CH08-320201
Algorithms and Data Structures

Prof. Dr.-Ing. Lars Linsen

Jacobs University
Spring 2016

0. Introduction

0.1 Syllabus and Organization

Course website

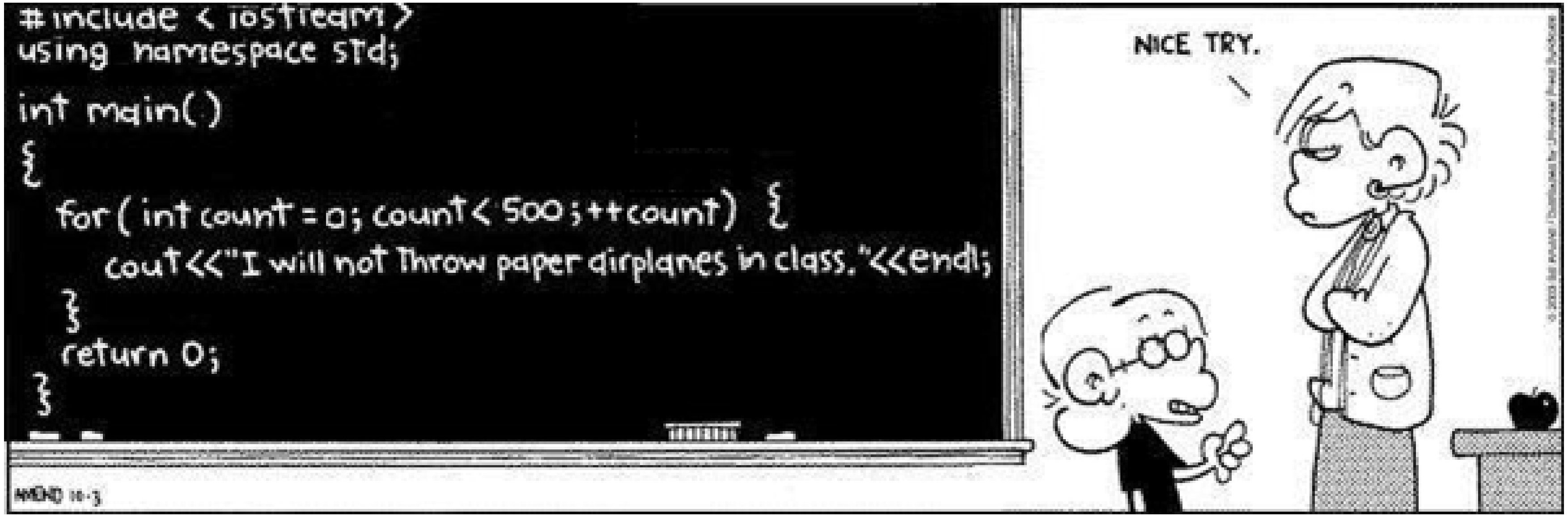
http://www.faculty.jacobs-university.de/llinsen/teaching/320201.htm

(accessible through CampusNet)

Content

- This course introduces a basic set of data structures and algorithms that form the basis of almost all computer programs. The data structures and algorithms are analyzed in respect to their computational complexity with techniques such as worst case and amortized analysis.
- Topics: Fundamental data structures (lists, stacks, trees, hash tables), fundamental algorithms (sorting, searching, graph traversal).

Prerequisites



```
#include <iostream>
using namespace std;
int main()
{
  for(int count=0; count<500;++count) {
     cout<<"I will not Throw paper airplanes in class."<<endl;
  }
  return 0;
}
```

Lectures

- Times:
  - Tuesday 11:15am-12:30am,
  - Thursday 9:45am-11:00am.
- Location: Research III Lecture Hall

Instructor

- Lars Linsen
- Office: Res I, 128.
- Phone: 3196
- E-Mail: l.linsen [@jacobs-university.de]
- Office hours: by appointment

Tutorials

- Times: tbd
- Location: tbd
- TAs: Rubin Deliallisi, Joana Halili, Jinbo Zhang.

Assignments

- The homework assignments include theoretical and practical problems that tackle topics from the lectures.
- The homework assignments are handed out on a regular basis.
- Solutions that are handed in late lead to reduced credit (-15% per day).
- Exceptions are only made with an official excuse.
- With an official excuse of up to 4 days, the deadline for the respective homework is extended by the same amount of days.
- With an official excuse of more than 4 days, the respective homework will not count.
- Handing in via JGrader.

Exams

- There will be one midterm and a final exam.
- There are no quizzes planned.

Grading

- **Homework** assignments contribute **35%** to the overall grade.
- The **midterm** exam contributes **25%.**
- The **final** exam contributes **40%.**

Dates – Lectures (1)

Week 1	02.02.	Lecture 1
	04.02.	Lecture 2
Week 2	09.02.	Lecture 3
	11.02.	Lecture 4
Week 3	16.02.	Lecture 5
	18.02.	Lecture 6
Week 4	23.02.	Lecture 7
	25.02.	Lecture 8
Week 5	01.03.	No lecture - off campus
	03.03.	Lecture 9
Week 6	08.03.	Lecture 10
	10.03.	Lecture 11
Week 7	15.03.	Lecture 12
	17.03.	Lecture 13

Dates – Lectures (2)

Week 8	29.03.	Midterm
	31.03.	Lecture 14
Week 9	05.04.	Lecture 15
	07.04.	Lecture 16
Week 10	12.04.	Lecture 17
	14.04.	Lecture 18
Week 11	19.04.	Lecture 19
	21.04.	Lecture 20
Week 12	26.04.	Lecture 21
	28.04.	Lecture 22
Week 13	03.05.	Lecture 23
	05.05.	Holiday
Week 14	10.05.	Lecture 24
	12.05.	Lecture 25

Dates - Exams

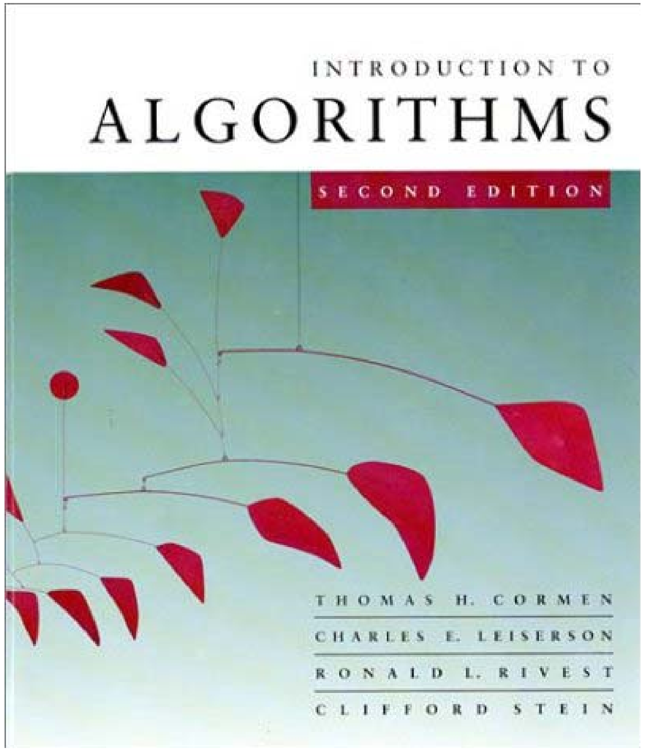
- Midterm: March 29, 2016
- Final: tbd (finals' week)

Dates - Assignments

Homework 1	Handed out: 05.02.	Due: 12.02. at 7pm
Homework 2	Handed out: 12.02.	Due: 19.02. at 7pm
Homework 3	Handed out: 19.02.	Due: 26.02. at 7pm
Homework 4	Handed out: 26.02.	Due: 04.03. at 7pm
Homework 5	Handed out: 04.03.	Due: 11.03. at 7pm
Homework 6	Handed out: 11.03.	Due: 18.03. at 7pm
Homework 7	Handed out: 18.03.	Due: 01.04. at 7pm
Homework 8	Handed out: 01.04.	Due: 08.04. at 7pm
Homework 9	Handed out: 08.04.	Due: 15.04. at 7pm
Homework 10	Handed out: 15.04.	Due: 22.04. at 7pm
Homework 11	Handed out: 22.04.	Due: 29.04. at 7pm
Homework 12	Handed out: 29.04.	Due: 06.05. at 7pm
Homework 13	Handed out: 06.05.	Due: 13.05. at 7pm

Literature

- Introduction to Algorithms. Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, 3rd edition, MIT Press, 2009.



0.2 Goals

Goals

- The objective of the course is to learn about
  - fundamental algorithms for solving problems efficiently,
  - basic algorithmic concepts,
  - the analysis of algorithms, and
  - fundamental data structures for efficiently storing, accessing, and modifying data.

0.3 Content

Content

1. Foundations
2. Sorting & Searching
3. Fundamental Data Structures
4. Design Concepts
5. Graph Algorithms
6. Computational Geometry

1. Foundations

1.1 Definitions

Definition: Algorithm

- An algorithm is a sequence of computational steps which transforms a set of values (input) to another set of values (desired output).
- It is a tool for solving a well-defined computational problem.
- Step-wise procedure that can be implemented in a computer program.
- Consists of a finite list of well-defined instructions (Turing machine).
- 'Algorithm' stems from 'Algoritmi', the Latin form of al-Khwārizmī, a Persian mathematician, astronomer and geographer.

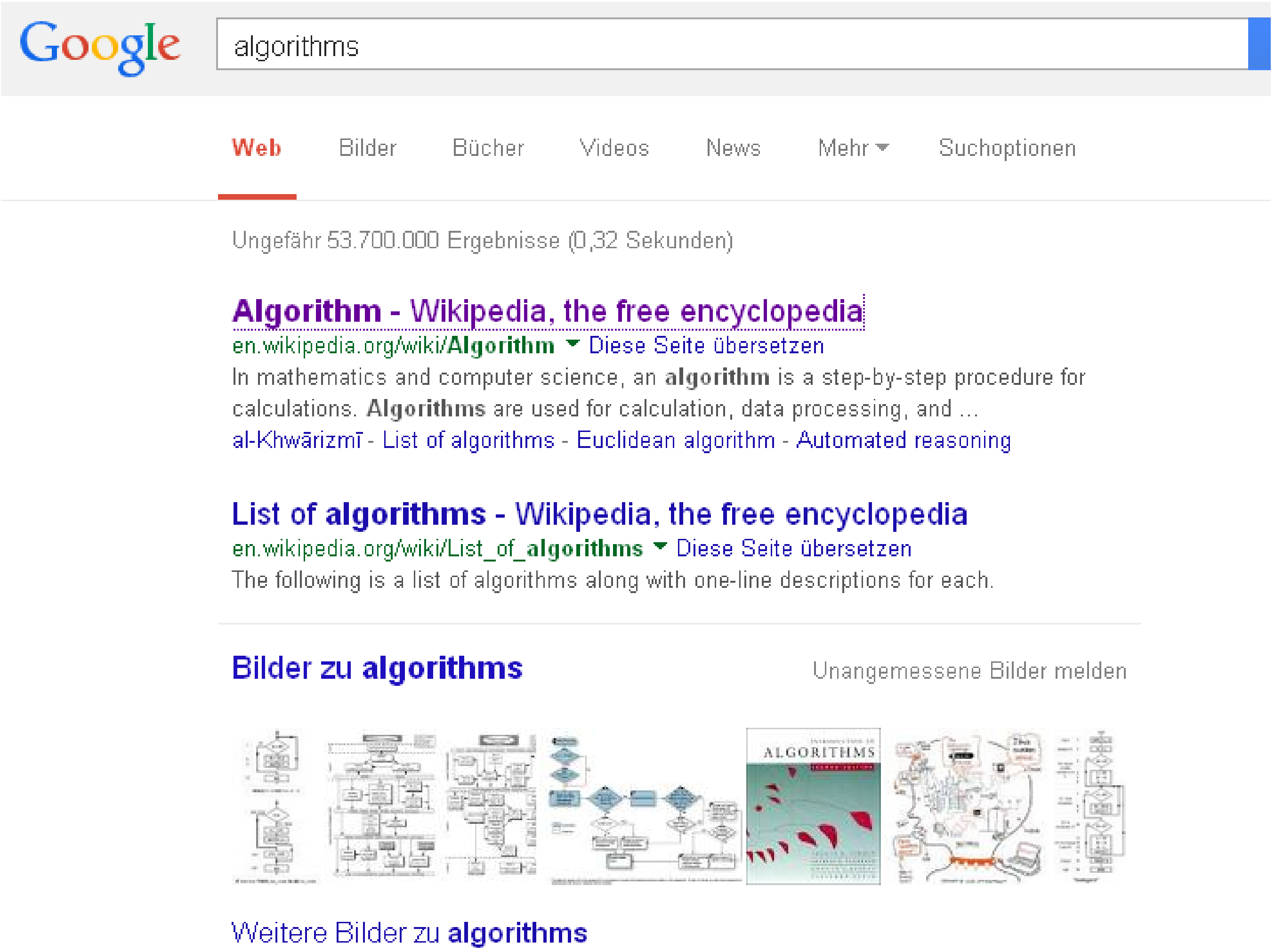
Example: Sorting Problem

- Input:
  sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

- Output:
  permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$
  such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$

- Example (instance of sorting problem):
  Input: 8 2 4 9 3 6
  Output: 2 3 4 6 8 9

Example: Searching



algorithms

Ungefähr 53.700.000 Ergebnisse (0,32 Sekunden)

Algorithm - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Algorithm** ▾ Diese Seite übersetzen
In mathematics and computer science, an **algorithm** is a step-by-step procedure for calculations. **Algorithms** are used for calculation, data processing, and ...
al-Khwārizmī - List of algorithms - Euclidean algorithm - Automated reasoning

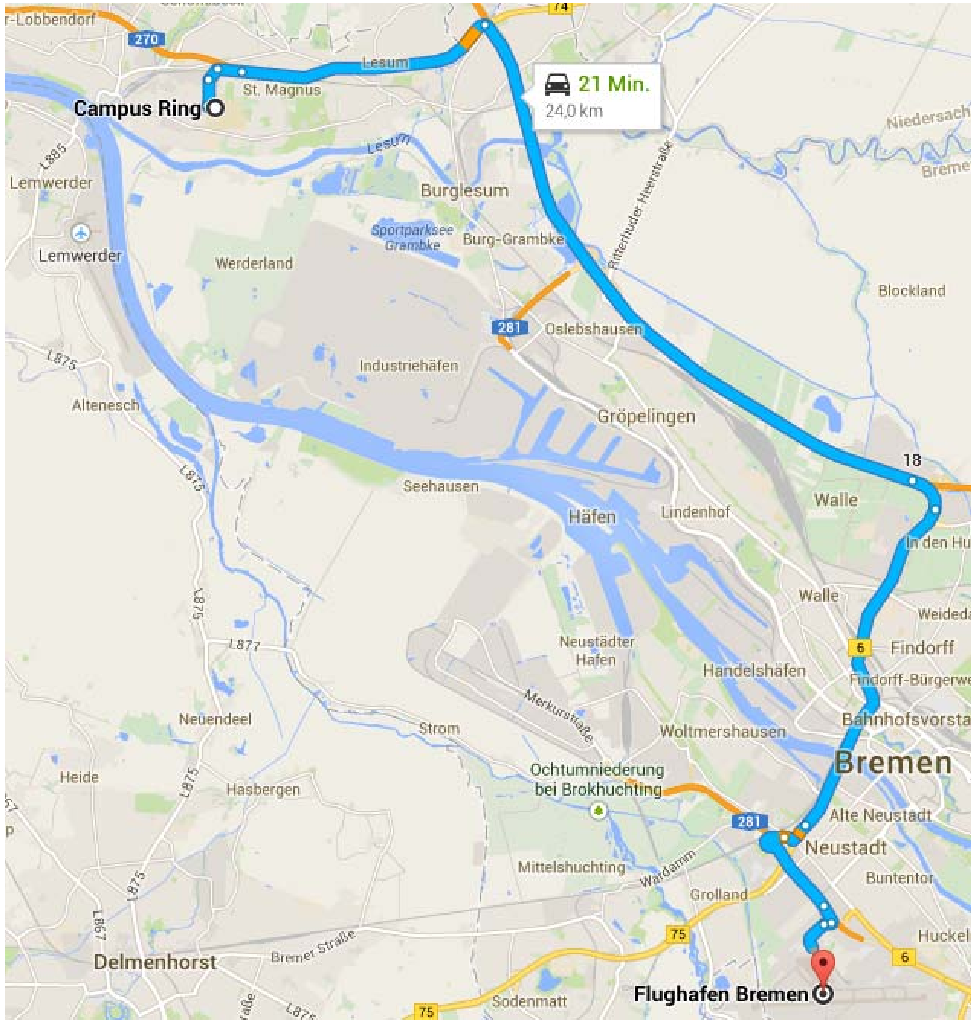
List of **algorithms** - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/List_of_**algorithms** ▾ Diese Seite übersetzen
The following is a list of algorithms along with one-line descriptions for each.

Bilder zu **algorithms**

Unangemessene Bilder melden

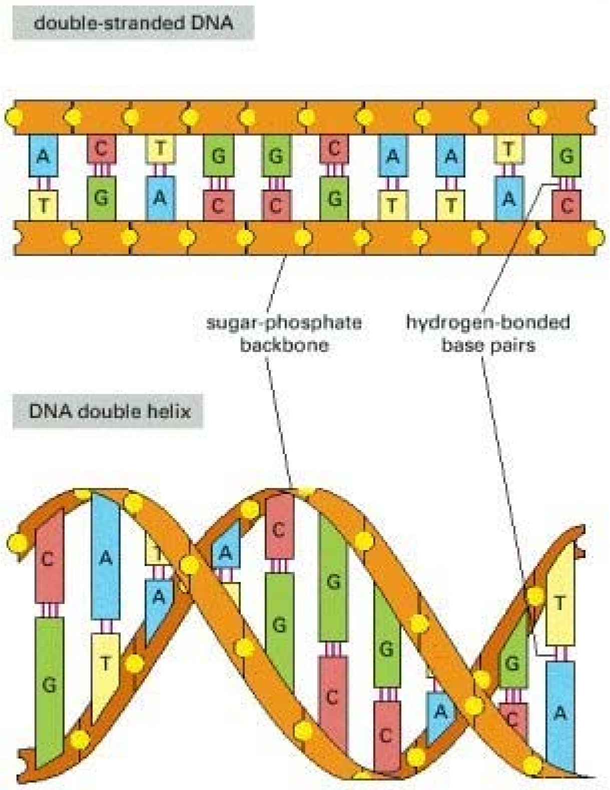
Weitere Bilder zu **algorithms**

Example: Road map



Graph algorithm

Example: DNA sequences



String matching

Analysis of algorithms

- The theoretical study of computer-program performance and resource usage.
- Other design goals?
  - correctness
  - functionality
  - robustness
  - reliability
  - user-friendliness
  - programmer time
  - simplicity
  - modularity
  - maintainability
  - extensibility

Performance of Algorithms

- Analysis helps us to understand scalability.
- Performance often draws the line between what is feasible and what is impossible.
- Algorithmic mathematics provides a language for talking about program behavior.
- “Performance is the currency of computing.“
- The lessons of program performance generalize to other computing resources.

Definition: Data Structure

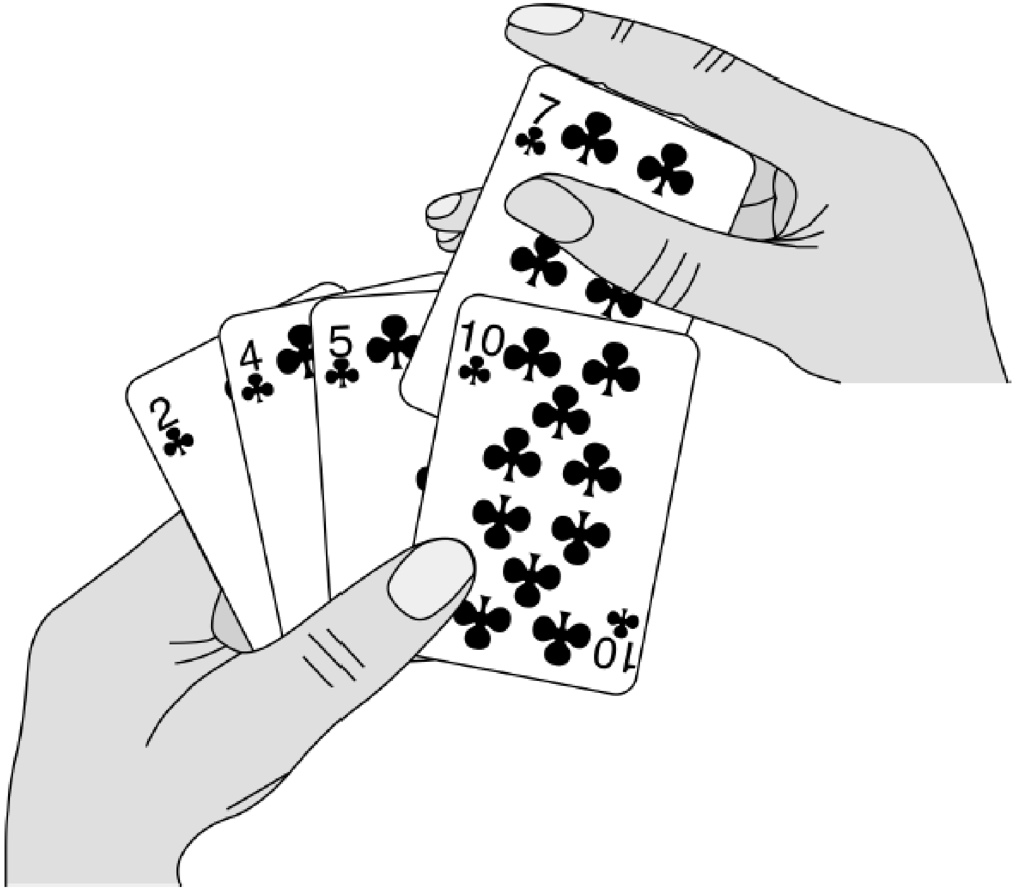
- A data structure is a way to store and organize data in order to facilitate access and modification.

- There is typically no best data structure, but each data structure has its strengths and weaknesses.
- Which data structure to use, depends on the problem that is to be solved.
- Sometimes there is a trade-off between storage (in a data structure) and speed (in accessing a data structure or of an algorithm).

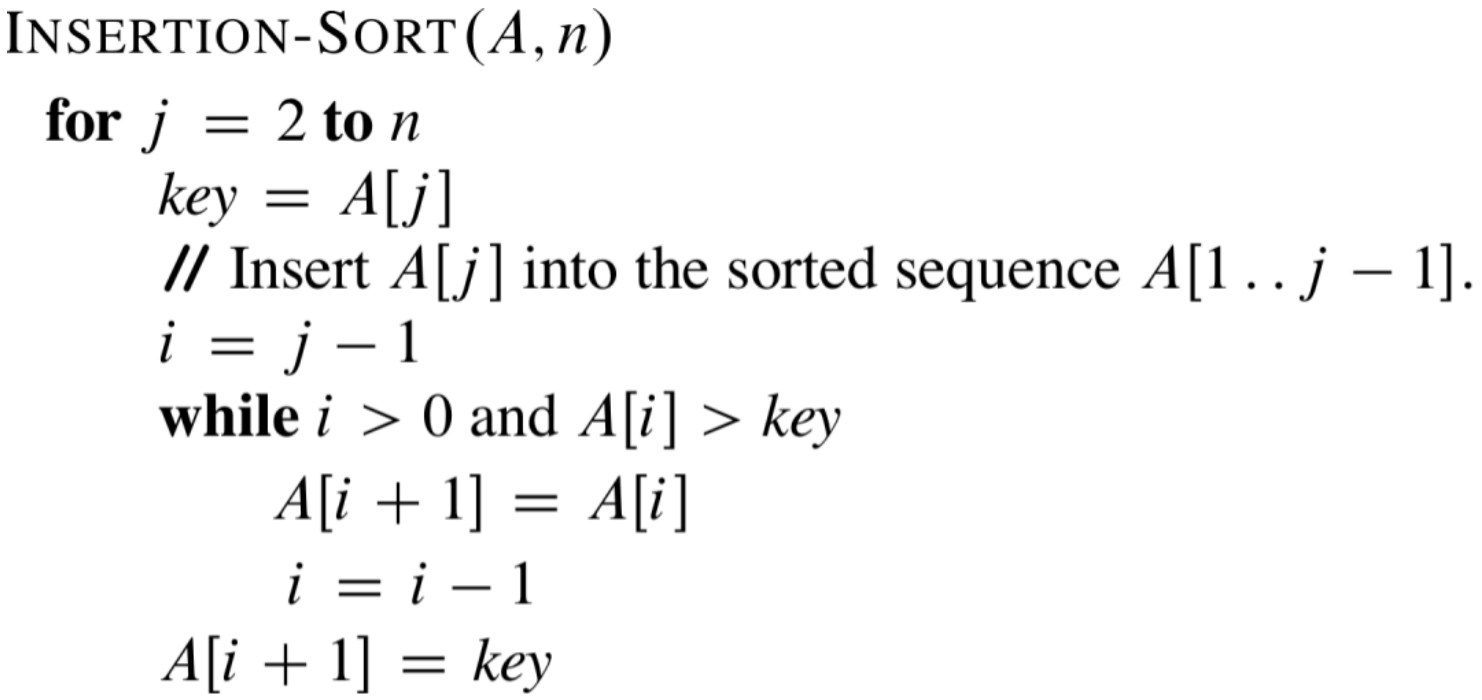
1.2 First example: Insertion Sort

Sorting problem

- First algorithm: Insertion Sort



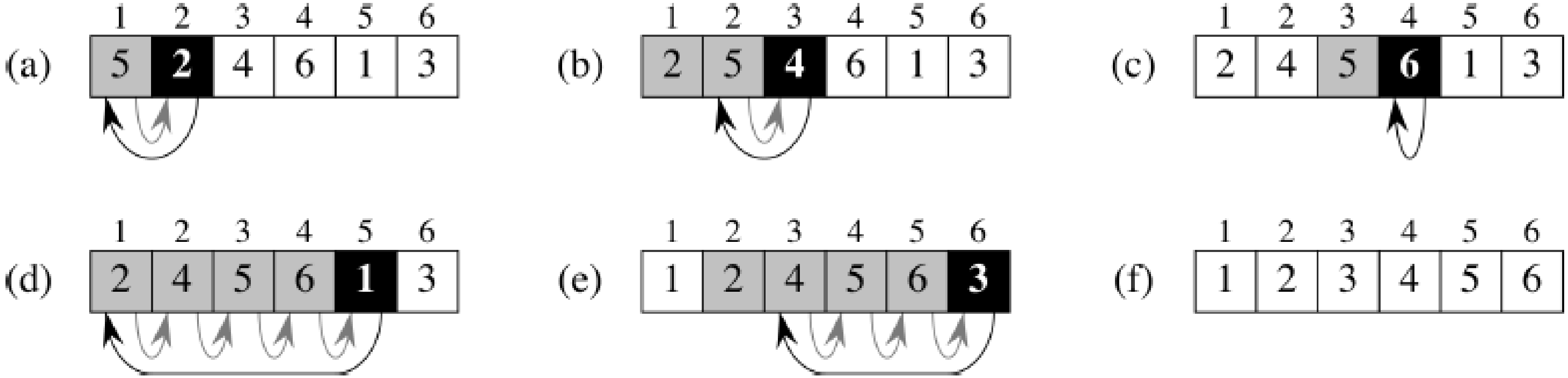
Insertion Sort



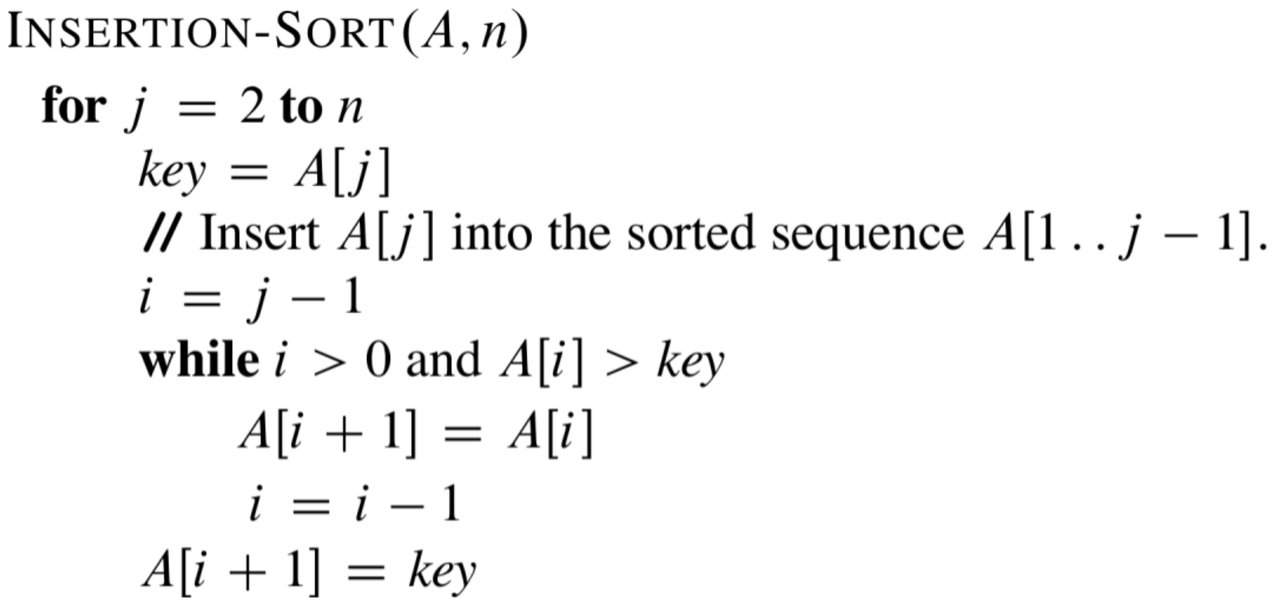
```
INSERTION-SORT$(A, n)$
  for $j = 2$ to $n$
      $key = A[j]$
      // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.
      $i = j - 1$
      while $i > 0$ and $A[i] > key$
          $A[i+1] = A[i]$
          $i = i - 1$
      $A[i+1] = key$
```

Example

- Sort A = < 5, 2, 4, 6, 1, 3 >



Correctness



```
INSERTION-SORT(A, n)
 for $j = 2$ to $n$
     $key = A[j]$
     // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.
     $i = j - 1$
     while $i > 0$ and $A[i] > key$
         $A[i+1] = A[i]$
         $i = i - 1$
     $A[i+1] = key$
```

- Loop invariant:
  At the start of each iteration of the for loop,
  the subarray A[1..j-1] consists of elements
  originally in A[1..j-1],
  but in sorted order.