CH08-320201
Algorithms and Data Structures

Prof. Dr. Michael Sedlmair

Jacobs University
Spring 2018

Overview today

0. Introduction
    - 0.1 Syllabus and Organization
    - 0.2 Goals
    - 0.3 Content
1. Foundations
    - 1.1 Definitions
    - 1.2 First example: Insertion Sort

0. Introduction

0.1 Syllabus and Organization

Online resources

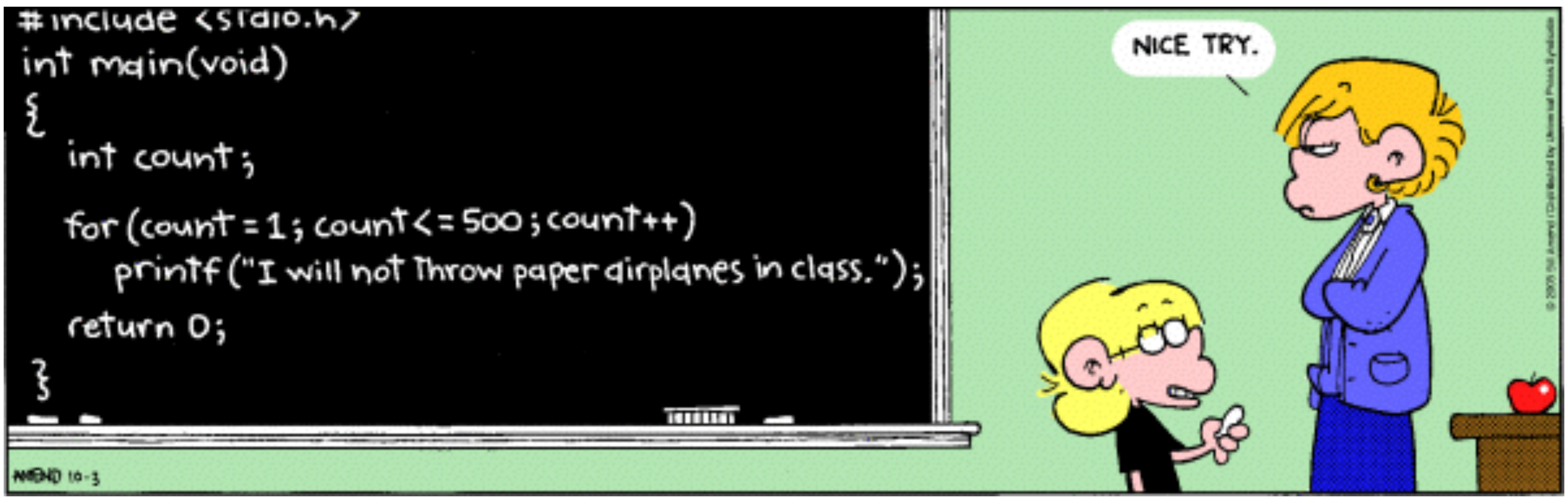
- Course Website
  - https://stevenabreu7.github.io/ads-website/
  - infos, slides, homework, etc.
  - NOTE: server will change —> stay tuned
- Moodle
  - https://moodle.jacobs-university.de/course/view.php?id=744
  - homework submissions, scores, announcements, forum, etc.
  - NOTE: “beta” version
    - you might not have access or even see the course yet
    - we will set up all the details over the next days/weeks
    - —> don’t worry / please give us some time

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

- C **or** C++ **or** python



Who “we” are:

- TAs:
  - Steven Abreu: s.abreu [@jacobs-university.de]
  - Oana Miron: o.miron [@jacobs-university.de]
  - Mohit Shrestha: mo.shrestha [@jacobs-university.de]
- Lecturer
  - Michael Sedlmair: old webpage / new webpage (coming soon)
  - Office: Res I, 168.
  - Phone: 3051
  - E-Mail: m.sedlmair [@jacobs-university.de]
  - Office hours: by appointment (usually Mon/Tue)

Lectures

- Times: Tuesdays
  - 08:15am — 09:30am **AND**
  - 09:45am — 11:00am
- Location: Research II Lecture Hall

Tutorials / Q&As

- Tutorials:
  - Time: Thursdays 11:15am — 12:30am
  - Location: Research II Lecture Hall
  - What: Discussing last homework & questions current content/homework (try to ask them already here!)
- Q&A:
  - Time: Weekend (tbd)
  - Where: Colleges (tbd)
  - What: questions current content/homework

Assignments

- Homeworks:
  - The homework assignments include theoretical and practical problems that tackle topics from the lectures.
  - The homework assignments are handed out on a regular basis (goal: in sync with the lectures).
- Submitting your homework
  - 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 Moodle.

Exams

- midterm — data and location tbd
- final exam — data and location tbd
- no quizzes

Grading

- Homework: 35%
- Midterm exam: 25%
- Final exam: 40%

Dates — Lectures

Date	Content
01.02. (Thu)	Foundations – definition, introductory example
06.02. (!!! 9:45am !!!)	Foundations – asymptotic analysis
13.02.	Foundations – divide & conquer as first concept,
	Foundations – recurrences
20.02.	Foundations – applying recurrences to divide & conquer
	Foundations – applying recurrences to divide & conquer
27.02.	Sorting & Searching – heap sort
	Sorting & Searching – heap sort
06.03.	Sorting & Searching – quick sort
	Sorting & Searching – randomized quick sort
13.03.	Sorting & Searching – lower bounds
	Sorting & Searching – sorting in linear time, searching
20.03.	Fundamental Data Structures – array, stack, queue,
	Fundamental Data Structures – lists, rooted tree, BST
27.03	Spring break

Dates — Lectures

03.04.	Fundamental Data Structures – red-black tree
	Fundamental Data Structures – red-black tree
10.04.	Fundamental Data Structures – hash tables
	Fundamental Data Structures – hash tables
17.04.	Design Concepts – dynamic programming
	Design Concepts – greedy algorithms
24.04.	Graph Algorithms – representations, graph searches
	Graph Algorithms – minimum spanning tree
01.05.	Holiday
08.05.	Graph Algorithms – shortest paths
	Graph Algorithms – maximum flow
15.05.	Computational Geometry – line intersections
	Computational Geometry – convex hull / Voronoi diagrams

Might slightly change

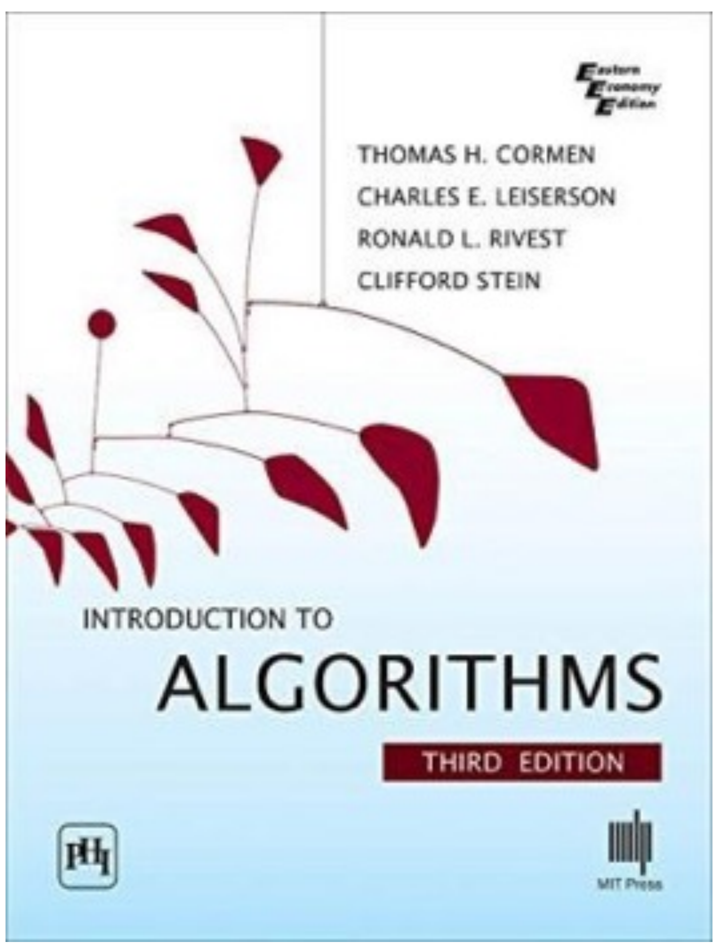
Dates — Assignments

Handout	Deadline	
06.02.	**12.02., 23:59**	Homework 1 (Foundations)
13.02.	**19.02., 23:59**	Homework 2 (Foundations)
20.02.	**26.02., 23:59**	Homework 3 (Foundations)
27.02.	**05.03., 23:59**	Homework 4 (Sorting & Searching)
06.03.	**12.03., 23:59**	Homework 5 (Sorting & Searching)
13.03.	**19.03., 23:59**	Homework 6 (Sorting & Searching)
20.03.	**02.04., 23:59**	Homework 7 (Fundamental Data Structures)
03.04.	**09.04., 23:59**	Homework 8 (Fundamental Data Structures)
10.04.	**16.04., 23:59**	Homework 9 (Fundamental Data Structures)
17.04.	**23.04., 23:59**	Homework 10 (Design Concepts)
24.04.	**07.05., 23:59**	Homework 11 (Graph Algorithms)
08.05.	**14.05., 23:59**	Homework 12 (Graph Algorithms)
not graded		Homework 13 (Computational Geometry)

Might slightly change

Literature

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



—> ebook in the library

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

- Foundations
- Sorting & Searching
- Fundamental Data Structures
- Design Concepts
- Graph Algorithms
- Computational Geometry

1. Foundations

1.1 Definitions

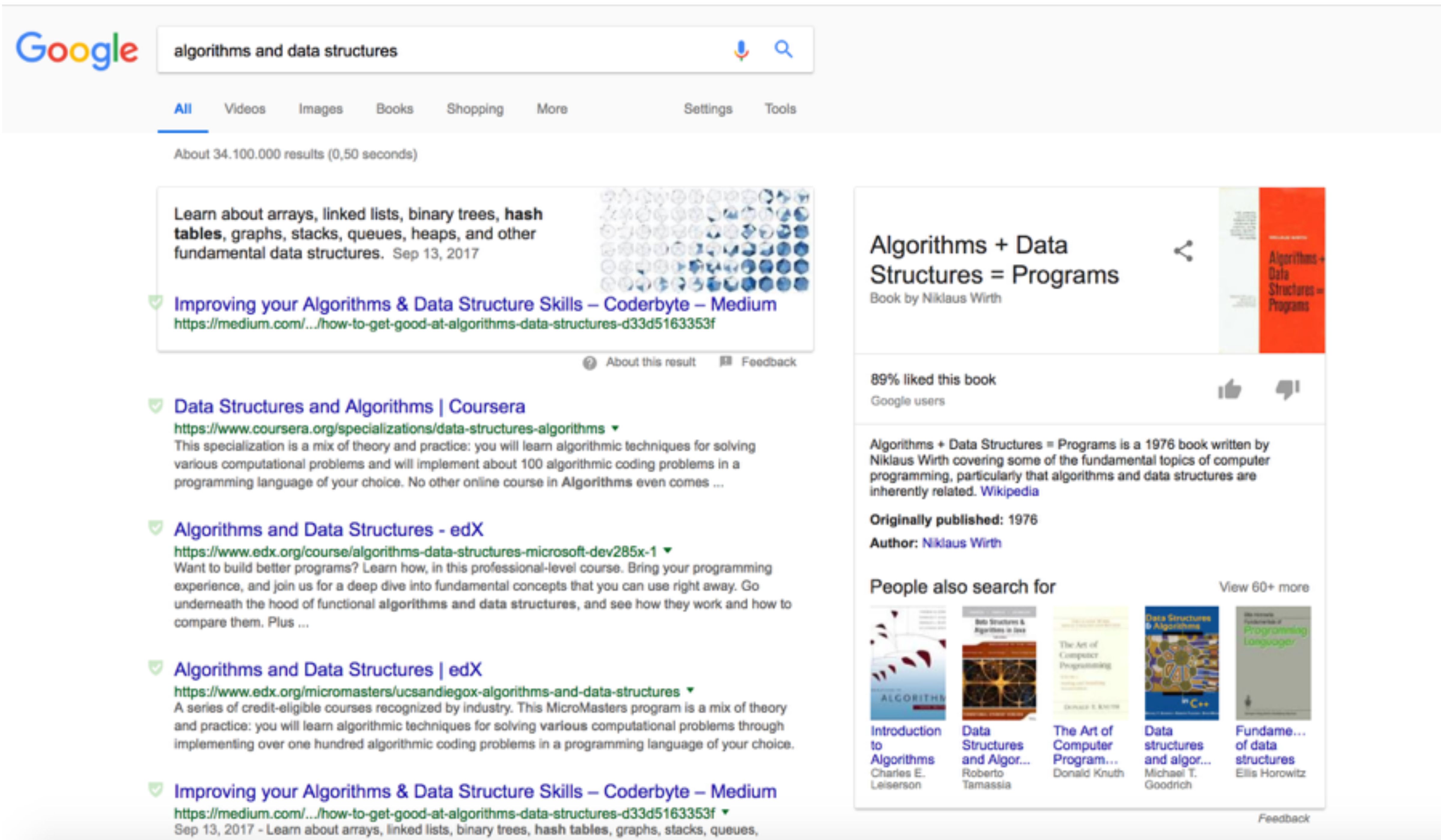
Definition: Algorithm

- 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 $\langle a_1, a_2, ..., a_n \rangle$ of numbers.
- Output
  permutation $\langle a'_1, a'_2, ..., a'_n \rangle$
  such that $a'_1 \leq a'_2 \leq ... \leq a'_n$
- Example (instance of sorting problem):
  Input: **8 2 4 9 3 6**
  Output: **2 3 4 6 8 9**

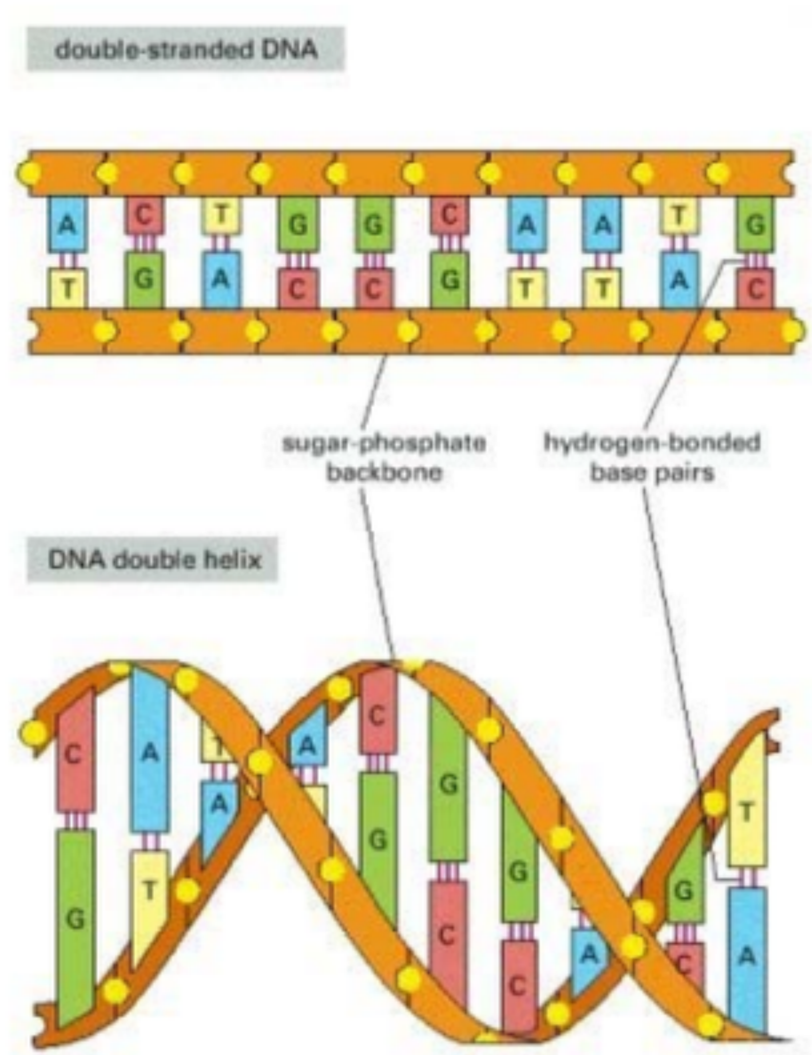
Example: Searching



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

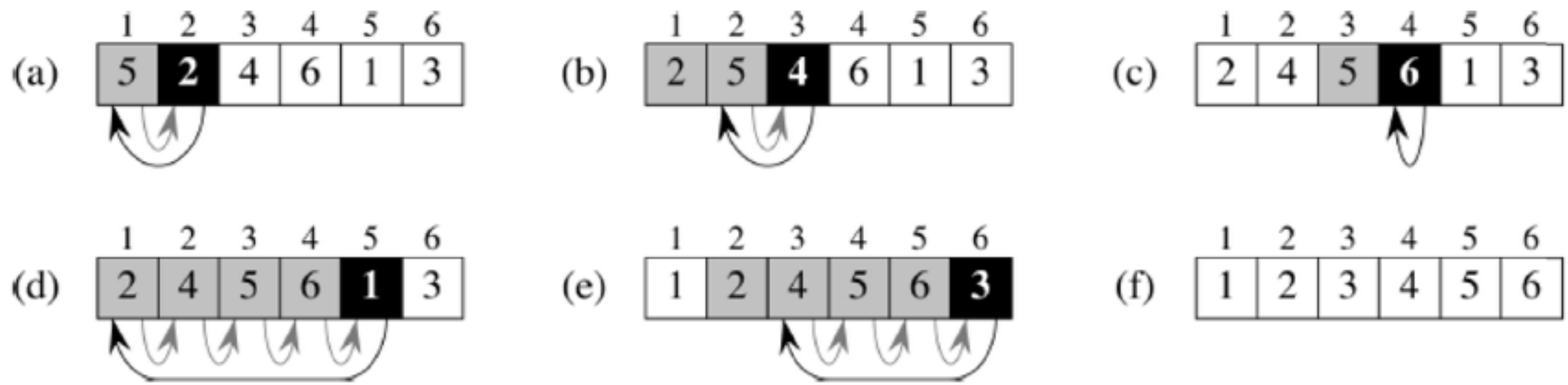
Insertion Sort

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 >




```
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
```

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.