

Introduction to Algorithms and Data Structures

Lectures: Monday 3 p.m. LT05
 Friday 11 a.m. LT19
Tutorials: Monday 1-2 p.m.
Consultation hour: Friday 3–4 p.m., 9.10o

Module Plan

Analysis of algorithms – iterative algorithms
 – recursive algorithms

Searching algorithms – sequential search
 – binary search

Sorting algorithms – selection sort
 – insertion sort
 – bubble sort
 – merge-sort
 – quick-sort

Data structures – arrays
 – lists
 – stacks
 – queues
 – priority queues

Books:

- A.V. Aho, J.E. Hopcroft, J.D. Ullman, “Data Structures and Algorithms”, Addison Wesley, 1983. CS25
- F.M. Carrano, J.J. Prichard, “Data Abstraction and Problem Solving with Java: Walls and Mirrors”, Addison Wesley, 2003.
- * T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, “Introduction to Algorithms”, MIT Press, 2001. CS25 CS33
- M.T. Goodrich, R. Tamassia, “Algorithm Design”, John Wiley & Sons, 2002. CS25
- * A. Levitin, “Introduction to the Design and Analysis of Algorithms”, Addison Wesley, 2002 CS25

Introduction

1. Why do you need to study algorithms?

- The standard set of important algorithms form a programmer's basic algorithms “toolkit”.
- The commonly used data structures form a programmer's basic data structure “toolkit”.
- The choice of an algorithm and data structure can make the difference between a program running in a few seconds or many days.

2. What is an algorithm?

The word *algorithm* derives from the name of the famous Persian mathematician *Mohammed ibn-Musa Al-Khowarizmi*, who lived from about 780 to 850. The Europeans translated his name into Latin as *Algorismus*.

Al-Khowarizmi described the procedure (a sequence of instructions) for solving linear and quadratic equations.

An *algorithm* is a step-by-step procedure for solving a problem in a finite amount of time. A *data structure* is a systematic way of organising and accessing data.

Examples of algorithms

- travel instructions
- cooking recipes
- a car manual page (how to remove the gearbox)
- a medical procedure

Properties of an algorithm

- It must be composed of a series of concrete steps.
- It must terminate after a finite number of steps.
- There must be no ambiguity as to which step will be performed next.
- It must be correct.

3. What is pseudocode?

Pseudocode is a mixture of natural language and a high-level programming language. It is more structured than natural language, less formal than a programming language. It is clear and informative.

Relaxed syntax:

- no input/output statements
- no variable declarations
- variable-length arrays: $A[0 \dots n-1]$
- no `{ } ;`
- \leq instead of `<=`, \neq instead of `!=`
- no technical details, shorthand descriptions

Types of statements:

- assignment

```
i ← 1
```

- If condition then true-actions [else false-actions]

```
if i < n
  application code
else
  application code
```

- For-loops

```
for i ← 1 to n do
  application code
```

- While-loops

```
i ← 1
while i ≤ n do
  application code
  i ← i + 1
```

Example 1. Suppose we have to find the maximum of three numbers a , b , and c and the output parameter is x . Compare the algorithm description in natural language and in pseudocode.

Natural language:

Compare a and b ; find the largest and assign it to variable x .

Compare c and x ; update x , if required.

Pseudocode:

```
ALGORITHM findLargest( $a, b, c$ )
//The algorithm finds the maximum of three numbers
//Input: integers  $a, b, c$ 
//Output: integer  $x$ 
 $x \leftarrow a$ 
if  $b > x$ 
     $x \leftarrow b$ 
if  $c > x$ 
     $x \leftarrow c$ 
return  $x$ 
```

Example 2. Compare the following fragments in pseudocode and Java:

Pseudocode	Java
if n is odd	if ($n\%2 == 1$)
if $low \leq x \leq high$	if ($low \leq x \ \&\& \ x \leq high$)
swap integers a and b	int $temp = a$; $a = b$; $b = temp$;
Copy the first half of array A to S_1 Copy the second half of A to S_2	// Input: array A of n integers // Output: arrays $S1$ and $S2$ int $i, n1, n2$; //Compute the sizes of two halves $n1 = n/2$; $n2 = n - n1$; //Copy the first half of A to $S1$ for ($i=0; i<n1; i++$) $S1[i] = A[i]$; //Copy the second half of A to $S2$ for ($i=0; i<n2; i++$) $S2[i] = A[n1+i]$;