` `!`
- The following logical expressions are equivalent:
`(x <= y)`
`((x == y) || (x < y))`
`(!((x != y) && (x >= y)))`

Combining Sequencing and Branching

- An example:

```
A; if Q then {B; C; } else {D; E; }
```

- The precedence of the operators matters

```
(1) if Q then {A; B; }
```

```
(2) {if Q then A; } B;
```

In case (2), B is always done.

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 steps 2.1. and 2.2. exactly $N - 1$ times
 - 2.1. Add the salary pointed at to the noted number;
 - 2.2. Point to the next salary;
3. Add the salary pointed at to the noted number
4. Produce the noted number as output

Run Time Example

Eberhard Ei Incorporated Ltd

Employee Name	Salary	Noted Number
		SGD 0000.00
Anneliese Aal	SGD 1324.35	SGD 1324.35
Boris Bratling	SGD 1122.44	SGD 2446.79
Claudia Creme	SGD 4044.04	SGD 6490.83
Doris Dattel	SGD 1000.01	SGD 7490.84
Output		SGD 7490.84

Computing $1^2 + 2^2 + 3^2 + \dots + 100^2$

- Adding Algorithm with one Loop in Java Script.

This slide: For Loop (bounded iteration)

- ```
var i; var j = 0; var h = 100;
for(i=1;i<=h;i=i+1)
 { j=j+(i*i); }
document.write("Result is "+j+".");
```
- Faster with  $j = \frac{1}{3}h^3 + \frac{1}{2}h^2 + \frac{1}{6}h = 338350$ .

# Conditional Iteration in Java Script

- While Loops and Do-While Loops to add  $1^2 + 2^2 + \dots + h^2$ .
- ```
var i = 1; var j = 0; var h = 100;
while (i <= h)
    { j = j+(i*i); i = i+1; }
document.write("Result is "+j+".");
```
- ```
var i = 1; var j = 0; var h = 100;
do
 { j = j+(i*i); i = i+1; }
while (i <= h);
document.write("Result is "+j+".");
```
- What happens if third variable initialized as `var h = 0`?  
First loop is never and second loop is executed once;  
Results are 0 and 1, respectively.

# Euclid's Algorithm in Java Script

- Euclid's Algorithm has one Conditional Loop containing one Condition Branching.
- Here Euclid's algorithm with input fixed to 125 and 100 written in Java Script.
- ```
var i = 125; var j = 100;
while (i != j)
    { if (i > j) { i = i-j; }
      else { j = j-i; } }
document.write("Result is "+i+".");
```

Another Example: Bubble Sort

- Given: A list of numbers
- Desired: Sorted list with the same numbers
- Algorithm: go “often enough” through the list and put neighbouring numbers into correct order

Bubble Sort

1. do substeps 1.1. and 1.2. $N - 1$ times

1.1. point to the first element

1.2. do substeps 1.2.1., 1.2.2. and 1.2.3. $N - 1$ times

1.2.1. compare the element pointed to with the next element

1.2.2. if the compared elements are in the wrong order
then exchange them

1.2.3. point to the next element

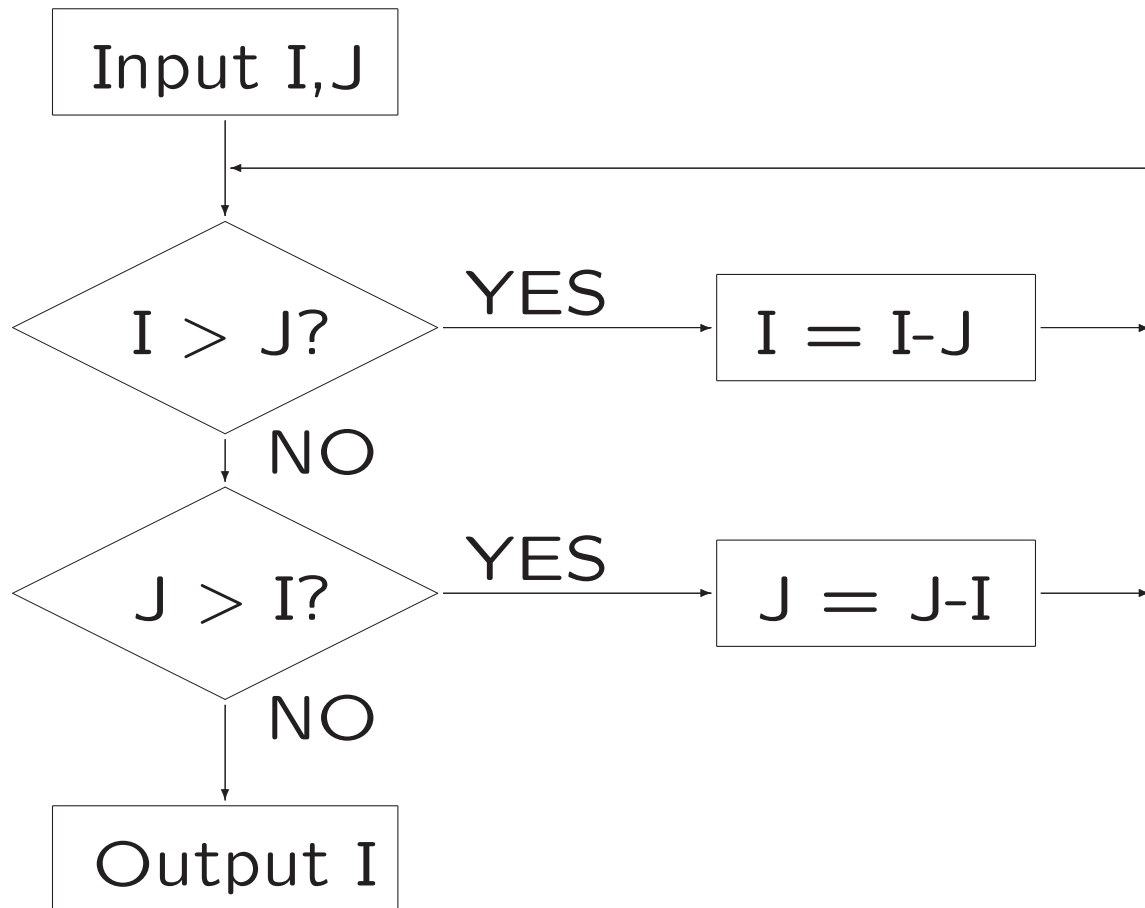
Sorting “this is a text”

- this is a text
- Exchange: “this” and “is”, “this” and “a”, “this” and “text”
- is a text this
- Exchange: “is” and “a”
- a is text this
- Exchange: none, list already sorted
- a is text this

Diagrams from Algorithms

- Standard instructions and tests in boxes and diamonds.
- Arrows are used to describe the flow of control.
- Easiest method to translate diagrams into FORTRAN is to use line numbers for all boxes and to connect them with goto statements and if statements.
- Programming languages without goto statements might have problems to translate diagrams exactly; but they have programs equivalent to given diagrams.

Euclid's Algorithm as a Flowchart



Subroutines: Example

Let us count the number of sentences in which the word “money” occurs for a sample text:

He has a lot of money, doesn't he. She has much more money than he, really much more money. The dog is brown.

A sentence is counted, if the word money occurs in it and there is a period (“.”) at the end of the sentence. Thus the result is 2.

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:
 - 1.1. 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

Sample text: He has a lot of money, doesn't he. She has much more money than he, really much more money. The dog is brown.

The Towers of Hanoi

- Three Pegs A, B, C
- M Rings R_1, R_2, \dots, R_M
- Initially all rings on peg A
- At the end all rings should be on peg B
- Order: If $I < J$ then R_I can never be below R_J
- Moves: In each step, only one ring can be moved

Recursion: Moving towertops

- $\text{Move}(1, X, Y)$ moves R_1 from peg X to peg Y ;
Note that R_1 is always on the top of a tower
- $\text{Move}(N, X, Y)$ is a series of moves
It can only be done if R_1, R_2, \dots, R_N are on peg X
It will leave them on peg Y
- $\text{Move}(2, A, B)$ first moves R_1 from A to C , then R_2 from A to B and then R_1 from C to A ; it uses the third tower

Algorithm for Subroutine Move(N, X, Y)

1. if R_1, \dots, R_N are not on top of tower X then abort
2. if $N = 1$ then output “move R_1 from X to Y ”
3. if $N > 1$ then
 - (a) call Subroutine Move($N - 1, X, Z$)
 - (b) output “move R_N from X to Y ”
 - (c) call Subroutine Move($N - 1, Z, Y$)

Main Algorithm

Call $\text{Move}(M, A, B)$.

Example for $M = 3$.

$$\text{Move}(3, A, B) \left\{ \begin{array}{l} \text{Move}(2, A, C) \\ \text{Move}(2, C, B) \end{array} \right. \left\{ \begin{array}{l} \text{Move}(1, A, B) \quad R_1 : A \rightarrow B \\ \text{Move}(1, B, C) \quad R_1 : B \rightarrow C \\ \text{Move}(1, C, A) \quad R_1 : C \rightarrow A \\ \text{Move}(1, A, B) \quad R_1 : A \rightarrow B \end{array} \right. \begin{array}{l} R_2 : A \rightarrow C \\ R_3 : A \rightarrow B \\ R_2 : C \rightarrow B \\ R_1 : A \rightarrow B \end{array}$$

Functions in Java Script

- Functions implement subroutines and recursion
- An Example – Function declaration:

```
function fibonacci(y)
{ var z;
  if (y > 1) { z = fibonacci(y-1)+fibonacci(y-2); }
  else { z = 1; }
  return(z); }
```
- Main call: `document.write("FN(20) = "+fibonacci(20));`
- Warning: This implementation is inefficient. Test it on <http://www.comp.nus.edu.sg/~gem1501/fibonacci.html>

Next Week

- An Overview on Programming languages
- Data Types in Java Script

Numbers: 0,1,2,3,4; 3.1415926353...

Texts: "Singapore", "Sydney", "Melbourne"

Logical (Boolean): false, true

And Much More ...

Laboratory Sessions

- Labs start next week
Labs in Week 3, Week 4, Week 5, Week 6, ...
- Location: COM1#B-11, “Programming Lab 4”
- Fridays 12:00-14:00 and 14:00-16:00 hrs
- Create your account: <https://mysoc.nus.edu.sg/~newacct>
- Read the Introduction to Java Script on the course homepage as preparation for laboratory session.