

# Entwurf, Analyse und Umsetzung von Algorithmen

Runtime analysis Minsort / Heapsort, Induction

Prof. Dr. Rolf Backofen

Bioinformatics Group / Department of Computer Science

Entwurf, Analyse und Umsetzung von Algorithmen



**iems**  
intelligente eingebettete  
mikrosysteme

# Structure

## Algorithms and Data Structures

- Structure

- Links

- Organisation

  - Daphne

  - Forum

  - Checkstyle

  - Unit Tests

  - Version management

  - Jenkins

## Sorting

- Minsort

- Heapsort

# Algorithms and Data Structures

## Topics of this Lecture

### Topics of the Lecture:

- ▶ Algorithms and Data Structures  
Efficient data handling and processing  
... for problems that occur in practical **any** larger program / project
- ▶ **Algorithm**  $\hat{=}$  Solving of complex computational problems
- ▶ **Data structure**  $\hat{=}$  Representation of data on computer

## Example 1: Sorting

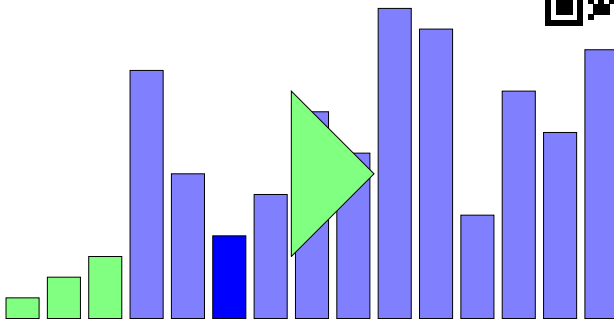


Figure: Sorting with *Minsort*

## Example 2: Navigation

- **Data structures:** How to represent the map as data?
- **Algorithms:** How to find the shortest / fastest way?

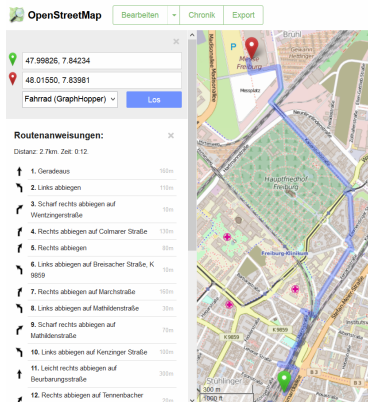




Figure: Navigationsplan  
© OpenStreetMap

# Example 3: Fault Tolerant Search



eyjafjallajökull

eyjafjallajökull - der unaussprechliche vulkanfilm

eyjafjallajökull film

eyjafjallajökull trailer

[Weitere Informationen](#)

## Ergebnisse für **eyjafjallajökull**

Stattdessen suchen nach: **ejafjatljökuk**

### Eyjafjallajökull – Wikipedia

[de.wikipedia.org/wiki/Eyjafjallajökull](https://de.wikipedia.org/wiki/Eyjafjallajökull) ▼

Der Name **Eyjafjallajökull** (isländisch für „Inselberge-Gletscher“) rührt von den so genannten Landeyjar (dt. Landinseln) her. Das sind felsige Erhebungen, ...

Name - Der Gletscher - Der Vulkan unter dem Gletscher - Eruptionsgeschichte





### Eyjafjallajökull - Der unaussprechliche Vulkanfilm Film 2014 ...

[www.kino.de](http://www.kino.de) > Filme ▼



31.07.2014 - **Eyjafjallajökull** - Der unaussprechliche Vulkanfilm, Irwitzige Komödie um ein verfeindetes Ex-Ehepaar, das wegen der Asche des isländischen ...

### Bilder zu eyjafjallajökull

[Unangemessene Bilder melden](#)



[Weitere Bilder zu eyjafjallajökull](#)



## Eyjafjallajökull

Gletscher in Island

Der Eyjafjallajökull, zu deutsch Eyjafjöll-Gletscher, ist der sechstgrößte Gletscher Islands. Er liegt an der äußersten Südküste, westlich des Gletschers Mýrdalsjökull in der Gemeinde Rangárbing eystra, die größte Höhe beträgt 1651 m.

[Wikipedia](#)

**Letzte Eruption:** April 2010

**Höhe:** 1.666 m

**Fläche:** 100 km²

**Prominenz:** 1.051 m

**Erstbesteiger:** Sveinn Pálsson

## Example 4: Protein Search

- Edit Distance: game changer in molecular biology

### Gapped **BLAST** and **PSI-BLAST**: a new generation of protein database search programs

[SF Altschul](#), [TL Madden](#), [AA Schäffer](#)... - Nucleic acids ..., 1997 - Oxford Univ Press

Abstract The **BLAST** programs are widely used tools for searching protein and DNA databases for sequence similarities. For protein comparisons, a variety of definitional, algorithmic and statistical refinements described here permits the execution time of the ...

Zitiert von: 55822 Ähnliche Artikel Alle 148 Versionen Zitieren Speichern

► [NCBI/BLAST/blastp suite](#) **Standard Protein BLAST**

[blastn](#) **[blastp](#)** [blastx](#) [tblastn](#) [tblastx](#)

Enter Query Sequence BLASTP programs search protein databases using a protein query. [more...](#)

Enter accession number(s), gi(s), or FASTA sequence(s) [Clear](#) [Query subrange](#)

From

To

MAKFASIIITLIFAALVLFPAFDAPANNZAQKLCCKPSGTVSGVCGNSNACINQCINLEGAKRGS  
CNYVFPARKICIVPC

Or, upload file [Bestand kiezen](#) [Geen bestand gekozen](#)

Job Title   
Enter a descriptive title for your BLAST search

☐ Align two or more sequences

Choose Search Set

Database [UniProtKB/Swiss-Prot\(swissprot\)](#)

Title: Non-redundant UniProtKB/SwissProt sequences.  
Molecule Type: Protein  
Update date: 2013/03/23  
Number of sequences: 454721

Organism Optional  ☐ Exclude [+](#)

Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown.

# Content of the Lecture 1 / 2

## General:

- ▶ Most of you had a lecture on basic programming ...  
performance was not an issue
- ▶ Here it is going to be:
  1. How fast is our program?
  2. How can we make it faster?
  3. How can we proof that it will always be that fast?
- ▶ **Important** issues:
  - ▶ Most of the time: application runtime
  - ▶ Sometimes also: resource / space consumption



# Content of the Lecture 2 / 2

## Algorithms:

- ▶ Sorting
- ▶ Dynamic Arrays
- ▶ Associative Arrays
- ▶ Hashing
- ▶ Edit distance
- ▶ Priority Queue
- ▶ Linked Lists
- ▶ Pathfinding / Dijkstra Algorithm
- ▶ Search Trees

## Mathematics:

- ▶ Runtime analysis
- ▶ Proof of correctness
- ▶  $\mathcal{O}$ -Notation

## After the lecture ...

- ... you should be able to understand the joke

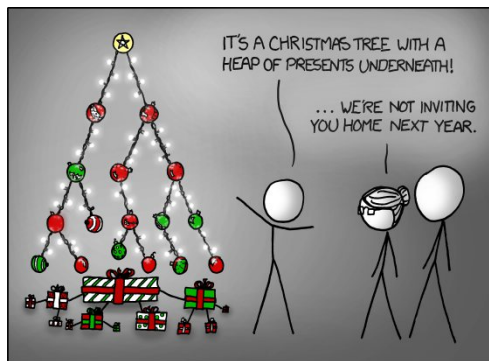
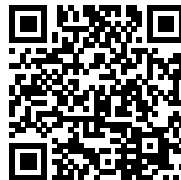


Figure: Comic © [xkcd/835](#)

- Hopefully your parents will still invite you

# Links



## Homepage:

- ▶ Exercise sheets
- ▶ Lectures
- ▶ Materials

Link to [Homepage](#)

## Lecture:

- ▶ Tuesday, 12:00 - 14:00, HS 00 006, Build. 082
- ▶ Recordings of the lecture will be uploaded to the webpage

## Exercises:

- ▶ One exercise sheet per week
- ▶ Submission / Correction / Assistance online
- ▶ Tutorial: (if needed)  
Wednesday, 13:00-14:00 - HS 00 006, Build. 082

## Exam:

- ▶ Planned: Sa. 23th March 2019, 10:00-12:00, Build. 101, Lec. theater 026 & 036

## Exercises:

- ▶ 80 % practical, 20 % theoretical
- ▶ We expect **everyone** to solve **every** exercise sheet

## Exam:

- ▶ 50 % of all points from the exercise sheets are needed
- ▶ Content of exam: whole lecture **and all exercises**

# Organisation - Exercises 3 / 5

## Exercises:

- ▶ Tutors: Tim Maffenbeier, Till Steinmann, Tobias Faller
- ▶ Coordinators: Michael Uhl, Florian Eggenhofer and Björn Grüning
- ▶ Deadline: ESE: 1 week, IEMS: none

# Organisation - Exercises 3 / 5

## Exercises:

- ▶ Post questions into the forum (link later)
- ▶ Submission via “commit” through svn and Daphne
- ▶ Feedback one week after deadline through “update” (svn)
- ▶ Unit test / checkstyle via Jenkins

# Organisation - Exercises 4 / 5

## Exercises - Points:

- ▶ Practical:
  - ▶ 60 % functionality
  - ▶ 20 % tests
  - ▶ 20 % documentation, Checkstyle, etc.
  - ▶ Program is not running  $\Rightarrow$  0 points
- ▶ Theoretical (mathematical proof):
  - ▶ 40 % general idea / approach
  - ▶ 60 % clean / complete



## Effort:

- ▶ 4 ECTS (ESE), 6 ECTS (IEMS)
- ▶ 120 / 180 working hours per semester
- ▶ 14 Lectures each 6 h / 8 h + exam
- ▶ 4 h / 6 h per exercise sheet (one per week)

# Daphne

## Daphne:

- ▶ Provides the following information:
  - ▶ Name / contact information of your tutor
  - ▶ Download of / info needed for exercise sheets
  - ▶ Collected points of all exercise sheets
  - ▶ Links to:
    1. Coding standards
    2. Build system
    3. The other systems
- ▶ Link: [Daphne](#)

## Forum:

- ▶ Please don't hesitate to ask if something is unclear
- ▶ Ask in the forum and not separate. Others might also be interested in the answer
- ▶ The [tutors](#) or the [coordinators](#) will reply as soon as possible
- ▶ Link: [Forum](#)

# Checkstyle

flake8

## Checkstyle / Linting (flake8):

- ▶ Installation: **python3** -m pip install flake8
- ▶ Check file: **python3** -m flake8 path/to/files/\*.py
- ▶ Link: [flake8](#)

# Unit Tests

## Why unit tests?

1. A non-trivial method without a unit test is probably wrong
2. Simplifies debugging
3. We and you can automatically check correctness of code

## What is a good unit test?

- ▶ Unit test checks desired output for a given input
- ▶ At least one **typical** input
- ▶ At least one **critical** case  
E.g. double occurrence of a value in sorting

# Unit Tests

## doctest

### Testing (doctest):

```
def subtract_one(n):  
    """Subtracts 1 from n
```

```
>>> subtract_one(5)  
4
```

```
>>> subtract_one(3)  
2  
"""
```

```
    return n-1
```

```
if __name__ == "__main__":  
    print("2 - 1 = %d" % subtract_one(2))
```

- ▶ Tests are contained in docstrings
- ▶ Module doctest runs them
- ▶ Run check with:  
**python3 -m doctest**  
*path/to/files/\*.py -v*

# Version management

## Subversion

### Version management (subversion):

- ▶ Keeps a history of code changes
- ▶ Initialize / update directory: **svn** checkout <URL>
- ▶ Add files / folders: **svn** add <file> --all
- ▶ Create snapshot: **svn** commit -m "<Your Message>"  
Data is uploaded to Jenkins automatically
- ▶ Link: [Subversion](#)

# Jenkins

## Jenkins:

- ▶ Provides our build system
- ▶ You can check if your uploaded code runs
  - ▶ Especially whether all **unit test** pass
  - ▶ And if **checkstyle** (flake8) is statisfied
- ▶ Will be shown in the first exercise
- ▶ Link: [Jenkins](#)



# Sorting 1 / 2

## Problem:

- ▶ Input:  $n$  elements  $x_1, \dots, x_n$
- ▶ Transitive operator “ $i$ ” which returns **true** if the left value is smaller than the right one
  - ▶ Transitivity:  $x < y, y < z \rightarrow x < z$
- ▶ Output:  $x_1, \dots, x_n$  sorted with operator

## Example

Input: 14, 4, 32, 19, 8, 44, 65

Output:

### Why do we need sorting?

- ▶ Nearly **every** program needs a sorting algorithm
- ▶ **Examples:**
  - ▶ Index of a search engine
  - ▶ Listing filesystem in explorer / finder
  - ▶ (Music) library
  - ▶ Highscore list

# Minsort - Algorithm

## Informal description:

- ▶ Find the minimum and switch the value with the **first** position
- ▶ Find the minimum and switch the value with the **second** position
- ▶ ...

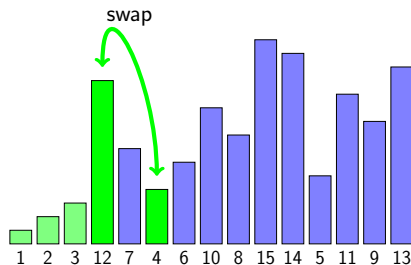


Figure: *Minsort*

# Minsort - Algorithm

## Minsort in Python:

```
def minsort(lst):  
    for i in range(0, len(lst)-1):  
        minimum = i  
  
        for j in range(i+1, len(lst)):  
            if lst[j] < lst[minimum]:  
                minimum = j  
  
        if minimum != i:  
            lst[i], lst[minimum] = \  
                lst[minimum], lst[i]  
  
    return lst
```

# Minsort - Runtime

## How long does our program run?

Table: Runtime for *Minsort*

$n$	Runtime / ms
$2 \times 10^3$	5.24
$4 \times 10^3$	16.92
$6 \times 10^3$	39.11
$8 \times 10^3$	67.80
$10 \times 10^3$	105.50
$12 \times 10^3$	150.38
$14 \times 10^3$	204.00
$16 \times 10^3$	265.98
$18 \times 10^3$	334.94

- ▶ We test it for different input sizes

- ▶ **Observation:**

It is going to be “disproportionately” slower the more numbers are being sorted

# Minsort - Runtime

## How long does our program run?

- ▶ We test it for different input sizes
- ▶ **Observation:**  
It is going to be “disproportionately” slower the more numbers are being sorted

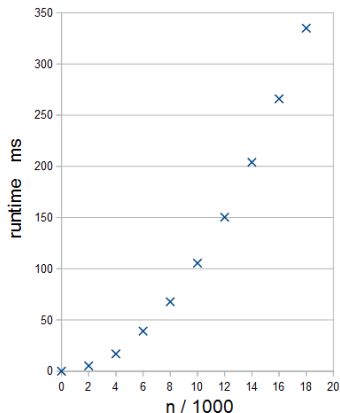


Figure: Runtime of *Minsort*

# Minsort - Runtime

## Runtime analysis:

- ▶ *Minsort* runtime depicted in a diagram
  - ▶ That is what you should do in the first exercise sheet
- ▶ **We observe:**
  - ▶ The runtime grows faster than linear
  - ▶ With double the input size we need four times the time

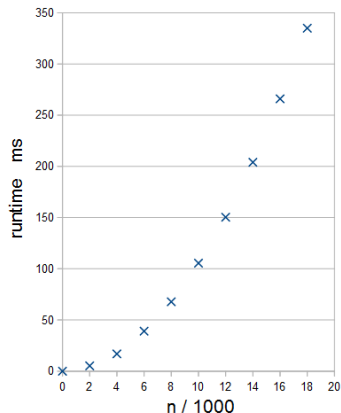


Figure: Runtime of *Minsort*

- ▶ Next lecture we will analyze deeper with other methods

# Heapsort - Algorithm 1 / 10

## Heapsort:

- ▶ The principle stays the same
- ▶ Better structure for finding the smallest element quicker

## Binary heap:

- ▶ Preferably a complete binary tree
- ▶ **Heap property:** Each child is **smaller** (larger) than the parent element



# Heapsort - Algorithm 2 / 10

## Min heap:

- ▶ **Heap property:** Each child is **smaller** (larger) than the parent element
- ▶ A valid heap fulfills the property at each node

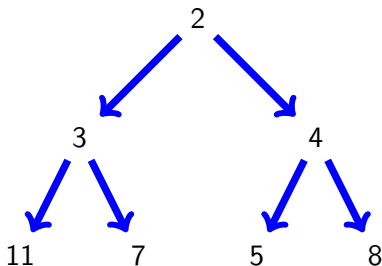


Figure: Valid min heap

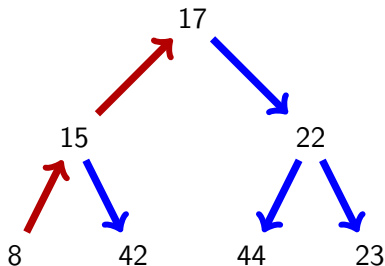


Figure: Invalid min heap

# Heapsort - Algorithm 3 / 10

## How to save the heap?

- ▶ We number all nodes from top to bottom and left to right starting at 0
  - ▶ The children of node  $i$  are  $2i + 1$  and  $2i + 2$
  - ▶ The parent node of node  $i$  is  $\text{floor}(\frac{i-1}{2})$

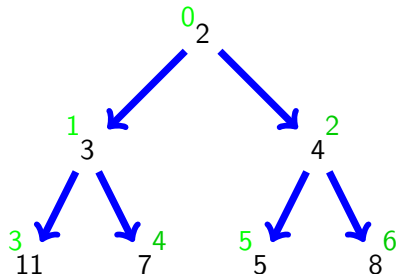


Table: Elements can be stored in array

0	1	2	3	4	5	6
2	3	4	11	7	5	8

Figure: Min heap

## Heapsort - Algorithm 4 / 10

### Repairing after taking the smallest element: `heap.pop()`

- ▶ Remove the smallest element (root node)
- ▶ Replace the root with the last node
- ▶ **Sift** the new root node down until the **heap property** is satisfied

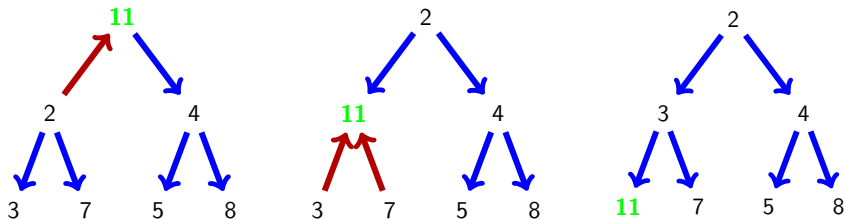


Figure: Repairing a min heap via sifting

# Heapsort - Algorithm 5 / 10

## Heapsort:

- ▶ Organize the  $n$  elements as heap
- ▶ While the heap still contains elements
  - ▶ Take the smallest element
  - ▶ Move the last node to the root
  - ▶ Repair the heap as described
- ▶ Output: 2, 3, ...

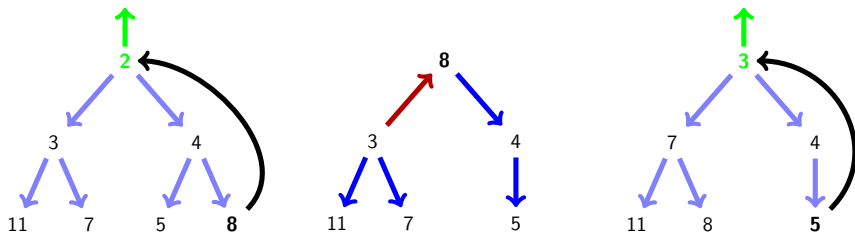


Figure: One iteration of Heapsort

# Heapsort - Algorithm 6 / 10

## Creating a heap:

- ▶ This operation is called **heapify**
- ▶ The  $n$  elements are already stored in an array
- ▶ Interpret the array as binary heap where the **heap property** is not yet satisfied
- ▶ We repair the heap from bottom up (in layers) with **sifting**

# Heapsort - Algorithm 7 / 10

Table: Input in array

0	1	2	3	4	5	6
11	7	8	3	2	5	4

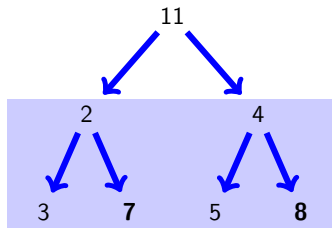
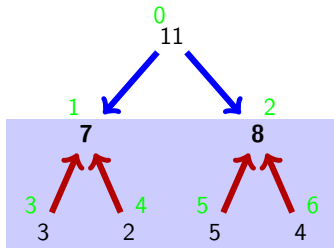


Figure: Heapify lower layer

## Heapsort - Algorithm 8 / 10

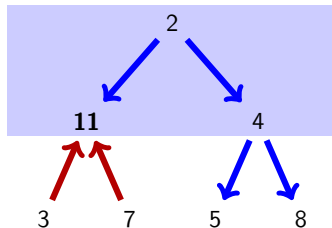
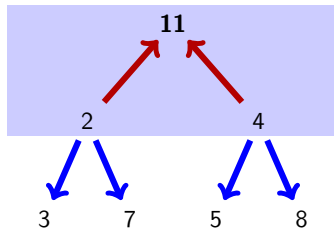


Figure: Heapify upper layer

## Heapsort - Algorithm 9 / 10

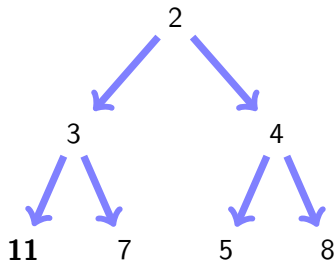


Figure: Resulting heap



# Heapsort - Algorithm 10 / 10

## Finding the minimum is intuitive:

- ▶ **Minsort:** Iterate through all non-sorted elements
- ▶ **Heapsort:** Finding the minimum is trivial (concept)

*Just take the root of the heap*

## Removing the minimum in Heapsort:

- ▶ Repair the heap and restore the **heap property**
  - ▶ We don't have to repair the whole heap
- ▶ More of this in the next lecture

# Further Literature

## ► Course literature

[CRL01] Thomas H. Cormen, Ronald L. Rivest, and Charles E. Leiserson.

*Introduction to Algorithms.*

MIT Press, Cambridge, Mass, 2001.

[MS08] Kurt Mehlhorn and Peter Sanders.

*Algorithms and Data Structures.*

Springer, Berlin, 2008.

<https://people.mpi-inf.mpg.de/~mehlhorn/ftp/Mehlhorn-Sanders-Toolbox.pdf>.

# Further Literature

## ► **Sorting**

[Wika] [Wikipedia - Heapsort](https://en.wikipedia.org/wiki/Heapsort)

`https://en.wikipedia.org/wiki/Heapsort`

[Wikb] [Wikipedia - Selectionsort](https://de.wikipedia.org/wiki/Selectionsort)

`https://de.wikipedia.org/wiki/Selectionsort`

# Further Literature

## ► Subversion

[Apa] [Apache Subversion](https://subversion.apache.org/)

`https://subversion.apache.org/`