ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

# Algorithm Theory

## Chapter 0 and 1a

**Christian Schindelhauer**

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08

CoNe Freiburg

IIF
INSTITUT FÜR
INFORMATIK
FREIBURG

Algorithms Theory

Chapter 0

# Introduction and Organization

# Thanks to

- **Prof. Thomas Ottmann and**
- **Prof. Susanne Albers for**
  - the great slides
  - the web recording using lecturnity
  - the allowance to use it
- **Feel free to use the German and English recordings**
  - except small exceptions (noted on the web pages) we follow the previous lectures

# Organization

- **Web page**
  - http://cone.informatik.uni-freiburg.de/teaching/vorlesung/algorithmentheorie-w08/
- **Forum**
  - registration for exercises
  - discussions, hints, critics, fun, etc.
- **Lectures**
  - Monday 2-4pm room 36 in 101
  - Thursday 2-3pm, room 36 in 101

# Lectures

- **Every week**
  - Monday, 2-4 pm, 101-036
  - Thursday, 2-3 pm, 101-036
- **Slides and Blackboard**
- **Contents matches old courses of**
  - Susanne Albers, Winter 2007/2008
  - Thomas Ottmann, Winter 2006/2007
- **plus some (small portion of) additional material**
  - will be noted on the web-page

# Exercise Groups

- **Group A: Alexander Schätzle (German)**
  - Tuesday, 9-10 am, room 051-00-034
- **Group B: Bente Luth (German)**
  - Wednesday, 2-3 pm, room 051-00-006
- **Group C: Stefan Rührup (English)**
  - Wednesday, 3-4 pm, room 051-00-006
- **Group D: Johannes Wendeberg (German)**
  - Friday, 10-11 am, room 051-00-006
- **Use forum to register**
  - even if you already registered with HIS

# Exercises

- **Appear weekly every Wednesday at**
  - `http://cone.informatik.uni-freiburg.de/teaching/vorlesung/algorithmentheorie-w08/exercise.html`
- **Solutions**
  - only single authored solutions
    - no points for the copier or copyist
  - must be submitted electronically until Monday noon
  - will be presented by the students at the exercises
    - extra point for presentation
  - best solutions will be rewarded by extra point
- **No mandatory requirements imposed by exercises**
  - It is ***HIGHLY RECOMMENDED*** to make exercises

# Bonus Points

- **Max 15 points**
- **1 point for each correctly solved exercise task**
  - submitted via e-mail until Monday 11:59:59 am
  - to algtheory08@informatik.uni-freiburg.de
  - with subject XX-Y-MMMMMMM Firstname Lastname
    - XX = sheet number
    - Y = group letter
    - MMMMMMMM = matriculation number
- **1 extra point for each correctly presented exercise task**
- **1 extra point for an excellent solution (one of the best)**

# Exam

- **Written exam**
- **Registration necessary in the online system for all**
  - bachelor and master students of (applied) computer science
- **8 parts**
  - 7 tasks @ 15 points
  - 15 bonus points from exercises
  - best 6 parts are chosen
  - ≥ 45 points are necessary to pass the exam
- **No prerequisites**

# Literature

- **Th. Ottmann, P. Widmayer:**
  - Algorithmen und Datenstrukturen
    4th Edition, Spektrum Akademischer Verlag,
    Heidelberg, 2002
- **Th. Cormen, C. Leiserson, R. Rivest, C. Stein:**
  - Introduction to Algorithms, Second Edition
    MIT Press, 2001
- **Original literature**
  - See also web pages

# Algorithms and Data Structures

- **Design and analysis techniques for algorithms**
  - Divide and conquer
  - Greedy approaches
  - Dynamic programming
  - Randomization
  - Amortized analysis

# Algorithms and Data Structures

- **Problems and application areas**
  - Geometric algorithms
  - Algebraic algorithms
  - Graph algorithms
  - Data structures
  - Internet algorithms
  - Optimization methods
  - Algorithms on strings

Algorithms Theory

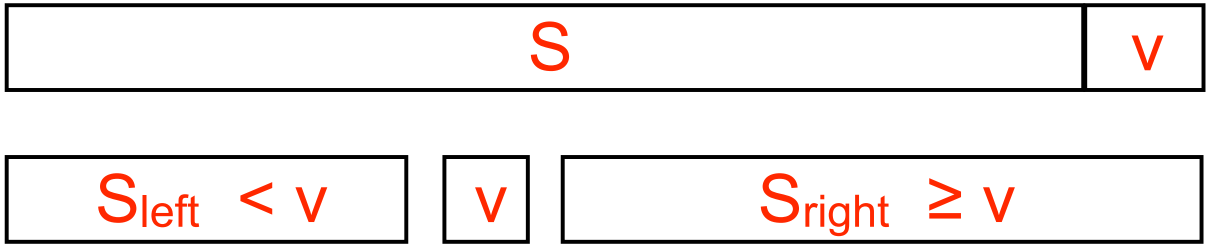
Chapter 1

# Divide and Conquer

# The Principle of Divide and Conquer

- **Remember Quicksort?**
- **Formulation and analysis of the principle**
- **Geometric Divide-and-Conquer**
  - Closest-Pair
  - Line segment intersection
  - Voronoi diagramm
- **Fast Fourier Transformation**

# Quicksort
# Sorting by Partitioning




```
function Quick (F : Folge) : Folge;
{returns the sorted sequence S}
begin
    if |S| ≤ 1 then Quick := F
    else { choose pivot element v in S;
              partition S in S_left with all elements < v
              and S_right with elements ≥ v
              Quick := Quick(S_left) v  Quick(S_right)  }
end;
```

# Divide and Conquer Paradigm

**Divide-and-conquer method for solving a problem of size $n$**

**1. Divide:**

**$n > c$: Divide the problem into $k$ sub-problems of sizes $n_1,\ldots,n_k$ ($k \geq 2$)**

**$n \leq c$: Use direct solution**

**2. Conquer:**

**Recursively solve the $k$ sub-problems (using Divide and Conquer)**

**3. Merge:**

**Combine the $k$ partial solutions to get the overall solution**

# Analysis

*T(n)*: **maximal number of steps necessary for solving an instance of size n**

$$T(n) = \begin{cases} a & n \leq c \\ T(n_1) + \ldots + T(n_k) & n > c \\ + \text{ cost for divide and merge} & \end{cases}$$

**special case:** $k = 2$, $n_1 = n_2 = n/2$

**cost for divide and merge** : $DM(n)$

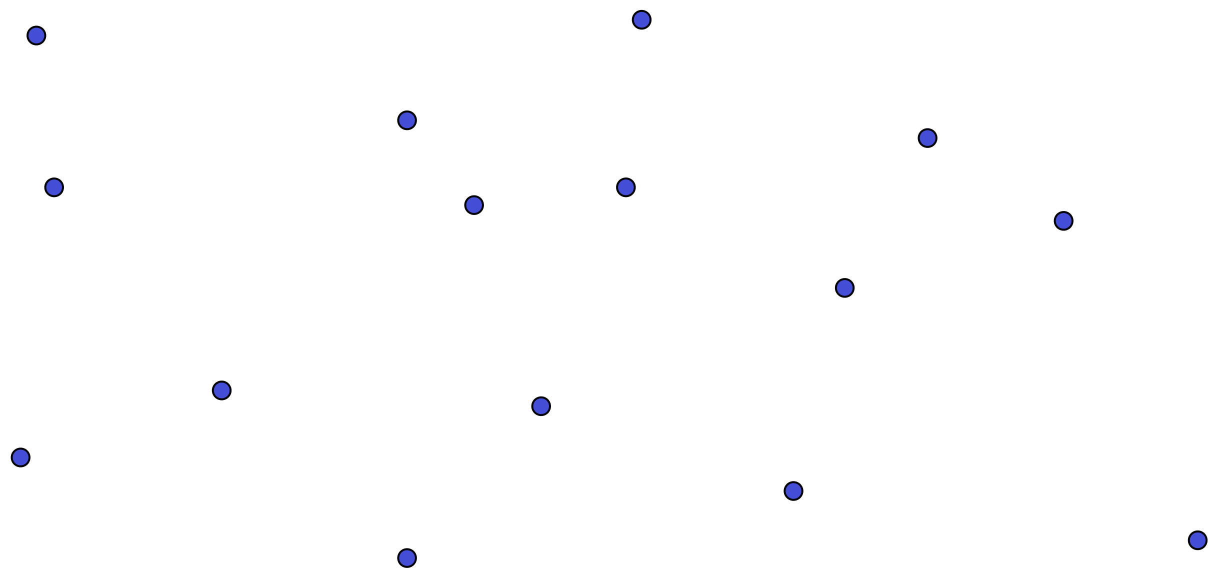
$T(1) = a$

$T(n) = 2 \cdot T(n/2) + DM(n)$

# Geometric Divide-and-Conquer

**Closest Pair Problem:**

**Given a set S of n points, find a pair of points with the smallest distance**



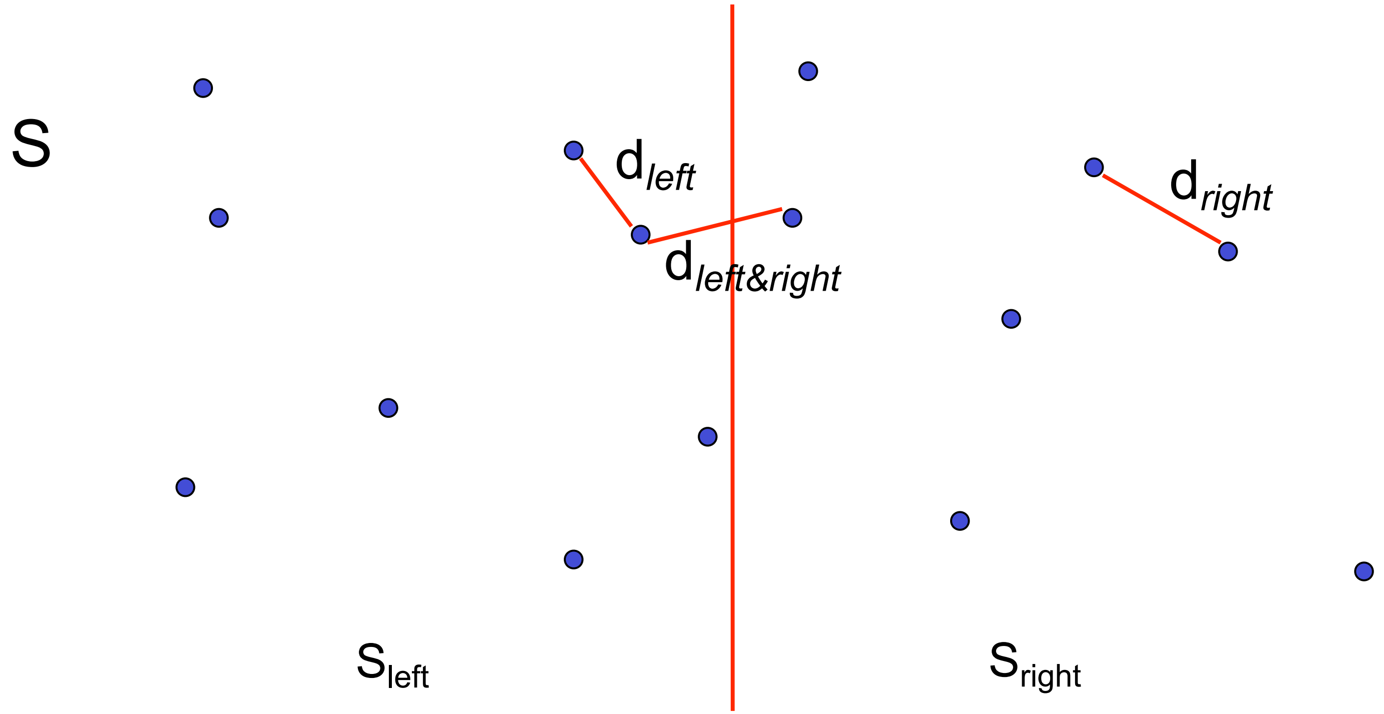
# Divide-and-Conquer Method

**1. Divide** : Divide S into two equally sized sets $S_{left}$ and $S_{right}$

**2. Conquer**: $d_{left} = \text{mindist}(S_{left})$ $d_{right} = \text{mindist}(S_{right})$

**3. Merge**: $d_{left\&right} = \min \{d(p_{left}, p_{right}) \mid p_{left} \in S_{left}, p_r \in S_{right}\}$

return $\min \{d_{left}, d_{right}, d_{left\&right}\}$



# Divide-and-Conquer Method

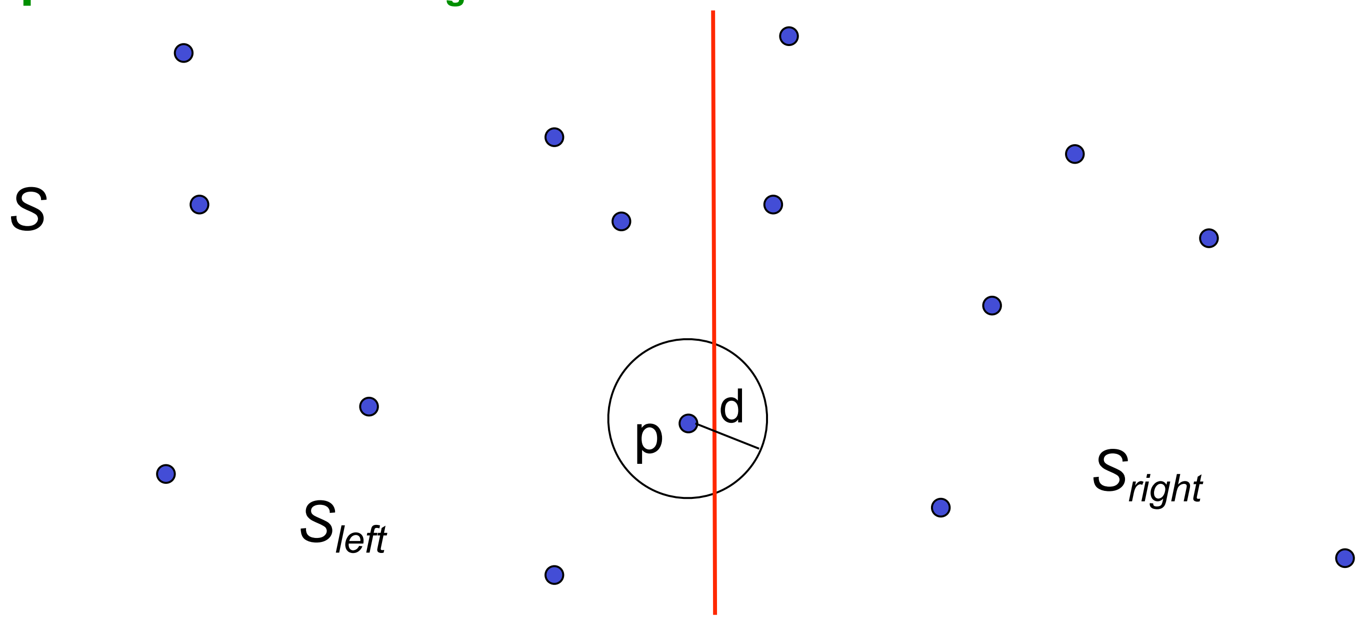
**1. Divide** : Divide S into two equally sized sets $S_{left}$ and $S_{right}$

**2. Conquer**: $d_{left}$ = mindist($S_{left}$) $d_{right}$ = mindist($S_{right}$)

**3. Merge**: $d_{left\&right} = \min \{d(p_{left}, p_{right}) \mid p_{left} \in S_{left}, p_r \in S_{right}\}$

return $\min \{d_{left}, d_{right}, d_{left\&right}\}$

**Computation of $d_{left\&right}$:**

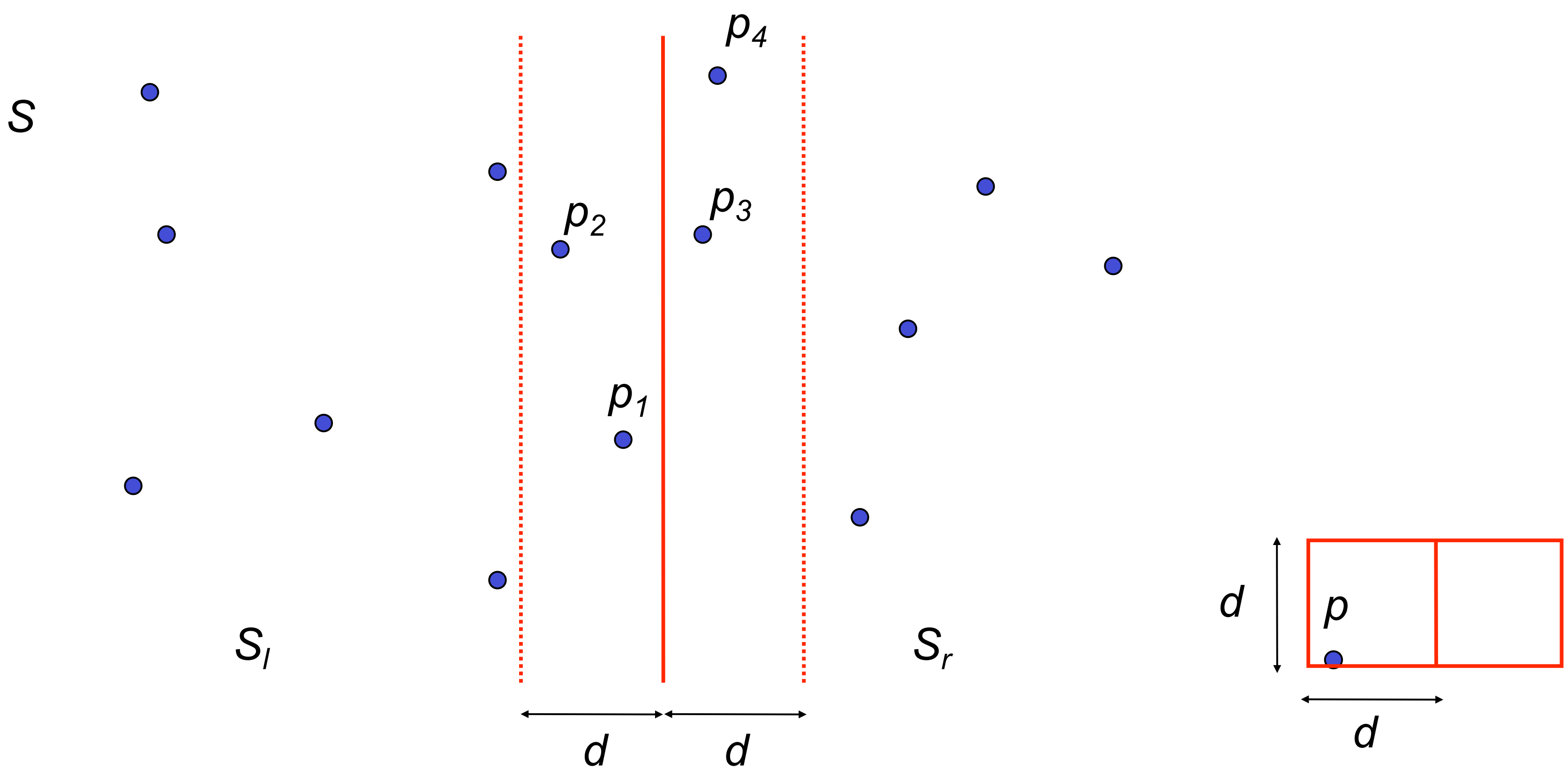



$d = \min \{d_{left}, d_{right}\}$

# Merge Step

- **Consider only points within distance d of the bisection line vertically ordered**
  - create an ordered list with increasing y-coordinates
- **For each point p consider all points q within vertical (y-) distance of at most d**
  - in the ordered lists these points are among the next 7 points

# Merge Step




$d = \min\{d_{left}, d_{right}\}$

# Implementation

- **Sort all points of S with respect to x-coordinates**
  - runtime: $O(n \log n)$
- **Sort all points of S with respect to y-coordinates**
  - runtime: $O(n \log n)$
- **Create sorted x- and y-coordinate lists for both sub-problems**
  - runtime: $O(n)$
- **After solving sub-problems in $S_{left}$ , $S_{right}$ create a sorted list of points in $S$ within distance d of the separation line with increasing y-coordinates**
  - use original sorted list according y-coordinates and erase far nodes
  - runtime: $O(n)$

# Running Time
# Divide-and-Conquer

$$T(n) = \begin{cases} 2T(n/2) + an & n > 3 \\ a & n \leq 3 \end{cases}$$

- **Guess solution by repeated substitution**
- **Verify by induction**

**Solution:** *O(n log n)*

ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

# Algorithm Theory

## end of lecture


**Christian Schindelhauer**

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik


Wintersemester 2007/08

CoNe Freiburg

IIF
INSTITUT FÜR
INFORMATIK
FREIBURG