CH08-320201

Algorithms and Data Structures

ADS

Lecture 1

Dr. Kinga Lipskoch

Spring 2019

Who am I?

- PhD in Computer Science at the Carl von Ossietzky University of Oldenburg
- University lecturer at the Computer Science Department
- Joined Jacobs University in January 2013
- Office: Research I, Room 94
- Telephone: +49 421 200-3148
- E-Mail: k.lipskoch@jacobs-university.de
- Office hours: Mondays 10:00 - 12:00

Agenda Today

- Introduction
  - Syllabus and Organization
  - Goals
  - Content
- Foundations
  - Definitions
  - First example: Insertion sort

Online Resources

- Course website
  https://grader.eecs.jacobs-university.de/courses/320201/2019_1/
- Slides and homework will be uploaded there
- Use Grader for homework submission (change semester to Spring 2019)

Teaching Assistants and Grading Criteria

- Oana Miron
- Milen Vitanov
- Abhik Pal
- Mohit Shrestha
- Benedikt Stock

- They will grade your homework
- Submit ZIP file containing one PDF file and source code files with makefile
- After 1st of March, 2019 5% deduction if no makefile
- Grading criteria
  https://grader.eecs.jacobs-university.de/courses/320201/2019_1/Grading_Criteria_ADS.pdf

Grader not Publicly Visible

- You can access Grader from campus without any additional connection or software
- To access Grader from outside of campus you need to use a VPN (Virtual Private Network) connection
- Tutorials from the Jacobs IRC IT team on how to install a VPN client:
  `https://www.youtube.com/user/jacobsircit`

Missing Homework, Quizzes, Exams according to AP

- https://www.jacobs-university.de/sites/default/files/bachelor_policies_v1.1.pdf (page 9)
- Illness must be documented with a sick certificate
- Sick certificates and documentation for personal emergencies must be submitted to the Student Records Office by the third calendar day
- Predated or backdated sick certificates will be accepted only when the visit to the physician precedes or follows the period of illness by no more than one calendar day
- Students must inform the Instructor of Record before the beginning of the examination or class/lab session that they will not be able to attend
- The day after the excuse ends, students must contact the Instructor of Record in order to clarify the make-up procedure
- Make-up examinations have to be taken and incomplete coursework has to be submitted by no later than the deadline for submitting incomplete coursework as published in the Academic Calendar

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 = method for analyzing a given algorithm's complexity, or how much of a resource, especially time or memory, it takes to execute
- Topics: fundamental data structures (lists, stacks, trees, hash tables), fundamental algorithms (sorting, searching, graph traversal)

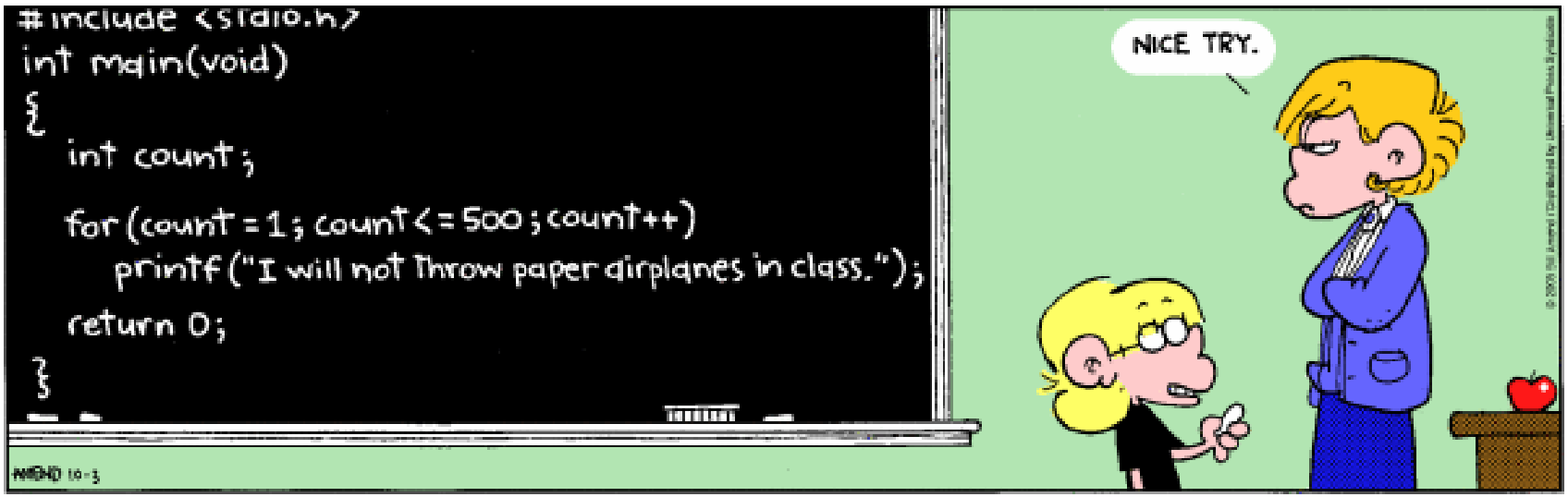
Objectives

Learn about:

- Fundamental algorithms for solving problems efficiently
- Basic algorithmic concepts
- Analysis of algorithms
- Fundamental data structures for efficiently storing, accessing, and modifying data

Requirements

Programming: freely choose between C or C++ or Python or Java if language is enforced by the problem statement



Lectures

- Time:
  - Tuesdays 9:45 – 11:00
  - Thursdays 11:15 – 12:30
- Location: Conrad Naber Hall, RLH

Tutorials

- Weekly tutorials given by one TA
- Tutorial before homework deadline
- West Hall 4, Thursdays, 19:00 – 21:00

Homework

- Homework
  - The homework assignments include theoretical and practical problems that tackle topics from the lectures
  - The homework assignments are handed out on a regular basis
- Submitting your homework
  - Extensions are possible only with an official excuse
  - Submit via Grader
    https://grader.eecs.jacobs-university.de/
- Homework deadline: Fridays, 23:00

Auditing the Course

- Here auditing is not just sitting in the lectures
- Have at least 45% as grade over all homework
- If less than 45% is reached no audit will be granted at the end of the semester

Exams

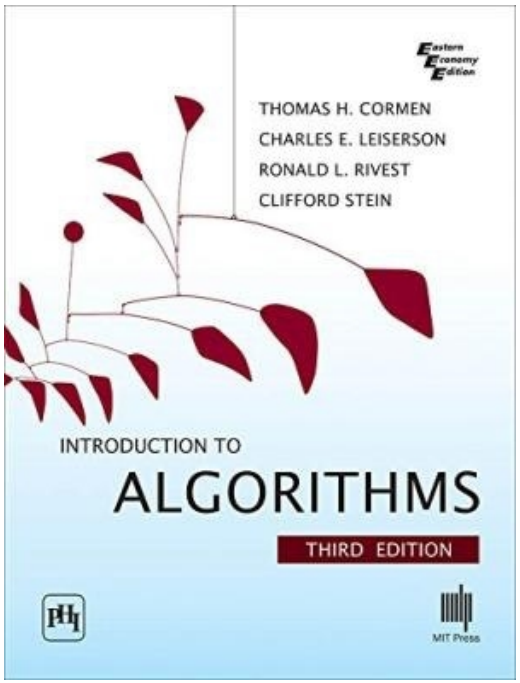
- Midterm exam
- Final exam
- No quizzes

Grading Components

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

Literature

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



Syllabus of Course

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

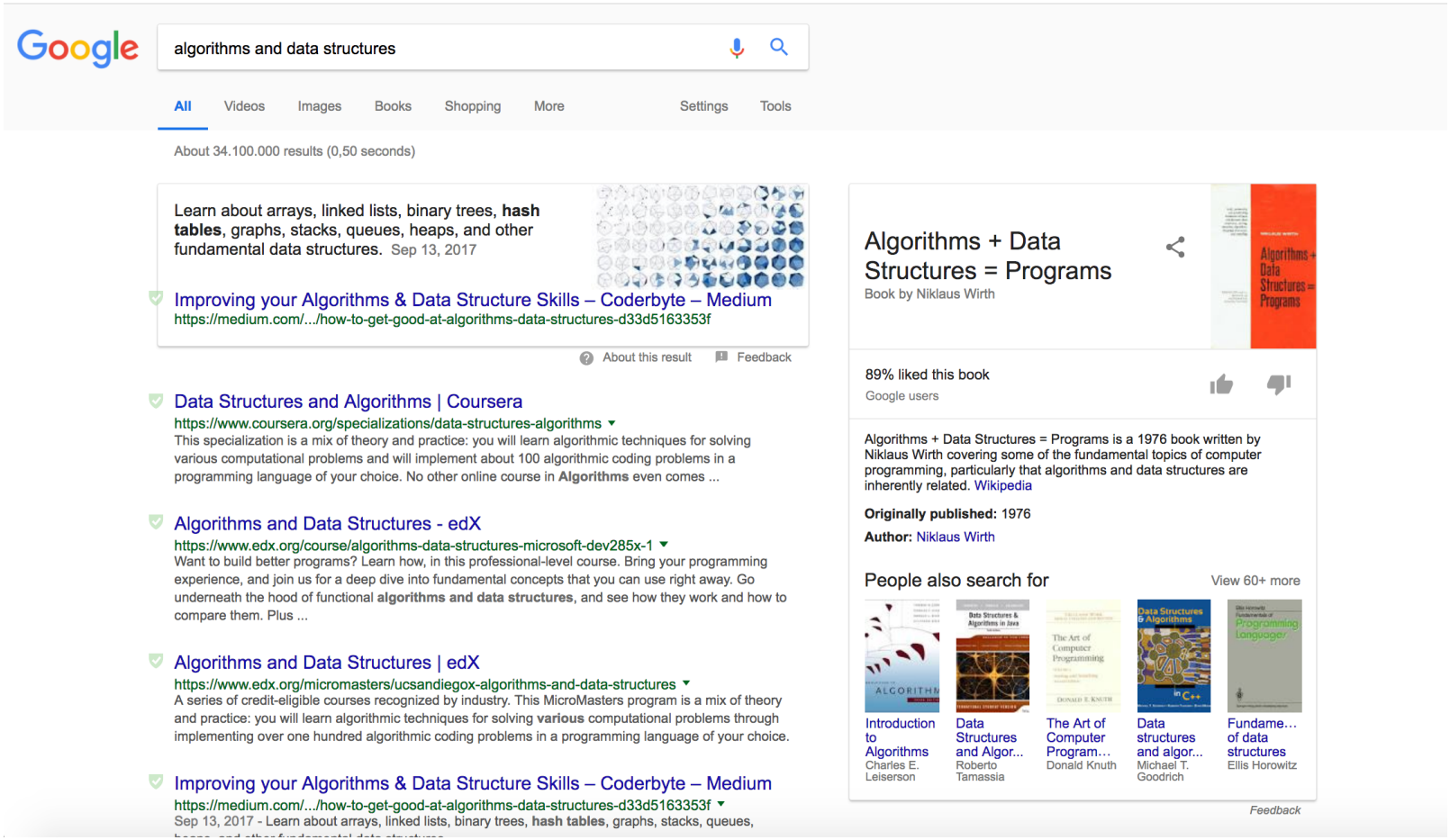
Definition: Algorithm

- 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

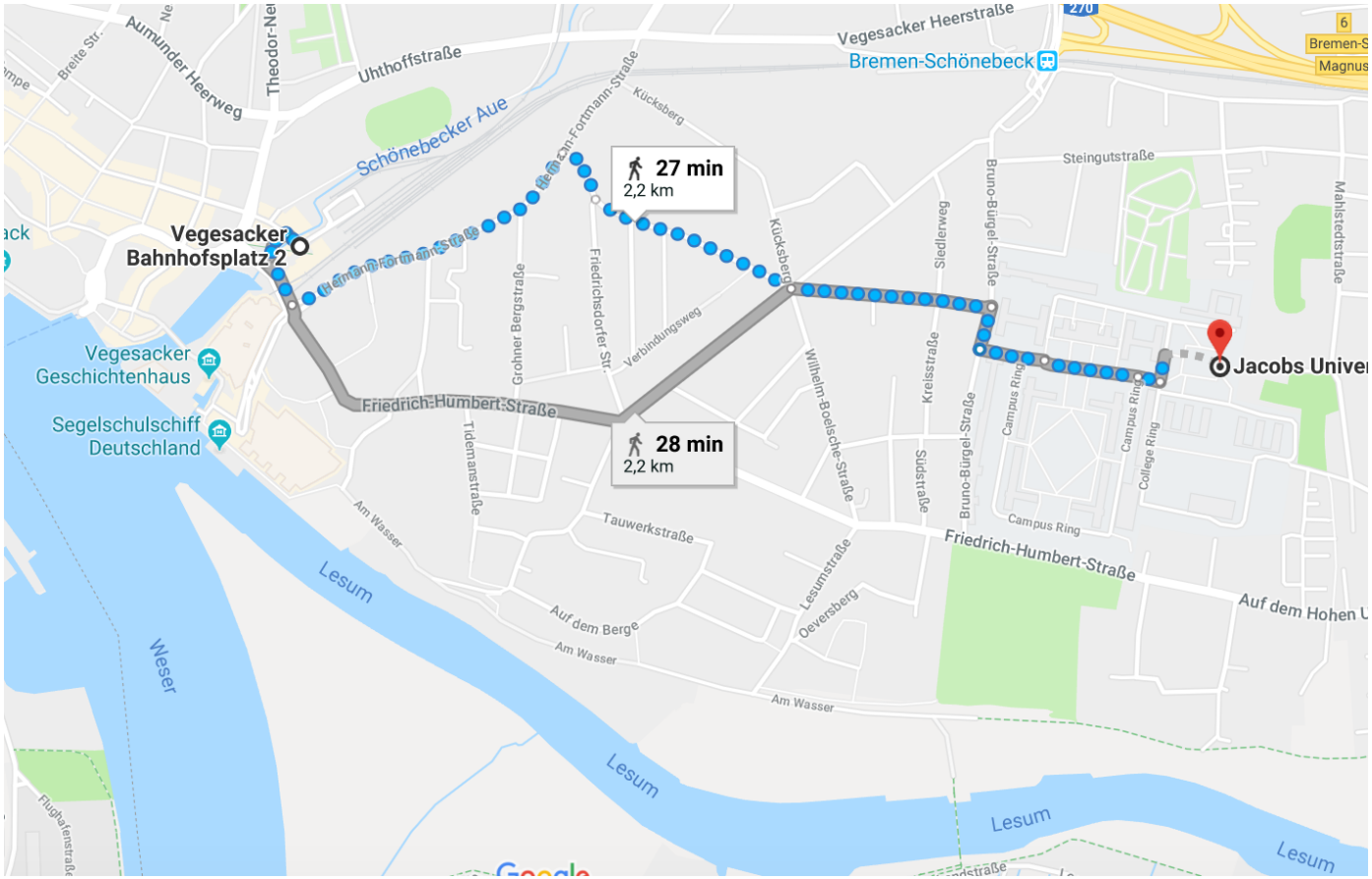
Example: Sorting Problem

- Input
  sequence $< a_1, a_2, ..., a_n >$ of numbers
- Output
  permutation $< a'_1, a'_2, ..., a'_n >$
  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

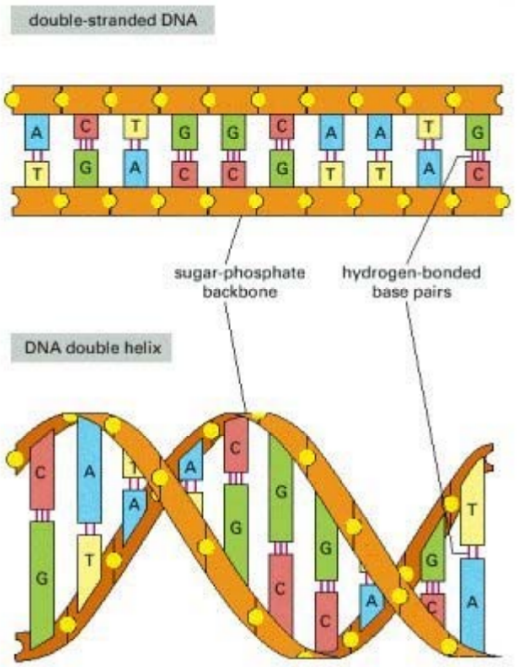


Example: Road map



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

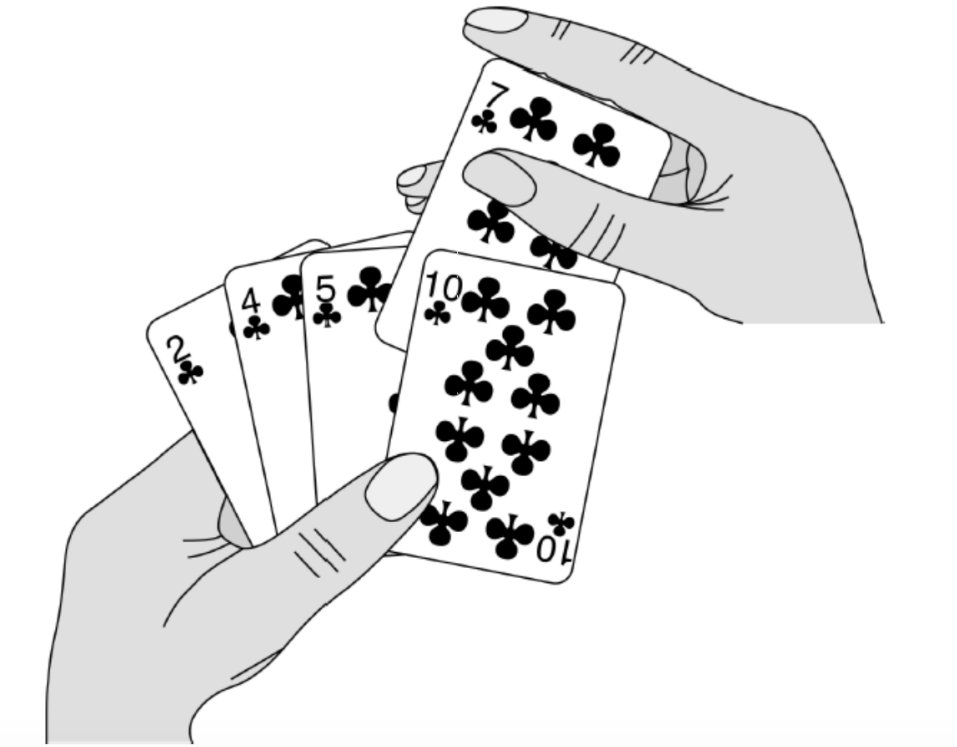
- Analysis helps us to understand scalability
- Performance often draws the line between what is feasible and what is impossible
- Algorithmic mathematics provides a language for talking about program behavior
- "Performance is the currency of computing"
- The lessons of program performance generalize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

Sorting Problem

First algorithm: Insertion sort

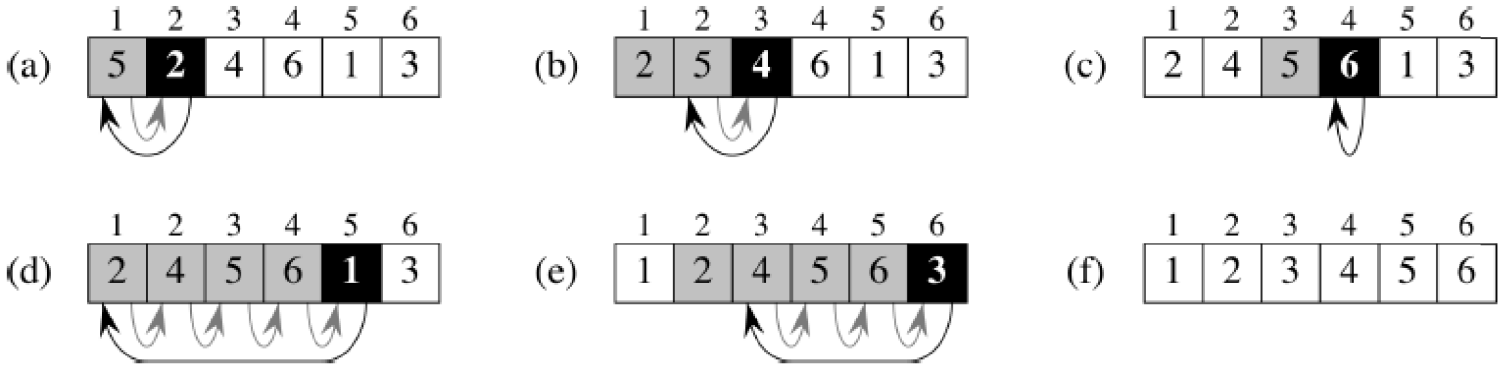


Insertion Sort

```
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
```

Insertion Sort: 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: is a property of a program loop that is true before (and after) each iteration

Example of 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.