Computational Geometry
Chapter 3: Closest Pair

Prof. Dr. Sándor Fekete

Algorithms Division
Department of Computer Science
TU Braunschweig

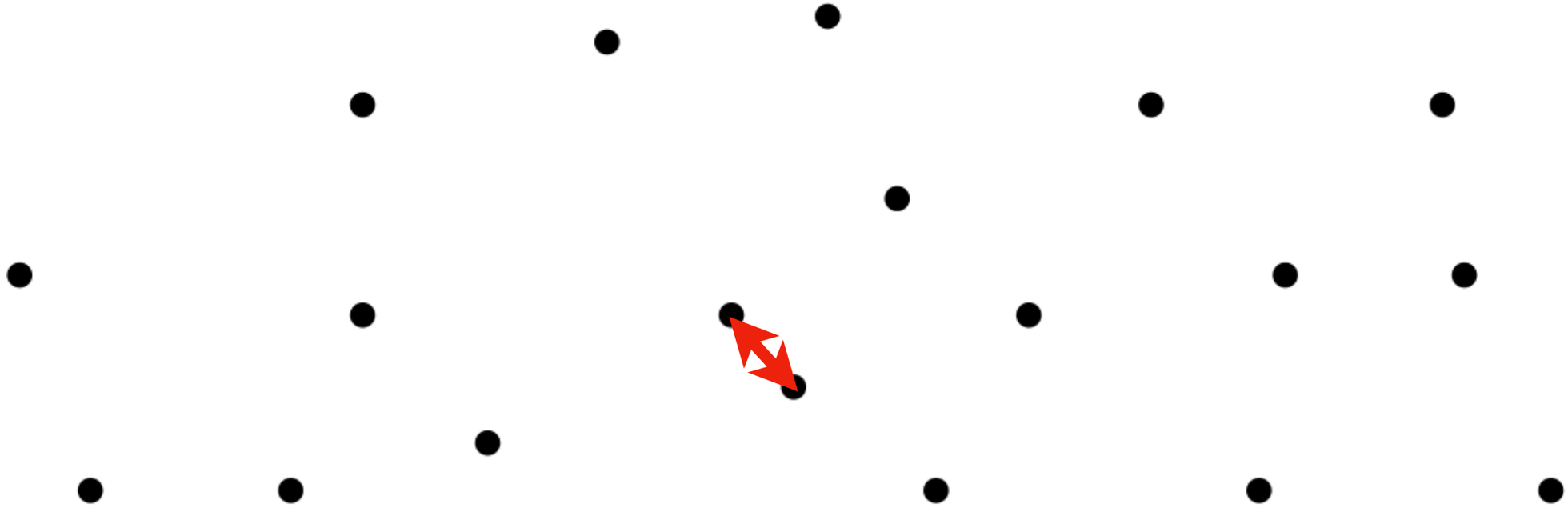
Technische Universität Braunschweig

1. Introduction
2. Prelude: Solving recursions
3. Lower bounds
4. Divide-and-conquer
5. Randomized incremental construction

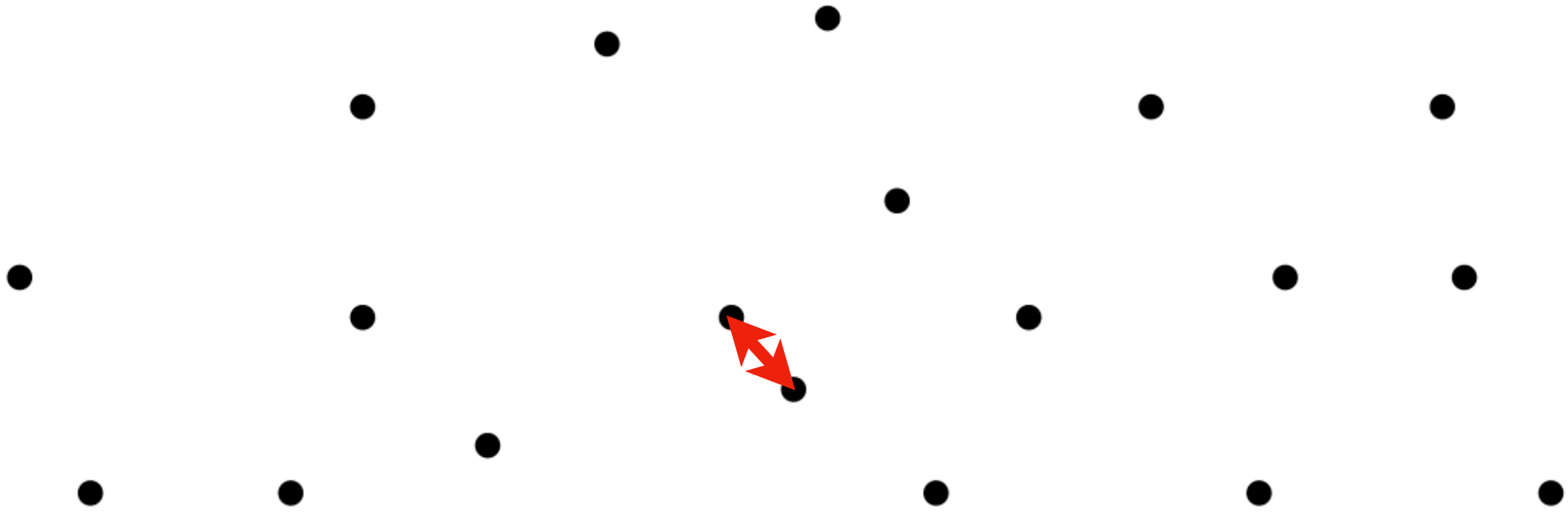
Problem 3.1, „Closest Pair“:

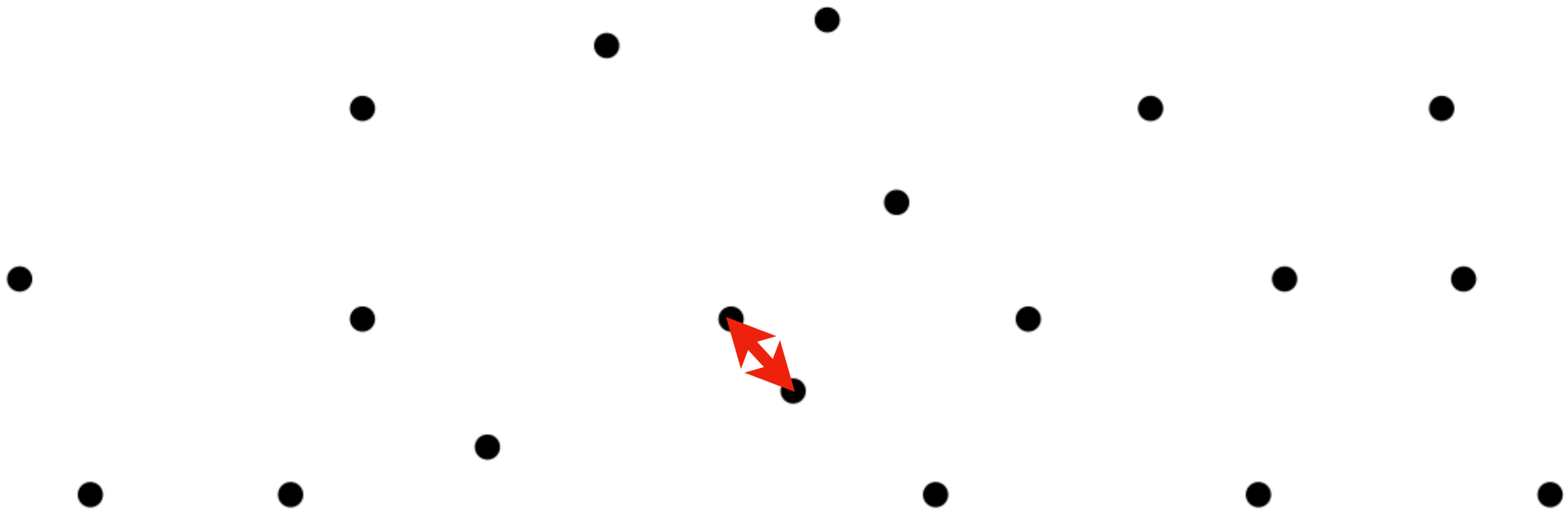
- Given: A finite point set $\mathcal{P} := \{p_0, \ldots, p_{n-1}\} \subset \mathbb{R}^d$
- Wanted: A pair with minimal distance wrt to metric d :

$$(p, q) \in \mathcal{P} \times \mathcal{P} \text{ with } d(p, q) = \min\{d(x, y) \mid x, y \in \mathcal{P}, x \neq y\}$$



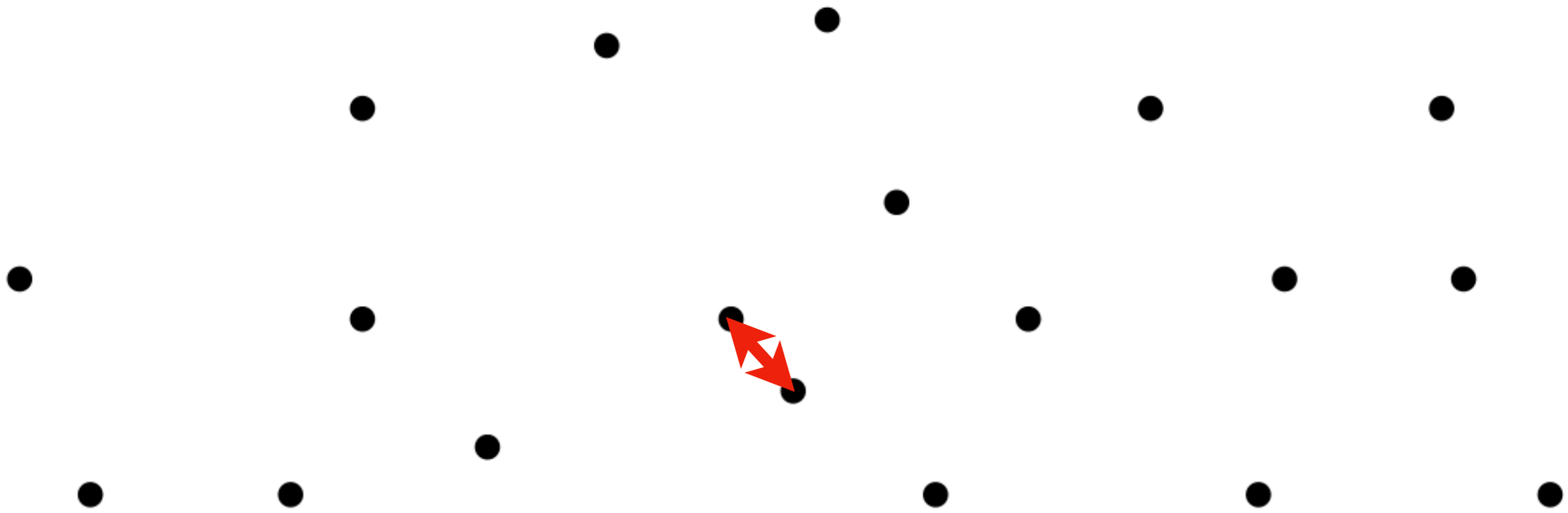
Definitions - I





Definition 3.2

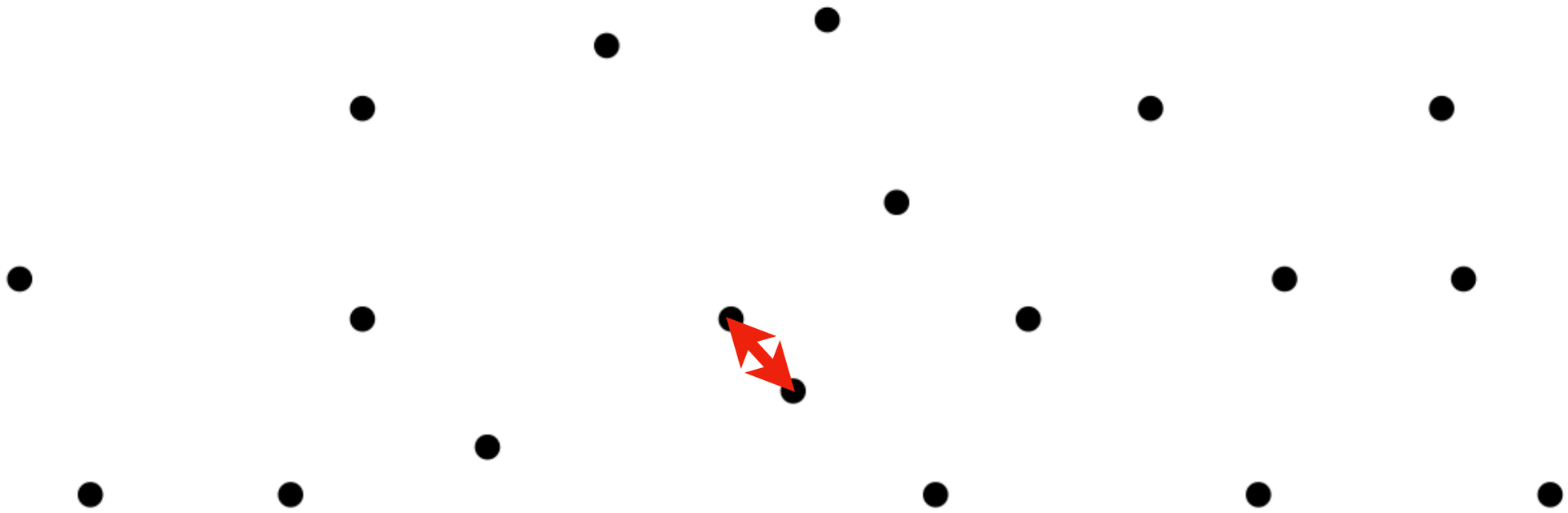
A metric is a mapping $d : M \times M \to \mathbb{R}_{\geq 0}$ with



Definition 3.2

A metric is a mapping $d : M \times M \to \mathbb{R}_{\geq 0}$ with

$1. \forall x, y \in M : d(x, y) = 0 \Leftrightarrow x = y$ (Definitness)

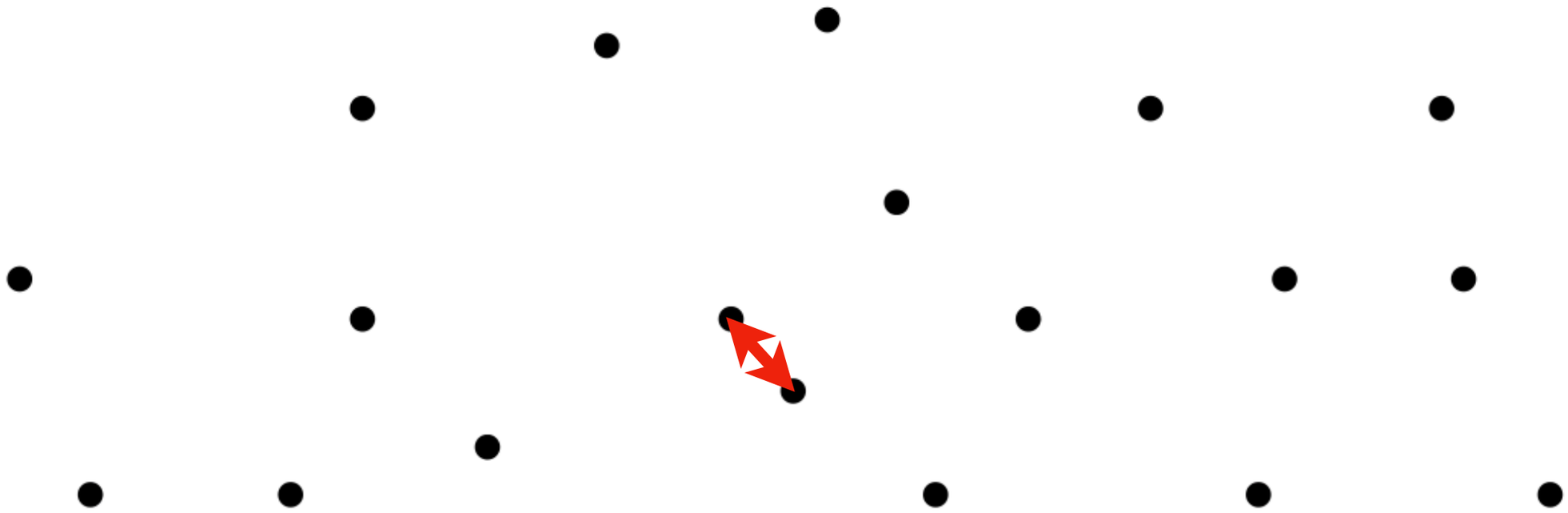


Definition 3.2

A metric is a mapping $d : M \times M \to \mathbb{R}_{\geq 0}$ with

$1. \forall x, y \in M : d(x, y) = 0 \Leftrightarrow x = y$ (Definitness)

$2. \forall x, y \in M : d(x, y) = d(y, x)$ (Symmetry)



Definition 3.2

A metric is a mapping $d : M \times M \to \mathbb{R}_{\geq 0}$ with

$1. \forall x, y \in M : d(x, y) = 0 \Leftrightarrow x = y$ (Definitness)

$2. \forall x, y \in M : d(x, y) = d(y, x)$ (Symmetry)

$3. \forall x, y, z \in M : d(x, z) \leq d(x, y) + d(y, z)$ (Triangle inequality)

Examples for $p, q \in \mathbb{R}^d$:

- Euclidean metric:

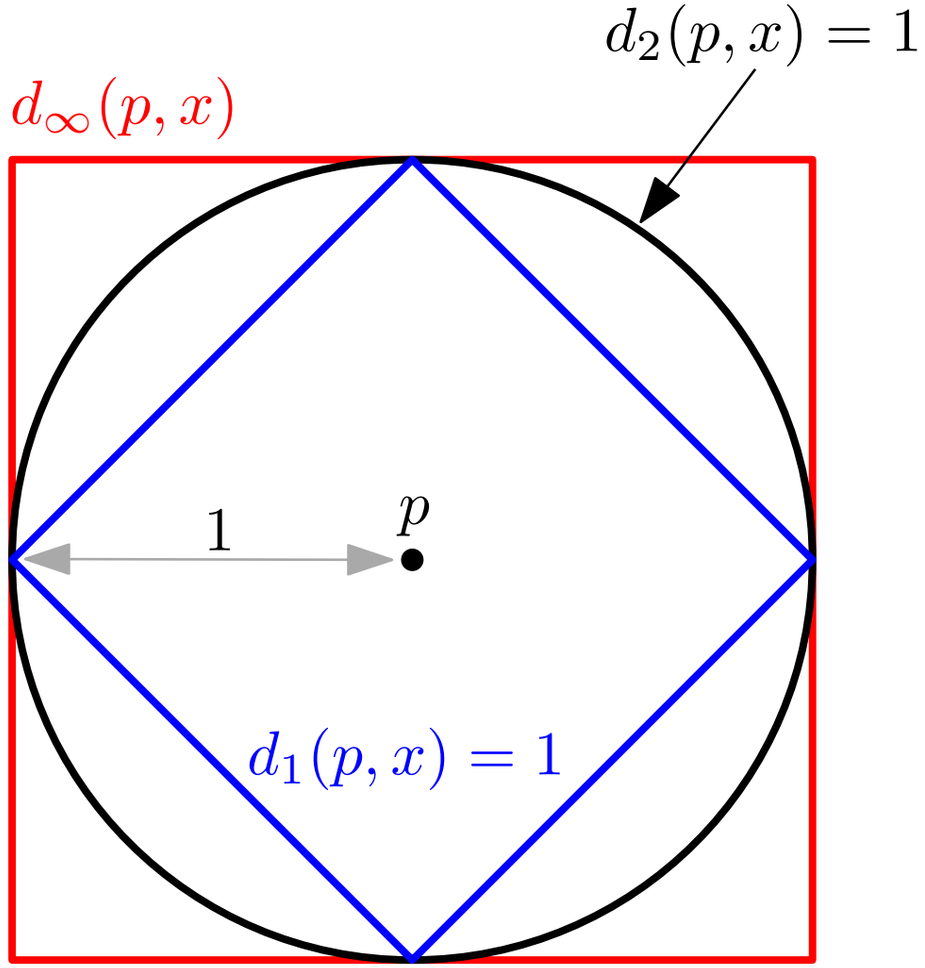
$$d_2(p,q) := \sqrt{\sum_{i=1}^{d} (p.x_i - q.x_i)^2}$$

- Manhattan metric:

$$d_1(p,q) := \sum_{i=1}^{d} |p.x_i - q.x_i|$$

- Maximum metric:

$$d_\infty(p,q) := \max_{i=1,\ldots,d} |p.x_i - q.x_i|$$



More generally:

- p-metric $(p \geq 1)$: $d_p(u,q) := \left(\sum_{i=1}^{p} (u.x_i - q.x_i)^p \right)^{1/p}$
- $\forall p, q \in \mathbb{R}^d, i, j \geq 1 : i \leq j \Rightarrow d_i(p,q) \geq d_j(p,q)$

Superhero!

Superhero!

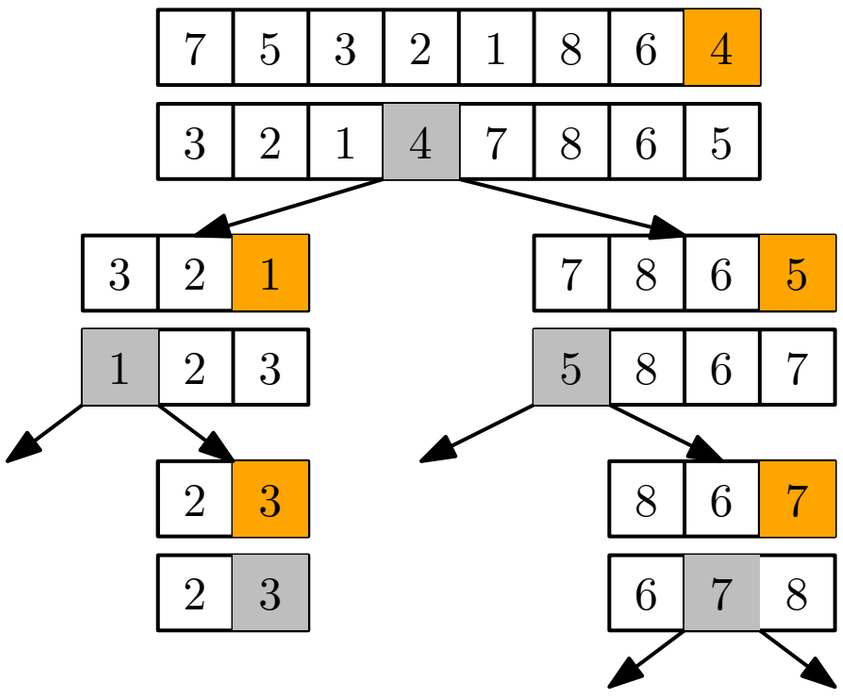
myshapers.com

myshapers.com

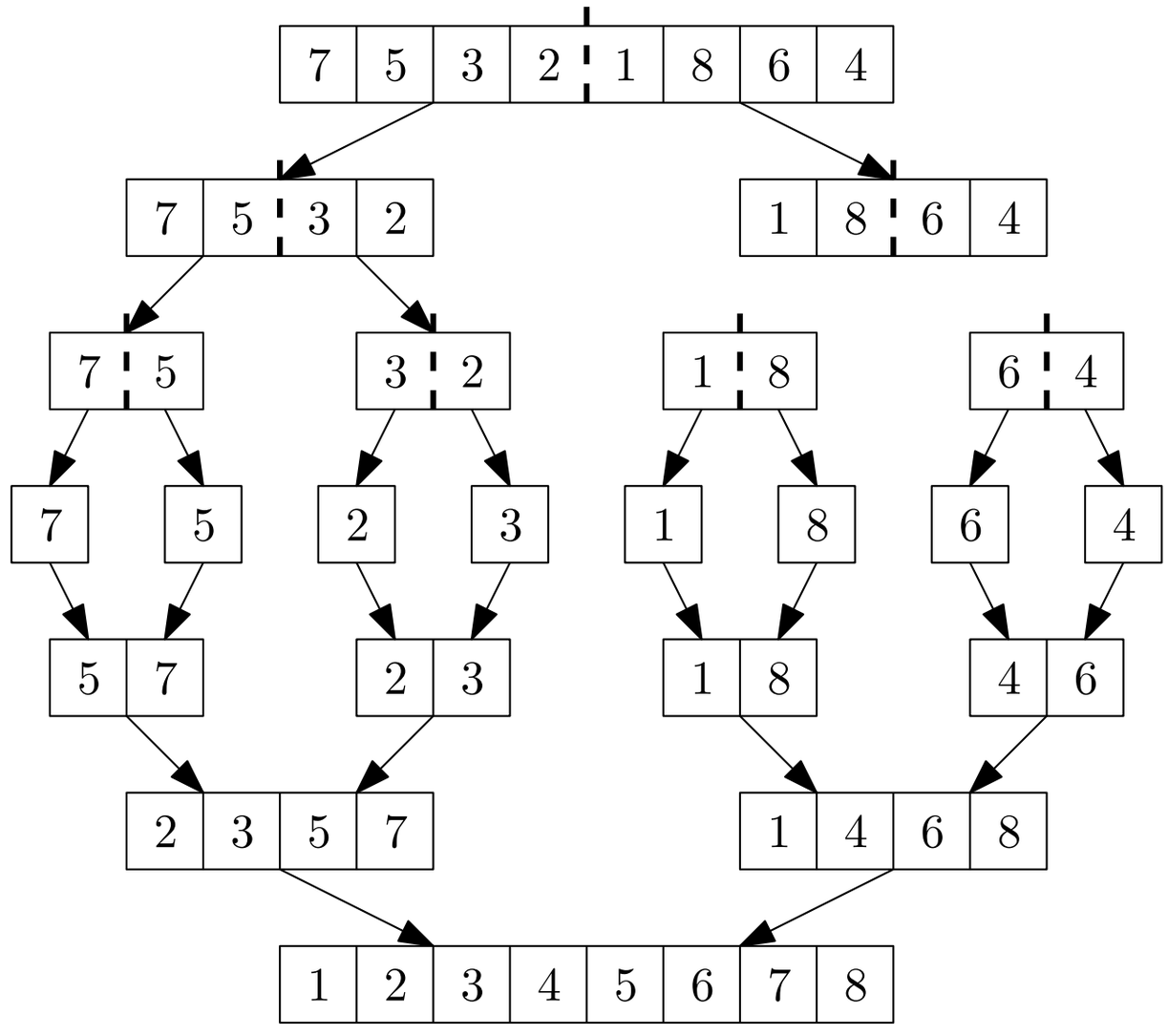
Closest Pear

1. Introduction
2. Prelude: Solving recursions
3. Lower bounds
4. Divide-and-conquer
5. Randomized incremental construction

- Split (Quicksort) or combine (Mergesort)



Quicksort



Mergesort

Divide-and-Conquer [Preparata and Hong, 1977]

4. Bobrow, D.G., Burchfiel, J.D., Murphy, D.L., and Tomlinson, R.S. TENEX, a paged time sharing system for the PDP-10. *Comm. ACM 15*, 3 (March 1972), 135–143.
5. Bobrow, D.G., and Murphy, D.L. The structure of a LISP system using two level storage. *Comm. ACM 10*, 3 (March 1967). 155–159.
6. Cheney, C.J. A nonrecursive list compacting algorithm. *Comm. ACM 13*, 11 (Nov. 1970), 677–678.
7. Deutsch, L.P. An interactive program verifier. Ph.D. Th., Comptr. Sci. Dep., U. of California, Berkeley, Calif., May 1973.
8. Deutsch, L.P. A LISP machine with very compact programs. Third Int. Joint Conf. on Artificial Intelligence, Stanford, Calif., 1973, pp. 697–703.
9. Fenichel, R.R., and Yochelson, J.C. A LISP garbage-collector for virtual-memory computer systems. *Comm. ACM 12*, 11 (Nov. 1969), 611–612.
10. Hansen, W.J. Compact list representation: Definition, garbage collection, and system implementation. *Comm. ACM 12*, 9 (Sept. 1969), 499–507.
11. Hehner, E.C.R. Matching program and data representations to a computing environment. Ph.D. Th., Comptr. Systems Res. Group, U. of Toronto, Toronto, Canada, Nov. 1974.
12. McCarthy, J., et al. *LISP 1.5 Programmer's Manual*. M.I.T. Press, Cambridge, Mass., 1962.
13. Minsky, M.L. A LISP garbage collector algorithm using serial secondary storage. Artificial Intelligence Proj. Memo 58 (rev.), Project MAC, M.I.T., Cambridge, Mass., Dec. 1963.
14. Quam, L.H. Stanford LISP 1.6 manual. Artificial Intelligence Proj., Stanford University, Stanford, Calif., Sept. 1969.
15. Reboh, R., and Sacerdoti, E. A preliminary QLISP manual. Tech. Note 81, Stanford Res. Inst. AI Center, Menlo Park, Calif., Aug. 1973.
16. Sacerdoti, E. The nonlinear nature of plans. Fourth Int. Joint Conf. on Artificial Intelligence, Tbilisi, Georgia, U.S.S.R., 1975, pp. 206–214.
17. Shannon, C.E. A mathematic theory of communication. *Bell System Tech. J. 27* (July 1948), 379–423.
18. Smith, D.H., Masinter, L.M., and Sridharan, N.S. Heuristic DENDRAL: Analysis of molecular structure. In *Computer Representation and Manipulation of Chemical Information*, W.T. Wipke, S. Heller, R. Feldman, and E. Hyde, Eds., Wiley, New York, 1974.
19. Teitelman, W. INTERLISP Reference manual. Xerox Palo Alto Res. Center, Palo Alto, Calif., 1974.
20. Van der Poel, W.L. A Manual of HISP for the PDP-9. Technical U., Delft, Netherlands.
21. Weissman, C. *LISP 1.5 Primer*. Dickenson Pub. Co., Belmont, Calif., 1967.
22. Wilner, W.T. Design of the B1700. Proc. AFIPS 1972 FJCC, Vol. 41, AFIPS Press, Montvale, N.J., pp. 579–586.
23. Zipf, G.K. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Reading, Mass., 1949.

Programming Techniques — G. Manacher S.L. Graham Editors

Convex Hulls of Finite Sets of Points in Two and Three Dimensions

F. P. Preparata and S. J. Hong
University of Illinois at Urbana-Champaign

The convex hulls of sets of n points in two and three *dimensions can be determined with O(n log n)* operations. The presented algorithms use the "divide and conquer" technique and recursively apply a merge procedure for two nonintersecting convex hulls. Since any convex hull algorithm requires at least O(n log n) operations, the time complexity of the proposed algorithms is optimal within a multiplicative constant.

Key Words and Phrases: computational complexity, convex hull, optimal algorithms, planar set of points, spatial set of points
CR Categories: 4.49, 5.25, 5.32

1. Introduction

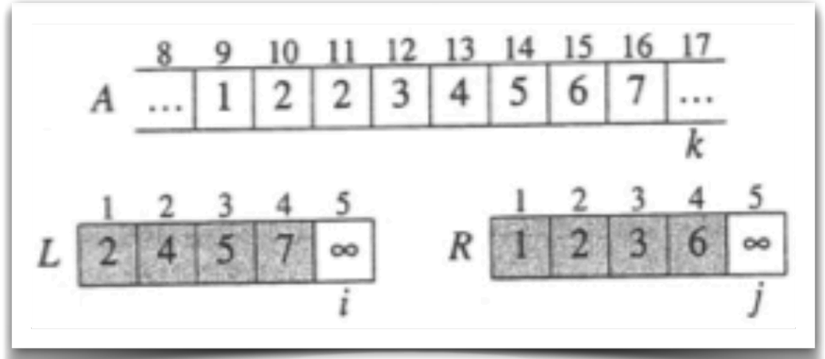
The determination of the convex hull of a finite set of points is relevant to several problems in computer graphics, design automation, pattern recognition and operations research: references [3, 4, 10]—just to cite a few—discuss some interesting applications in these areas, which require convex hull computation.

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DAAB-07-72-C-0259.

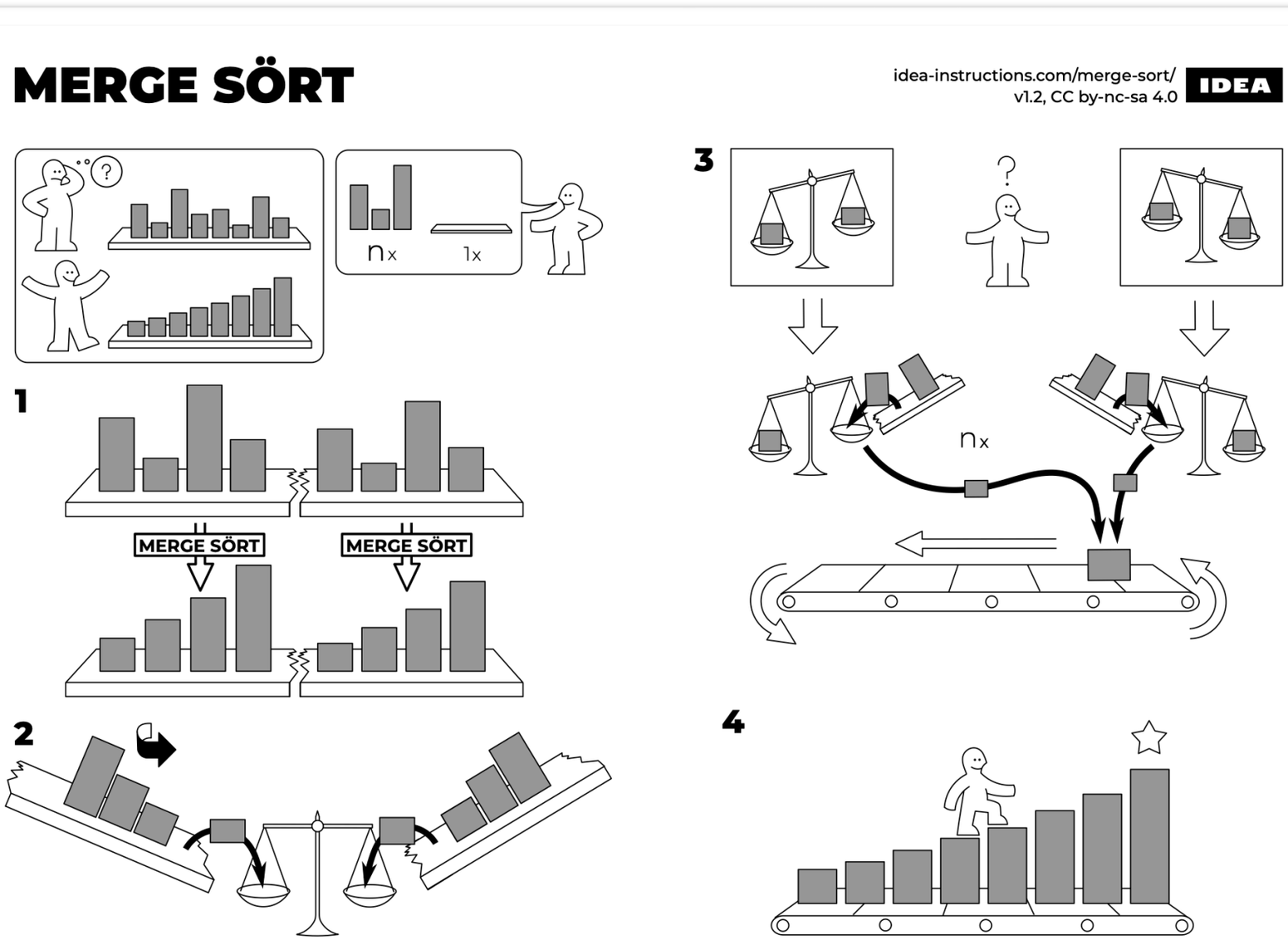
Authors' addresses: F.P. Preparata, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801; S.J. Hong, IBM Systems Product Division, Poughkeepsie, NY 12602. This work was completed while the second author was on leave at the University of Illinois.

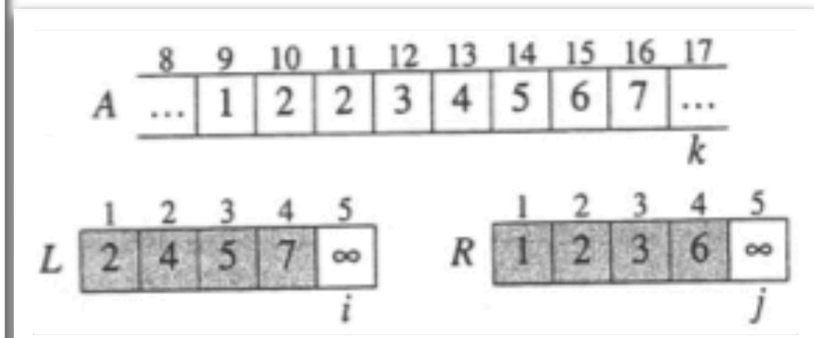
Basic idea:

- Balanced subdivision, solve recursively
- Analogy: Mergesort
  - Split array in a balanced fashion into two arrays
  - Sort recursively
  - Combine



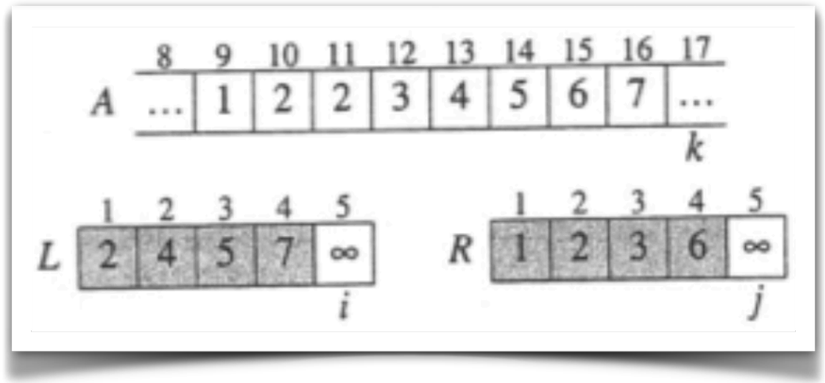
Divide-and-Conquer [Preparata and Hong, 1977]





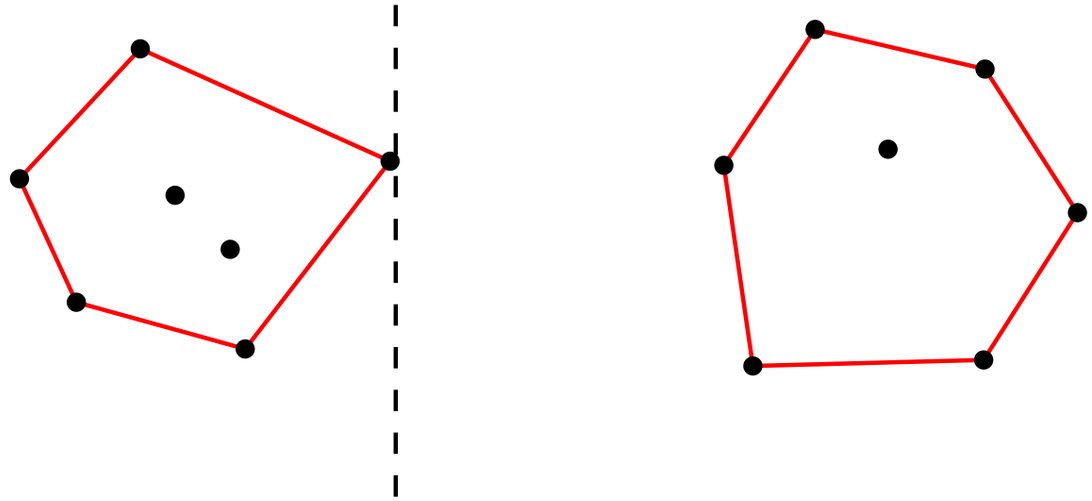
Basic idea:

- Balanced subdivision, solve recursively
- Analogy: Mergesort
  - Split array in a balanced fashion into two arrays
  - Sort recursively
  - Combine



Transfer to $\mathbb{R}^2$:

- Separate by x-median
- Recursively: right and left hull
- Combine left and right hull



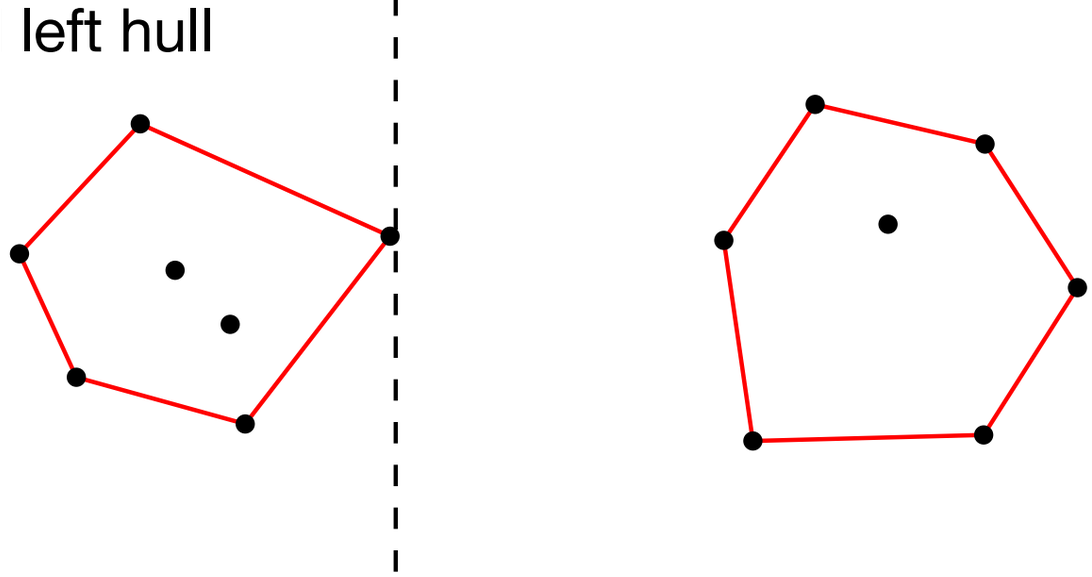
Assumption: General position (1. distinct x-coordinate and
2. no three points collinear)

- Split wrt. x-median $\Rightarrow$ Left and right hull disjoint
- Runtime:

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n) + f_{merge}(n) \textit{ with } f(n) \in \Theta(n)$$

$f_{merge}(n)$:= time for merging right and left hull

- Goal: $f_{merge}(n) \in \Theta(n)$
- Then $T(n) \in \mathcal{O}(n \log n)$



Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

$$T(n) = \sum_{i=1}^{m} T(\alpha_i \cdot n) + \Theta(n^k),$$

where $\alpha_i \in \mathbb{R}$ *with* $0 < \alpha_i < 1$, $m \in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \text{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \text{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \text{with } \sum_{i=1}^{m} \alpha_i^c = 1 \text{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$

- Runtime:

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n) + f_{merge}(n) \text{ with } f(n) \in \Theta(n)$$

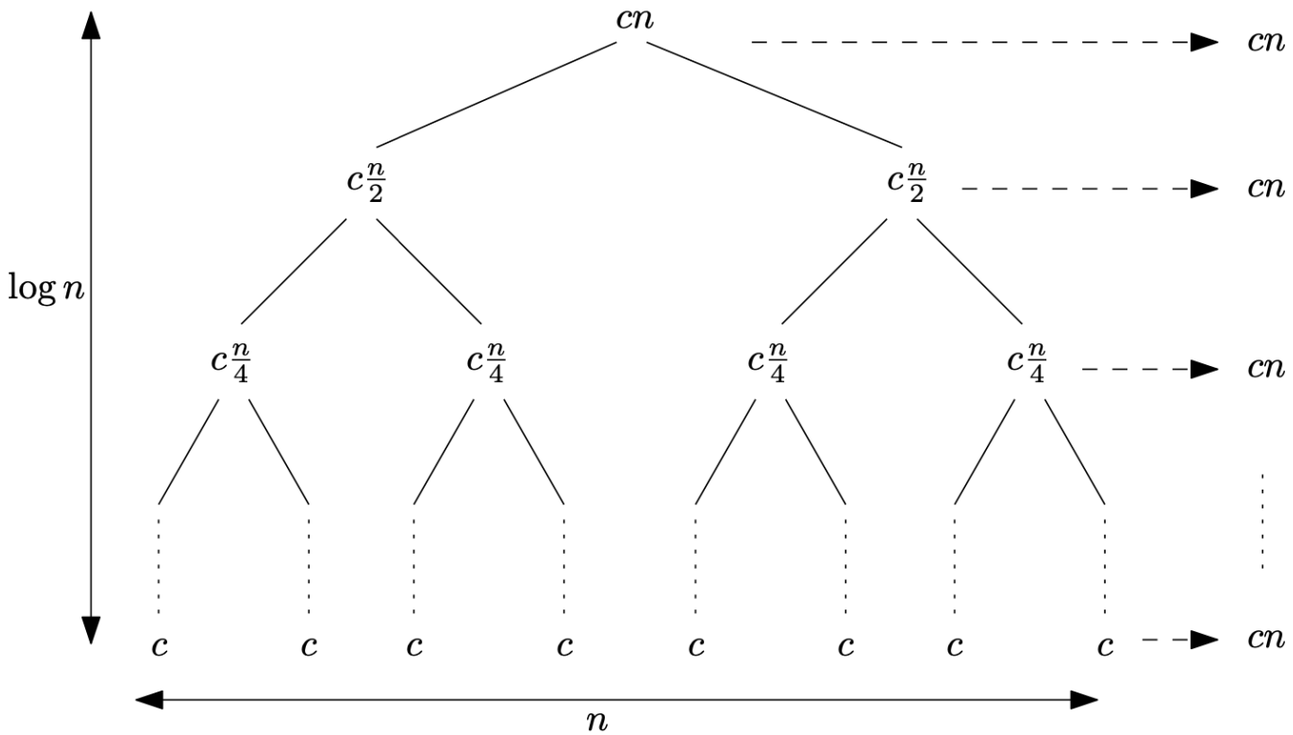
$f_{merge}(n)$:= time for merging right and left hull

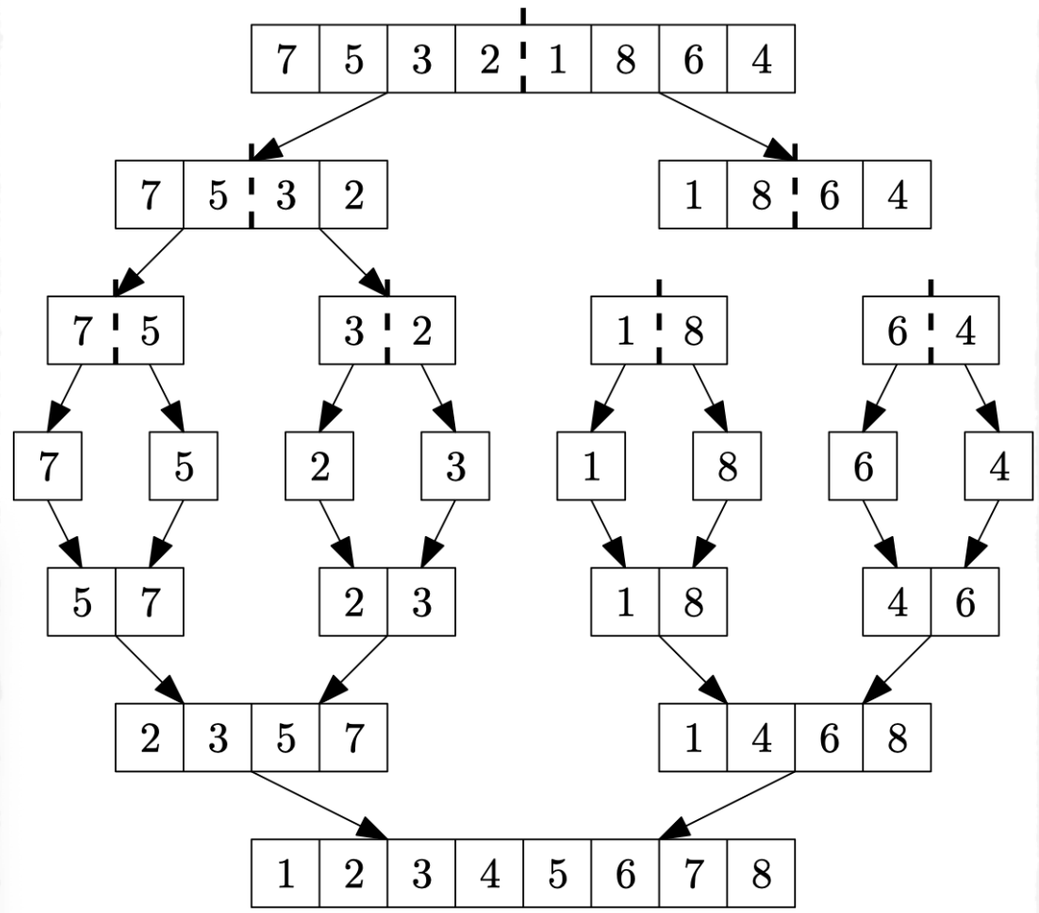
Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

$$T(n) = \sum_{i=1}^{m} T(\alpha_i \cdot n) + \Theta(n^k),$$

where $\alpha_i \in \mathbb{R}$ *with* $0 < \alpha_i < 1$, $m \in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \text{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \text{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \text{with } \sum_{i=1}^{m} \alpha_i^c = 1 \text{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$





Mergesort

$$T(n) = 2 \cdot T(\frac{n}{2}) + c \cdot n, \quad n \geq 2, c \in \mathbb{R}$$

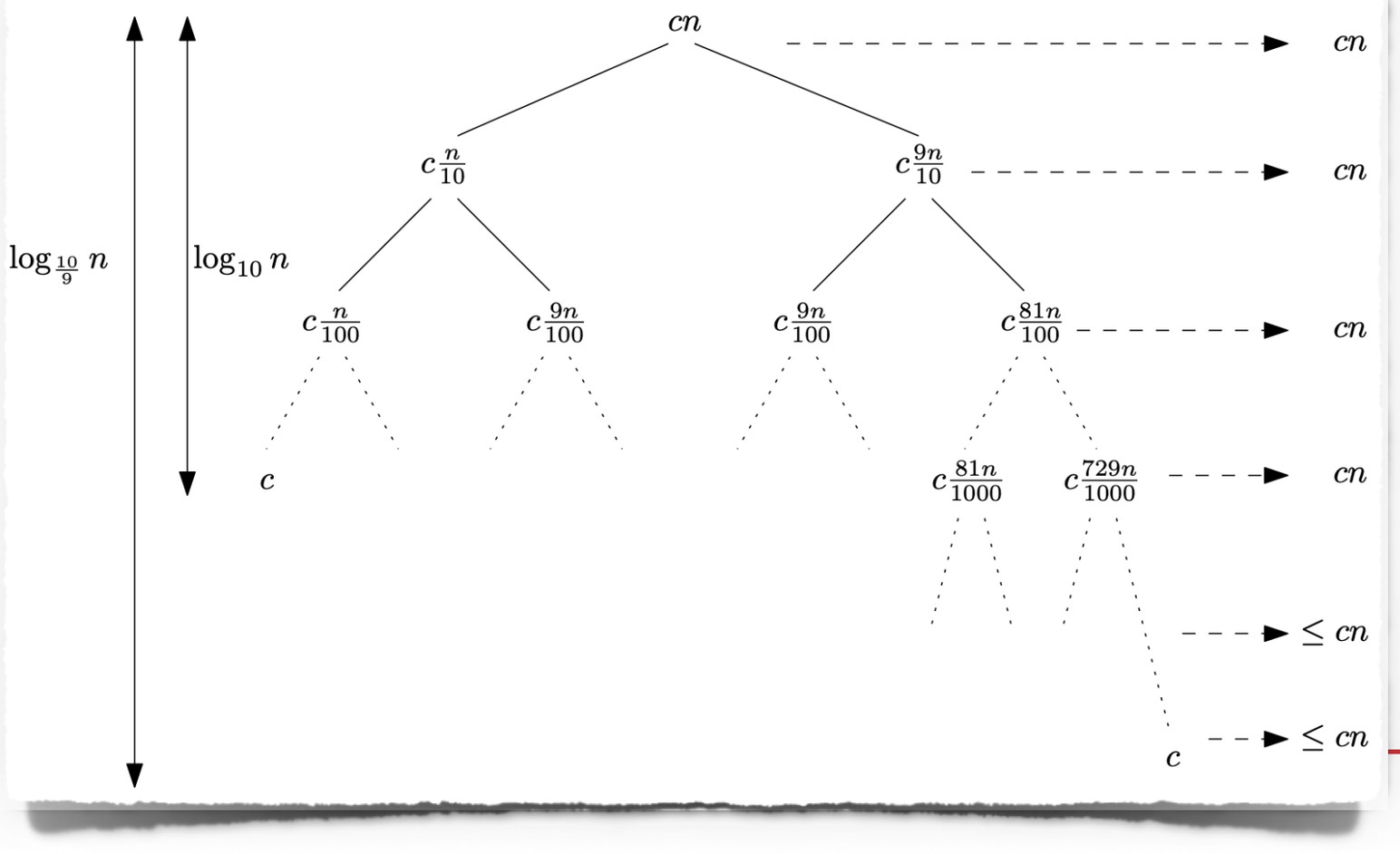
Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

$$T(n) = \sum_{i=1}^{m} T(\alpha_i \cdot n) + \Theta(n^k),$$

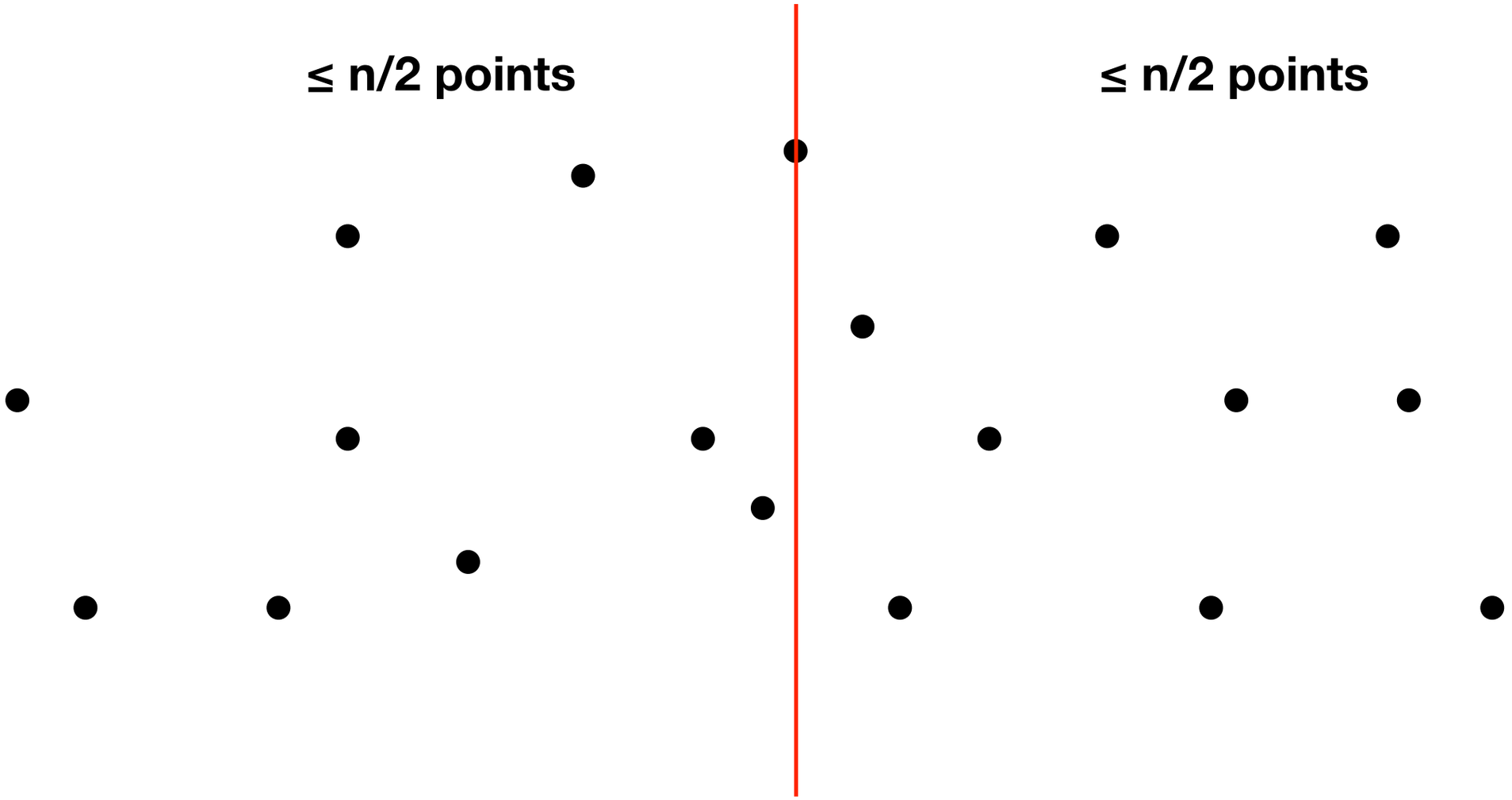
where $\alpha_i \in \mathbb{R}$ *with* $0 < \alpha_i < 1$, $m \in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \text{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \text{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \text{with } \sum_{i=1}^{m} \alpha_i^c = 1 \text{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$

$$T(n) \leq T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + cn$$



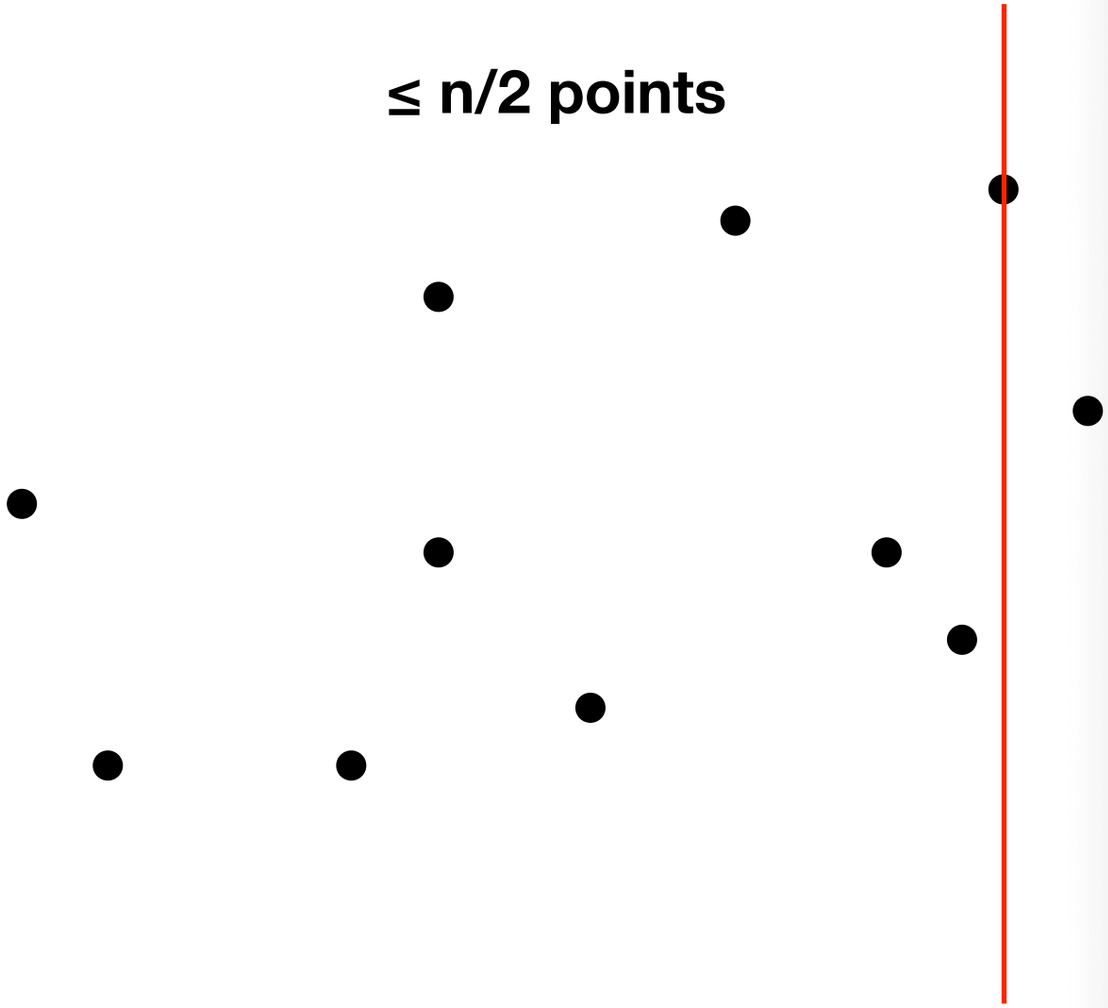
Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]



Theorem 3.4

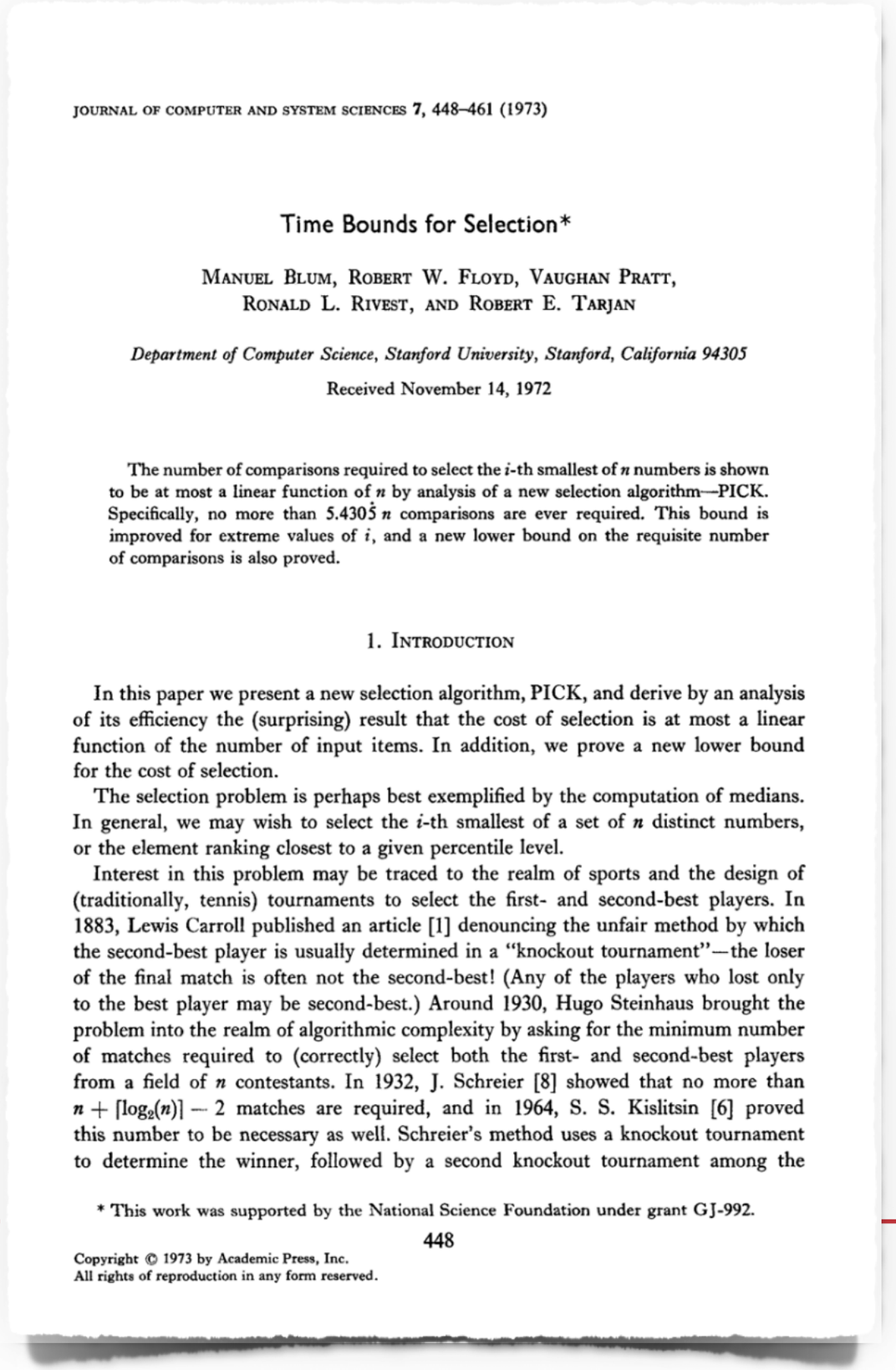
A median for n numbers can be computed in $\mathcal{O}(n)$.

Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]



Theorem 3.4

A median for n numbers can be computed in $\mathcal{O}(n)$.



JOURNAL OF COMPUTER AND SYSTEM SCIENCES 7, 448–461 (1973)

Time Bounds for Selection*

MANUEL BLUM, ROBERT W. FLOYD, VAUGHAN PRATT,
RONALD L. RIVEST, AND ROBERT E. TARJAN

Department of Computer Science, Stanford University, Stanford, California 94305

Received November 14, 1972

The number of comparisons required to select the i-th smallest of n numbers is shown to be at most a linear function of n by analysis of a new selection algorithm—PICK. Specifically, no more than $5.4305\,n$ comparisons are ever required. This bound is improved for extreme values of i, and a new lower bound on the requisite number of comparisons is also proved.

1. INTRODUCTION

In this paper we present a new selection algorithm, PICK, and derive by an analysis of its efficiency the (surprising) result that the cost of selection is at most a linear function of the number of input items. In addition, we prove a new lower bound for the cost of selection.

The selection problem is perhaps best exemplified by the computation of medians. In general, we may wish to select the i-th smallest of a set of n distinct numbers, or the element ranking closest to a given percentile level.

Interest in this problem may be traced to the realm of sports and the design of (traditionally, tennis) tournaments to select the first- and second-best players. In 1883, Lewis Carroll published an article [1] denouncing the unfair method by which the second-best player is usually determined in a "knockout tournament"—the loser of the final match is often not the second-best! (Any of the players who lost only to the best player may be second-best.) Around 1930, Hugo Steinhaus brought the problem into the realm of algorithmic complexity by asking for the minimum number of matches required to (correctly) select both the first- and second-best players from a field of n contestants. In 1932, J. Schreier [8] showed that no more than $n + \lceil \log_2(n) \rceil - 2$ matches are required, and in 1964, S. S. Kislitsin [6] proved this number to be necessary as well. Schreier's method uses a knockout tournament to determine the winner, followed by a second knockout tournament among the

* This work was supported by the National Science Foundation under grant GJ-992.

Theorem 3.4

A median for n numbers can be computed in $\mathcal{O}(n)$.

Proof idea:

$$X = \{1, 22, 10, 13, 24, 6, 18, 21, 4, 25, 11, 16, 2, 20, 8, 17, 5, 12, 19, 14, 3, 9, 15, 7, 23\}$$

- Group the numbers into sets of 5.

1	6	11	17	3
22	18	16	5	9
10	21	2	12	15
13	4	20	19	7
24	25	8	14	23

$\mathcal{O}(n)$

- Sort all quintuples.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$\mathcal{O}(n)$

Proof idea (cont):

- Compute the median of medians.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$$T(\frac{n}{5})$$

- Use the median of medians as pivot to reduce the set of numbers.

≥ n/4 numbers

≥ n/4 numbers

3	2	1	5	4
7	8	10	12	6
9	11	13	14	18
15	16	22	17	21
23	20	24	19	25

$$T(\frac{3n}{4})$$

$$T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + \Theta(n)$$

Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

$$T(n) = \sum_{i=1}^{m} T(\alpha_i \cdot n) + \Theta(n^k),$$

where $\alpha_i \in \mathbb{R}$ *with* $0 < \alpha_i < 1$, $m \in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \text{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \text{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \text{with } \sum_{i=1}^{m} \alpha_i^c = 1 \text{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$

$$T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + \Theta(n)$$

$$\sum_{i=1}^{m} \alpha_i^k = \frac{1}{5} + \frac{3}{4} = \frac{19}{20} < 1.$$

Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ ...

$$T(n) \in \Theta(n^k) = \Theta(n)$$

where $\alpha_i \in$ ... $\in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \text{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \text{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \text{with } \sum_{i=1}^{m} \alpha_i^c = 1 \text{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$

$$T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + \Theta(n)$$

$$\sum_{i=1}^{m} \alpha_i^k = \frac{1}{5} + \frac{3}{4} = \frac{19}{20} < 1.$$

1. Introduction
2. Prelude: Solving recursions
3. Lower bounds
4. Divide-and-conquer
5. Randomized incremental construction

Theorem 3.5

SORTING has a lower bound of $\Omega(n \log n)$.

Proof:

- Initially, we have $\mathcal{O}(n!)$ possible permutations.
- Each comparison partitions the remaining permutations into two sets.
- In the worst case, the larger set remains.
- Thus, we can at best guarantee that half the number of permutations remain.
- Until we get a unique permutation, we need at least $\Omega(\log(n!))$ comparisons.
- $$\log_2(n!) = \sum_{i=1}^{n} \log_2(i) \geq \sum_{i=\frac{n}{2}}^{n} \log_2(i) \geq \frac{n}{2} \log_2\left(\frac{n}{2}\right) = \frac{n}{2}(\log_2 n - 1) \in \Omega(n \log(n))$$

Theorem 3.6

Computing a closest pair for n numbers t

Theorem 1. *Any algebraic computation tree that solves the n-element distinctness problem must have complexity of at least* $\Omega(n \log n)$.

Lower Bounds For Algebraic Computation Trees

(Preliminary Report)

Michael Ben-Or †

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract — A topological method is given for obtaining lower bounds for the height of algebraic computation trees, and algebraic decision trees. Using this method we are able to generalize, and present in a uniform and easy way, almost all the known nonlinear lower bounds for algebraic

Despite the extensive research in algebraic complexity theory in recent years, no general lower bound method has been provided for algorithms that involve arithmetical operations and comparisons. Much less is known if we further allow the operation of root extraction or the algebraic operation of finding the root of a polynomial. Consider the following decision problem:

Example 1. Element Distinctness. *Given* $x_1, \ldots, x_n \in \mathbf{R}$, *is there a pair* i, j *with* $i \neq j$ *and* $x_i = x_j$ *?*

One can solve the element distinctness problem with the help of any efficient sorting algorithm using $O(n \log n)$ comparisons, or by computing the product $\prod_{i \neq j}(x_i - x_j)$ and comparing the computed result to zero (using $O(n \log n)$ mult/div). Allowing linear operations for free, we know of no previous result that indicates why more than $O(1)$ operations are required to solve this problem in the model considered here.

In this paper we provide a new topological method for obtaining lower bounds for this general type of algorithms, formally described as *algebraic computation trees.* Before giving the detailed computational model it is worthwhile to mention a concrete application of the method presented here.

Theorem 1. *Any algebraic computation tree that solves the n-element distinctness problem must have complexity of at least* $\Omega(n \log n)$.

This result extends the lower bounds of Dobkin and Lipton [5] for the linear decision tree model, and the lower bounds of Baur and Strassen [1] for the straight line complexity of the above product.

Our new lower bound method rests heavily on a result from real algebraic geometry due to Milnor [11] and Thom [23] that bounds the "topological complexity" of real algebraic varieties. Except for this result the proofs of our main theorems are elementary and require only basic knowledge of algebra and topology. The new method also provides a unified and easy way to prove nonlinear lower bounds for straight line computations, algebraic decision trees, and other previously untouchable problems such as lower bounds for the complexity of constructions with a ruler and compass in plane Euclidean geometry.

In the next section we rigorously specify our basic computational model. The third section is devoted to a technical result needed for our main theorems that are presented in section four. In section four we also show how to extend our computational model to allow more algebraic operations such as taking k-th roots or computing the roots of a polynomial.

In section five we show how to apply our method to the bounded degree algebraic decision tree model, thus solving the open problems in [20]. Section six is devoted to applications and in particular to the proof of the result on the element distinctness problem (Theorem 1) mentioned above.

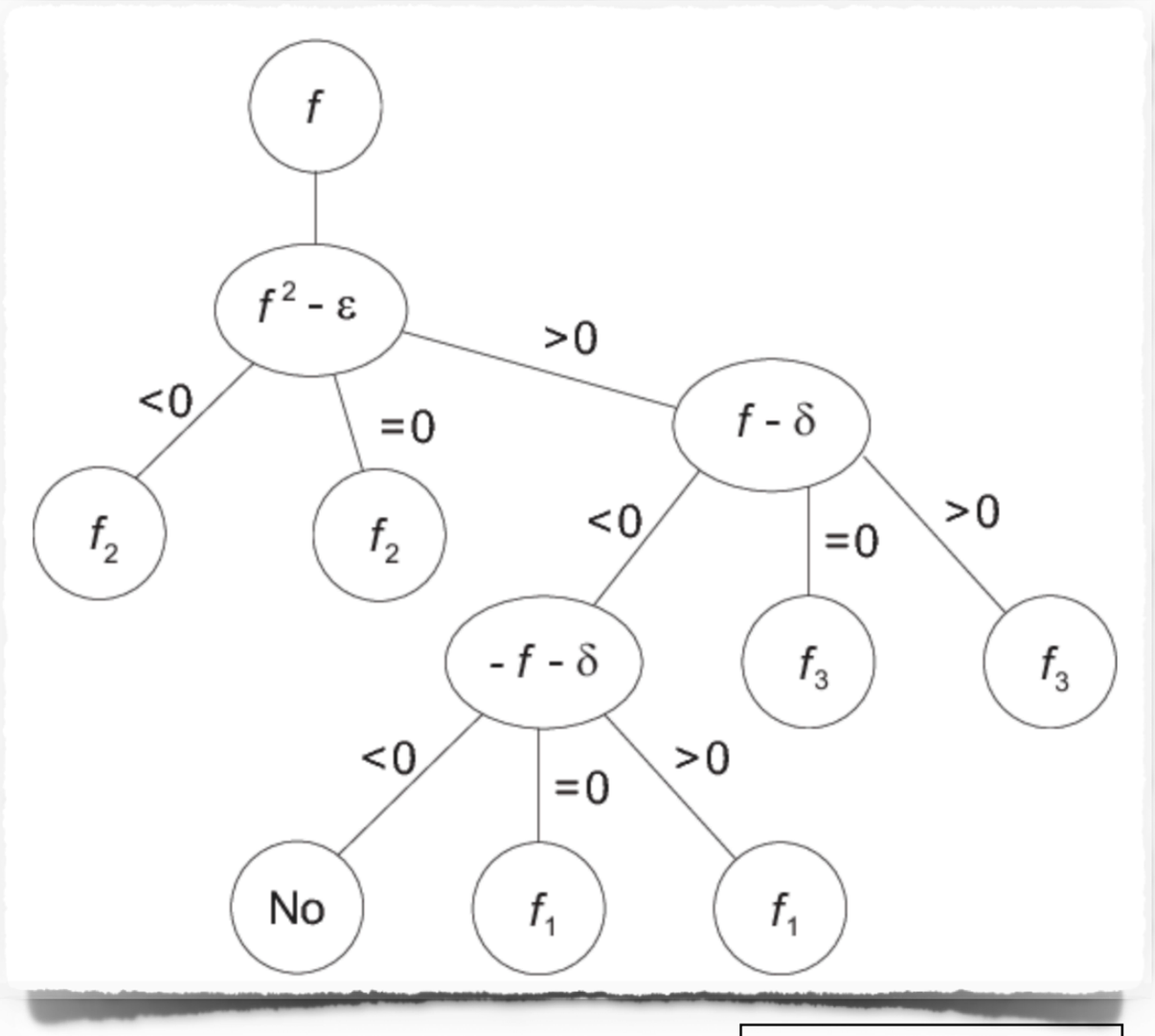
† Research supported by a Weizmann Postdoctoral fellowship and by NSF grant MCS-8006938.

Theorem 3.6

Computing a closest pair for n numbers takes at least $\Omega(n \log n)$.

Ideas:

- Consider *algebraic decision trees.*



A Gabrielov, N Vorobjov

Theorem 3.6

Computing a closest pair for n numbers

Theorem 7. *Let $W \subseteq \mathbf{Q}^n$ be any set, and let T be a computation tree that solves the membership problem for W. Then*

$$M(T) = O(\log N - n)$$

where N is the number of connected components of $\overline{W}$ in $\mathbf{R}^n$ with non null interior.

Ideas:

- Consider *algebraic decision trees.*
- Each set of numbers corresponds to a point in $\mathbb{R}^n$.
- Consider subsets of $\mathbb{R}^n$ that share the same membership properties.
- Consider the number N of connected components of $\mathbb{R}^n$.
- Show that the height of an algebraic decision tree is at least $\Omega(\log N - n)$.

Theorem 3.6

Computing a closest pair for n numbers takes at least $\Omega(n \log n)$.

Ideas:

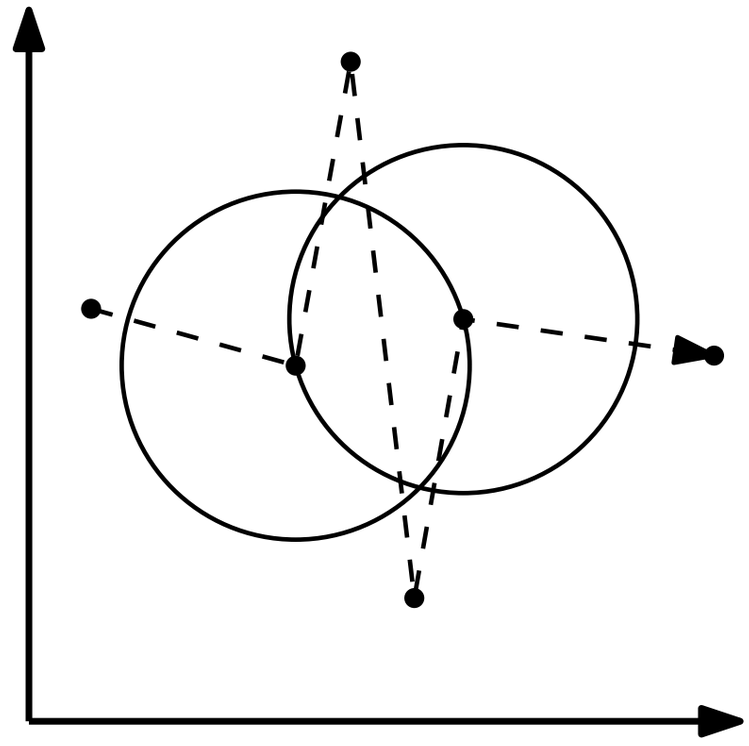
- Consider *algebraic decision trees.*
- Each set of numbers corresponds to a point in $\mathbb{R}^n$.
- Consider subsets of $\mathbb{R}^n$ that share the same membership properties.
- Consider the number N of connected components of $\mathbb{R}^n$.
- Show that the height of an algebraic decision tree is at least $\Omega(\log N - n)$.
- Show that ELEMENT UNIQUENESS has many connected components.
- Note that determining a small minimum distance solves ELEMENT UNIQUENESS.

1. Introduction
2. Prelude: Solving recursions
3. Lower bounds
4. Divide-and-conquer
5. Randomized incremental construction

Theorem 3.7
Computing a closest pair for $\mathcal{P} \subset \mathbb{R}$ can be computed in $\mathcal{O}(n \log n)$.

Proof:

- Sort.
- Run through points in order and consider distance.



Attempt: Apply to $\mathbb{R}^2$:

- Sort (wrt.$\leq_x$ / $\leq_y$?)
- Closest pair does not have to be adjacent.

Divide-and-Conquer [Bentley and Shamos 1976]

DIVIDE-AND-CONQUER IN MULTIDIMENSIONAL SPACE

Jon Louis Bentley
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27514

and

Michael Ian Shamos
Departments of Computer Science and Mathematics
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

We investigate a divide-and-conquer technique in multidimensional space which decomposes a geometric problem on N points in k dimensions into two problems on N/2 points in k dimensions plus a single problem on N points in k-1 dimension. Special structure of the subproblems is exploited to obtain an algorithm for finding the two closest of N points in O(N log N) time in any dimension. Related results are discussed, along with some conjectures and unsolved geometric problems.

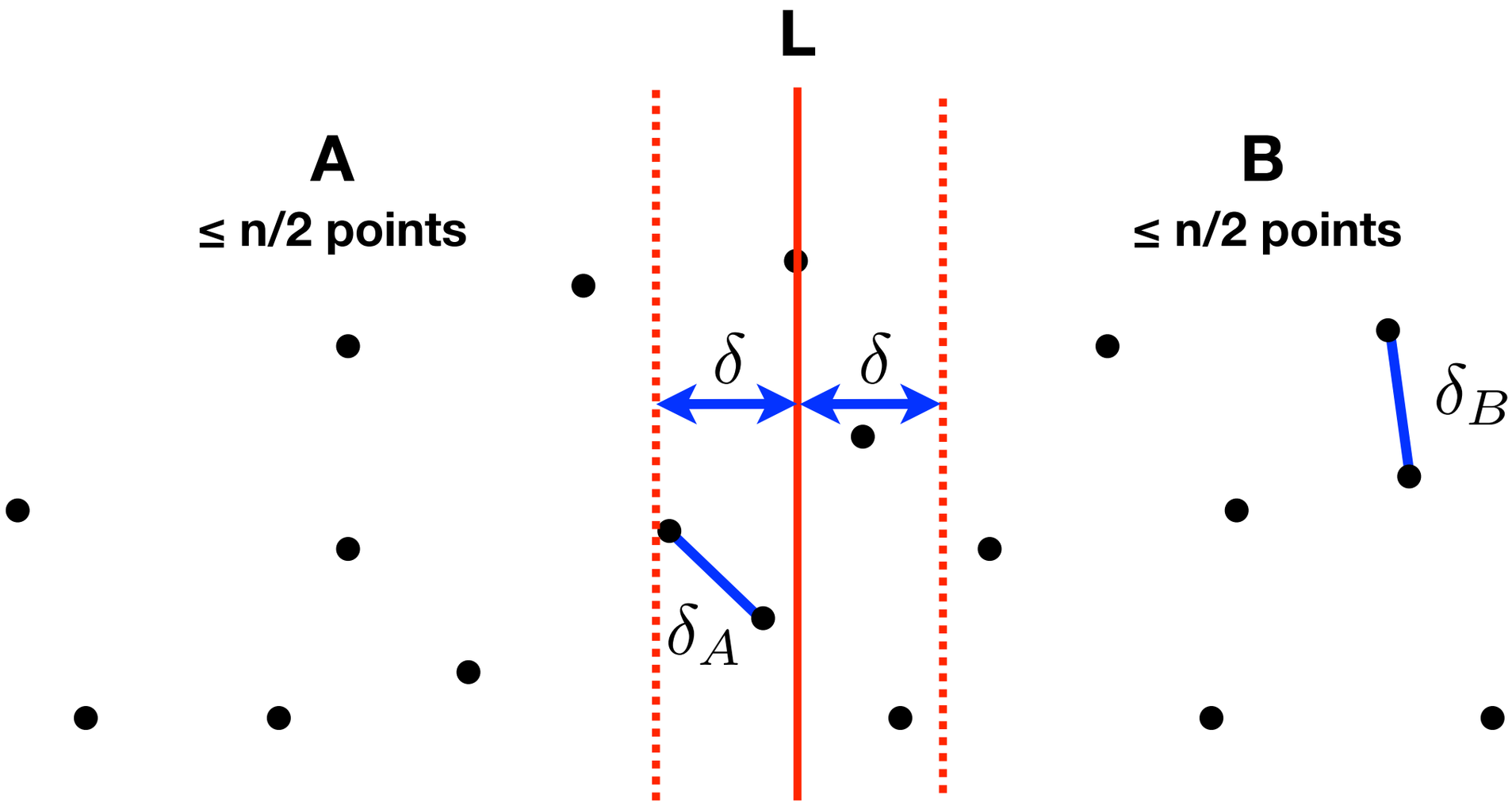
Introduction

A principal failing of computational geometry [Shamos, Shamos and Hoey] is that it has not successfully addressed problems in greater than two dimensions. Such a study would have important practical and theoretical benefits -- it would shed light on linear programming, multidimensional data analysis, geometric optimization, and retrieval on multiple keys, as well as provide a link between complexity and dimensionality. Some preliminary results are known. [Preparata and Hong] show that the convex hull of N points can be found in $O(N \log N)$ time in three dimensions but that $O(N^2)$ is a lower bound in any higher dimension. The maxima of a set of vectors can be found in $O(N \log^{k-2} N)$ time in dimension k [Kung et al.] and we conjecture that this time suffices to identify the extreme points of a k-dimensional set.

The most basic questions, however, have not been studied. These include finding nearest and farthest points, determining separability of point sets and other elementary properties. We intend in this paper to begin a systematic investigation of higher-dimensional geometry and its relation to complexity.

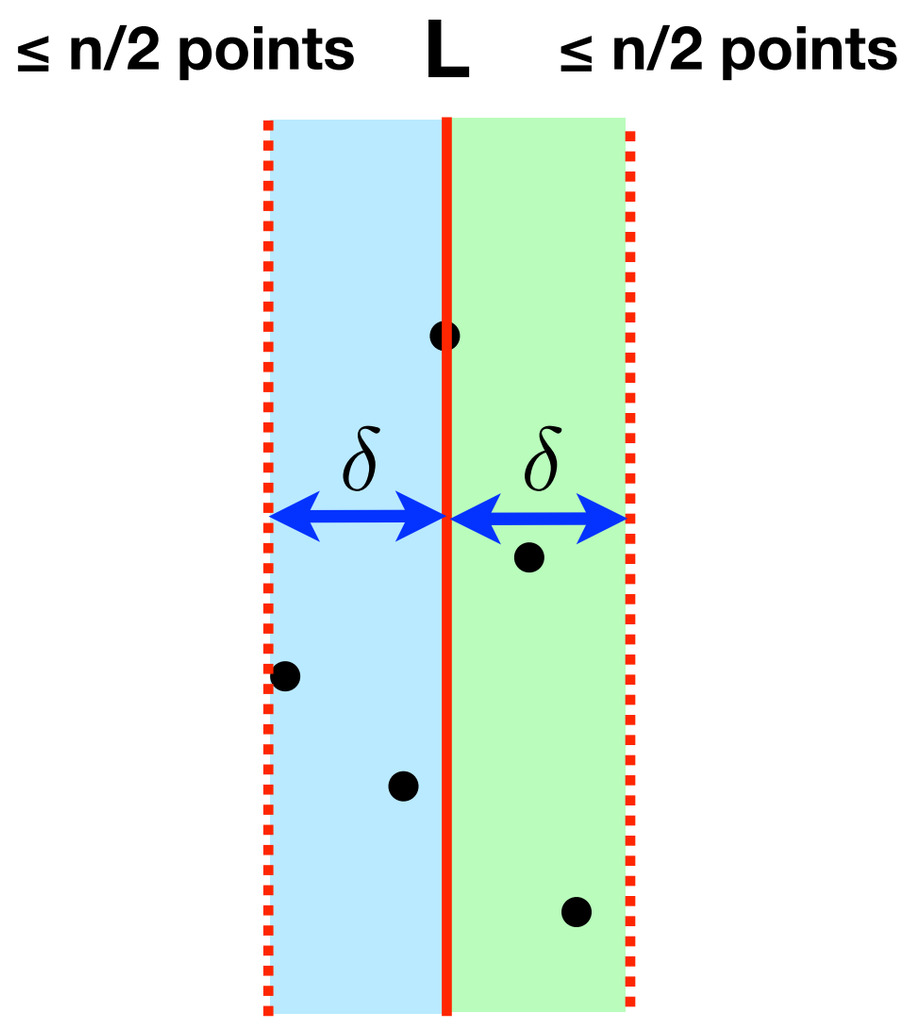
Closest-Point Problems

In this section we will investigate a number of problems dealing with the proximity of N points in Euclidean k-space. The most primitive closest-point problem is that of finding the two nearest of the N points; we will let P(N,k) denote the worst-case time of the best possible algorithm (the minimax complexity) for solving the closest-pair problem. The fixed radius-near-neighbor problem asks for all pairs of points within some fixed distance δ of one another. A special case of this



- Subdivide the set by a median line.
- Solve the two subproblems recursively in $2T(\frac{n}{2})$.
- Merge the two subproblems in $\mathcal{O}(n)$.

Divide-and-Conquer 2 [Bentley and Shamos 1976]



- Issue: Still $\Theta(n)$ on either side possible.
- Brute-force enumeration yields $\Theta(n^2)$.
- Additional ideas are necessary to get merge time $\mathcal{O}(n)$.

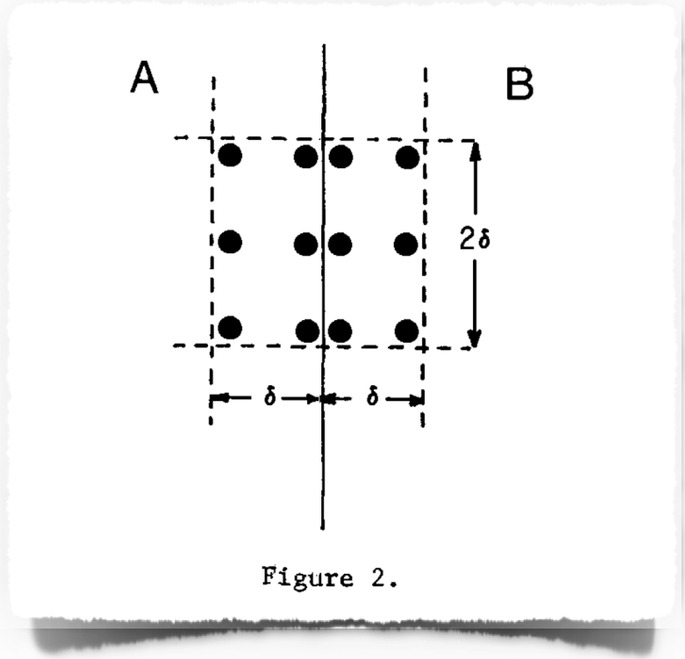
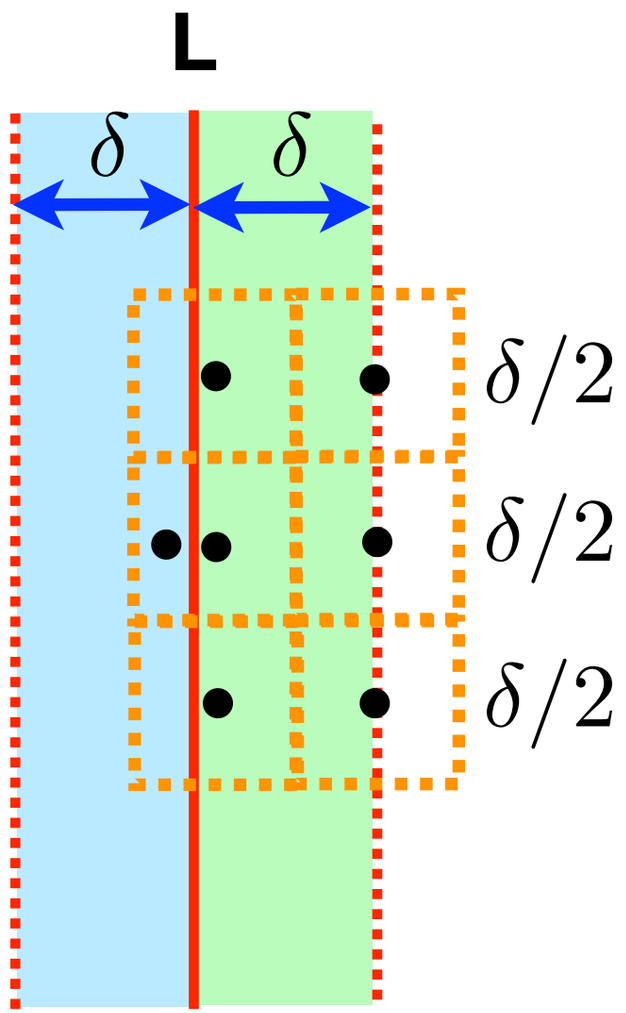
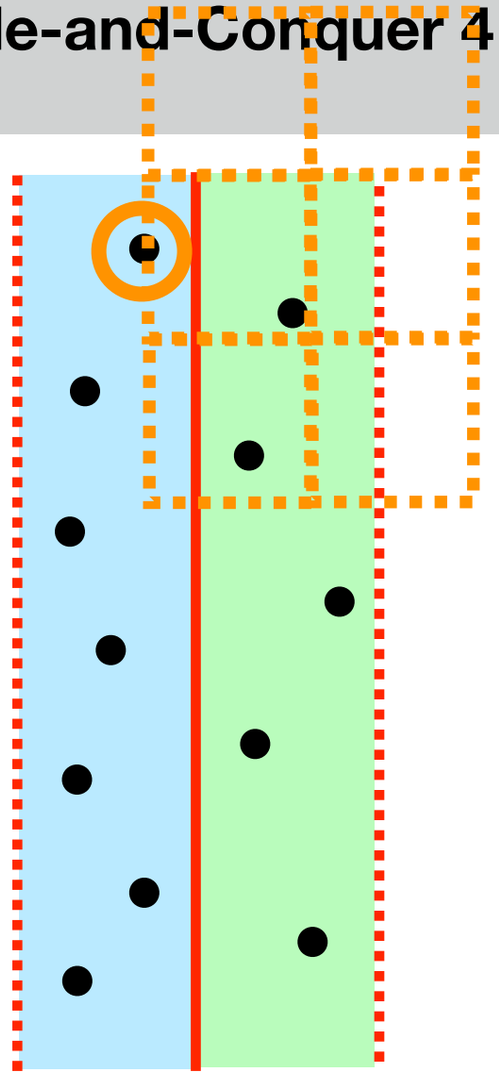


Figure 2.



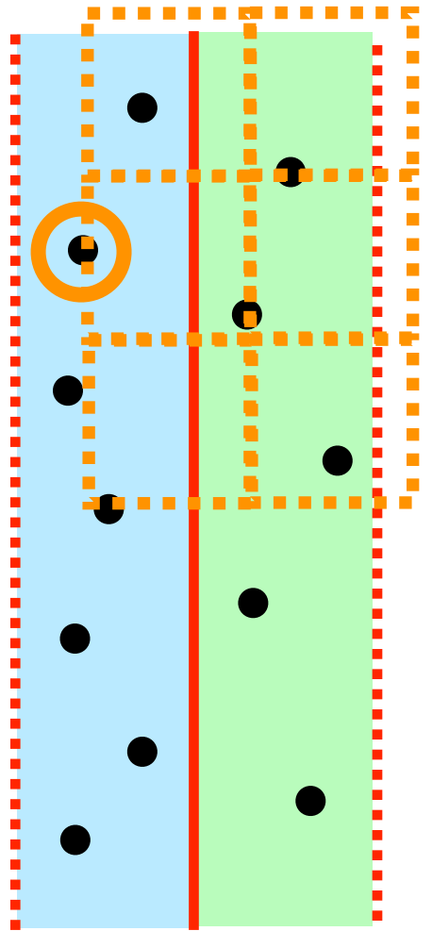
- Observe: Points on one side cannot be closer than δ.
- As consequence, no point can have many close neighbors on the other side.
- Based on a packing arguments for $\delta/2$-balls; works in any fixed dimension.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



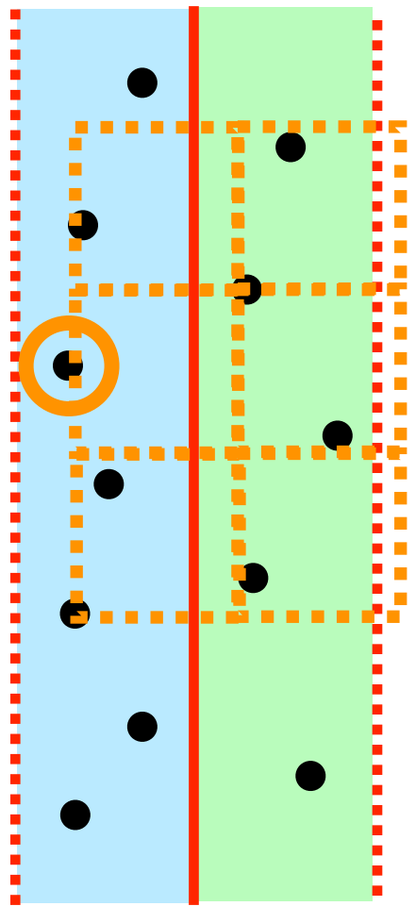
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



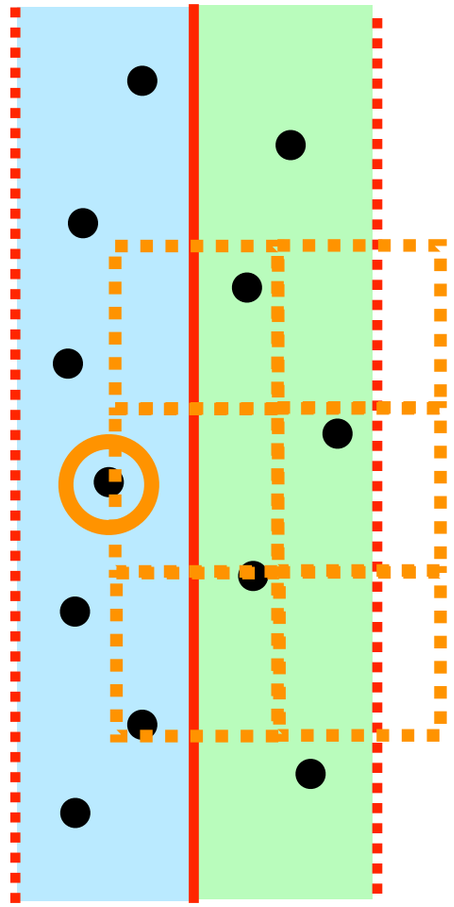
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



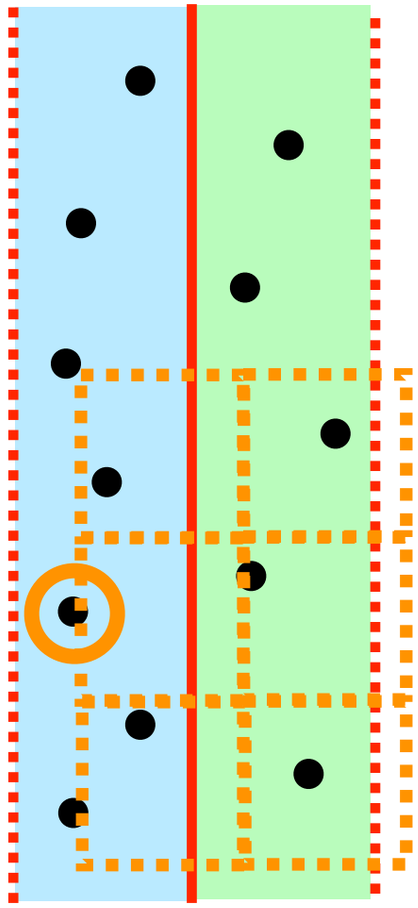
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



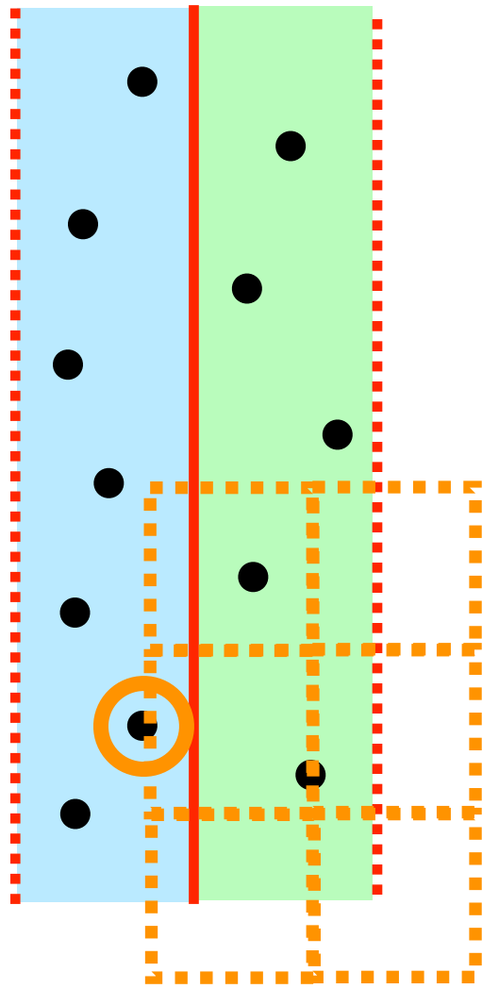
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



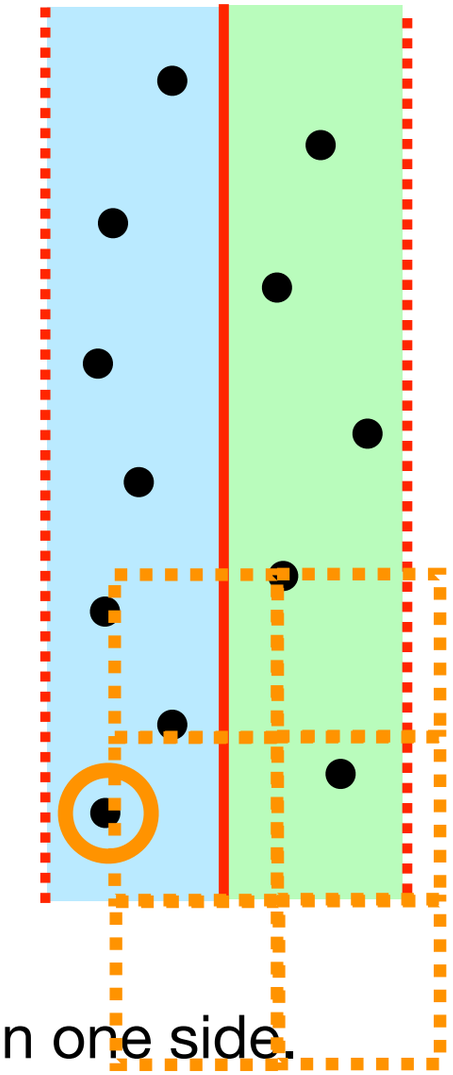
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



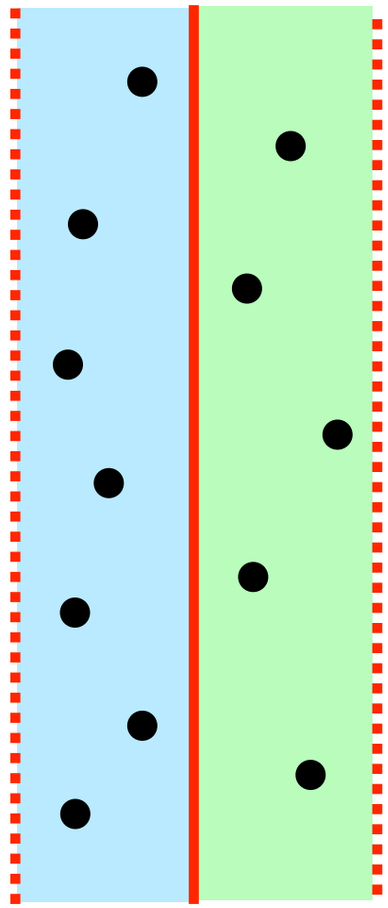
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



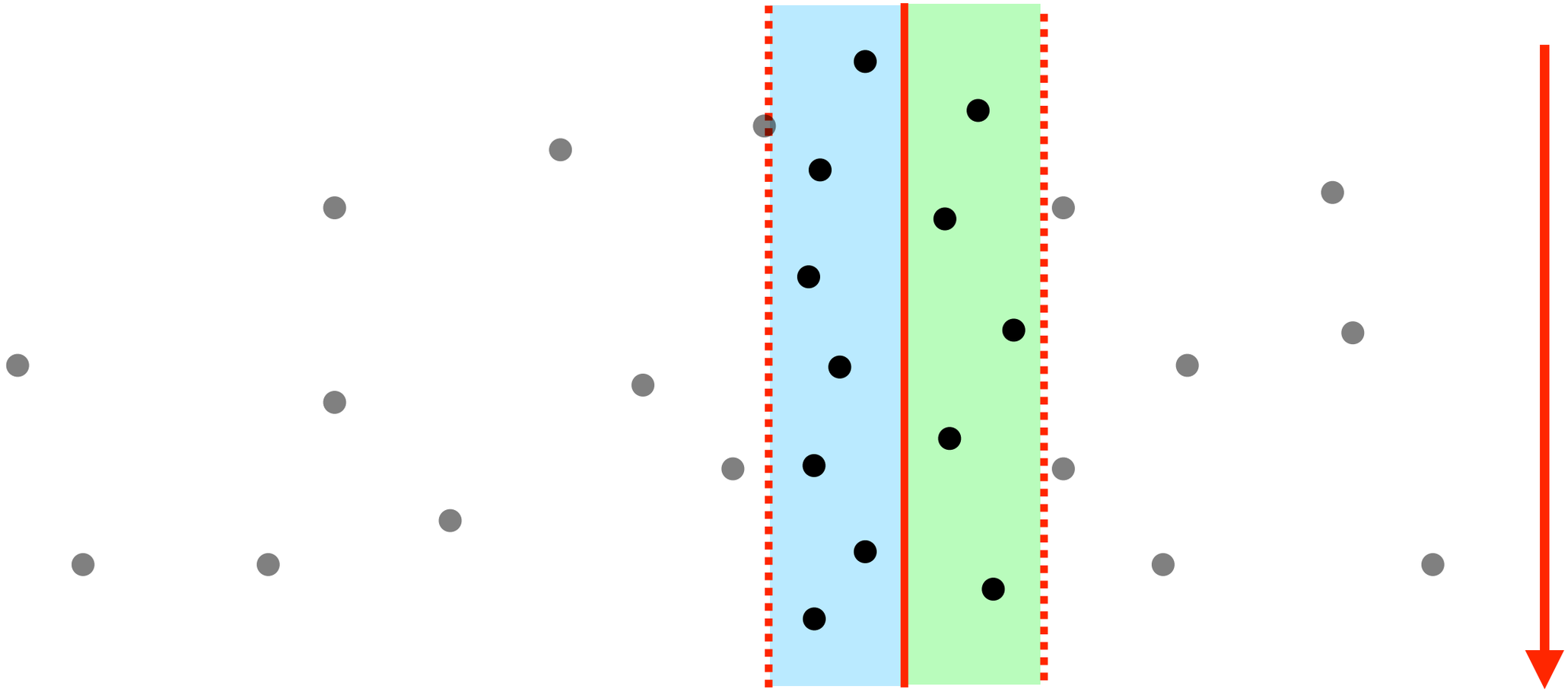
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



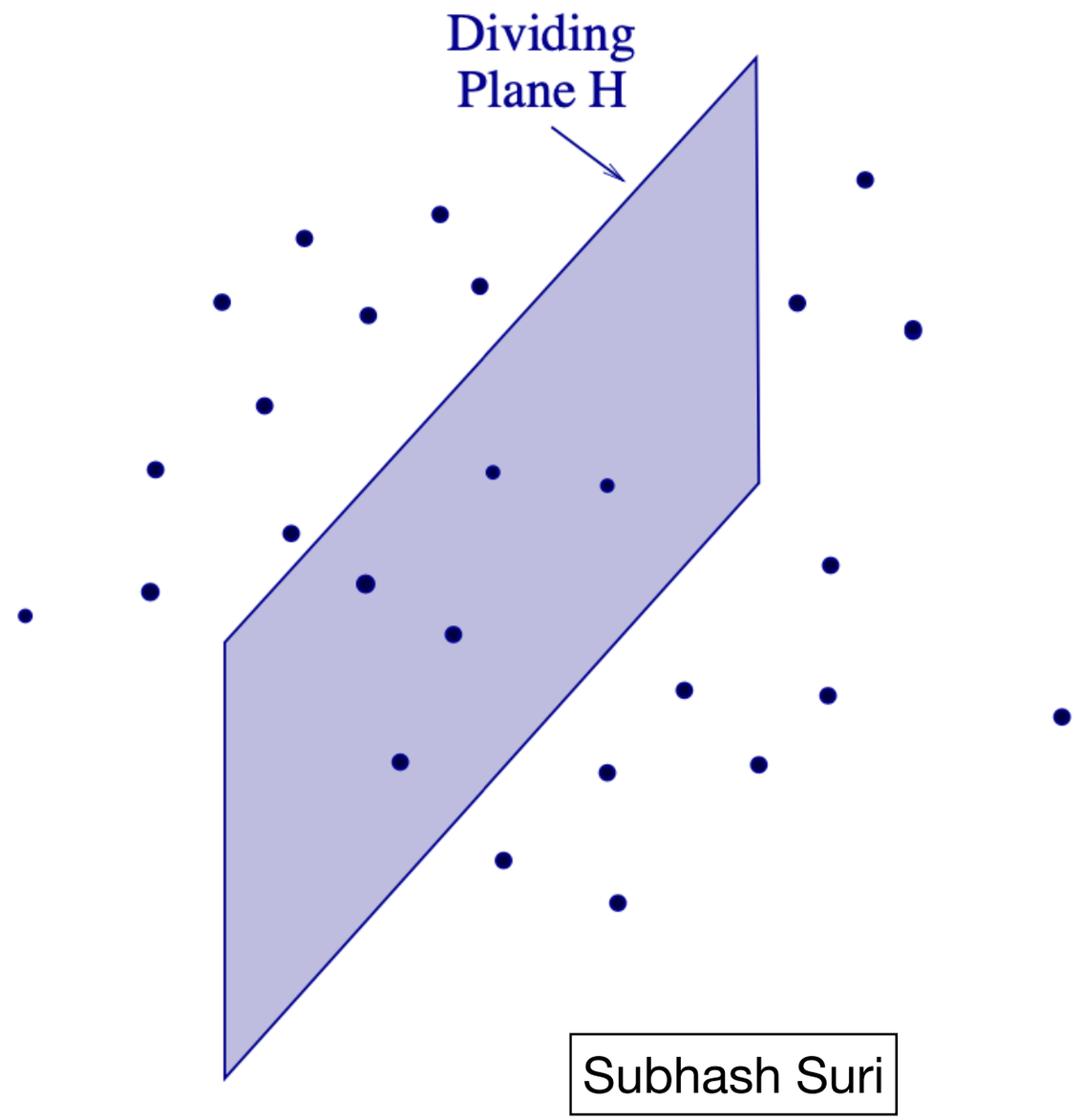
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.
- Sorting takes $\mathcal{O}(n \log n)$; total time $\mathcal{O}(n \log^2 n)$.

Divide-and-Conquer 4 [Bentley and Shamos 1976]



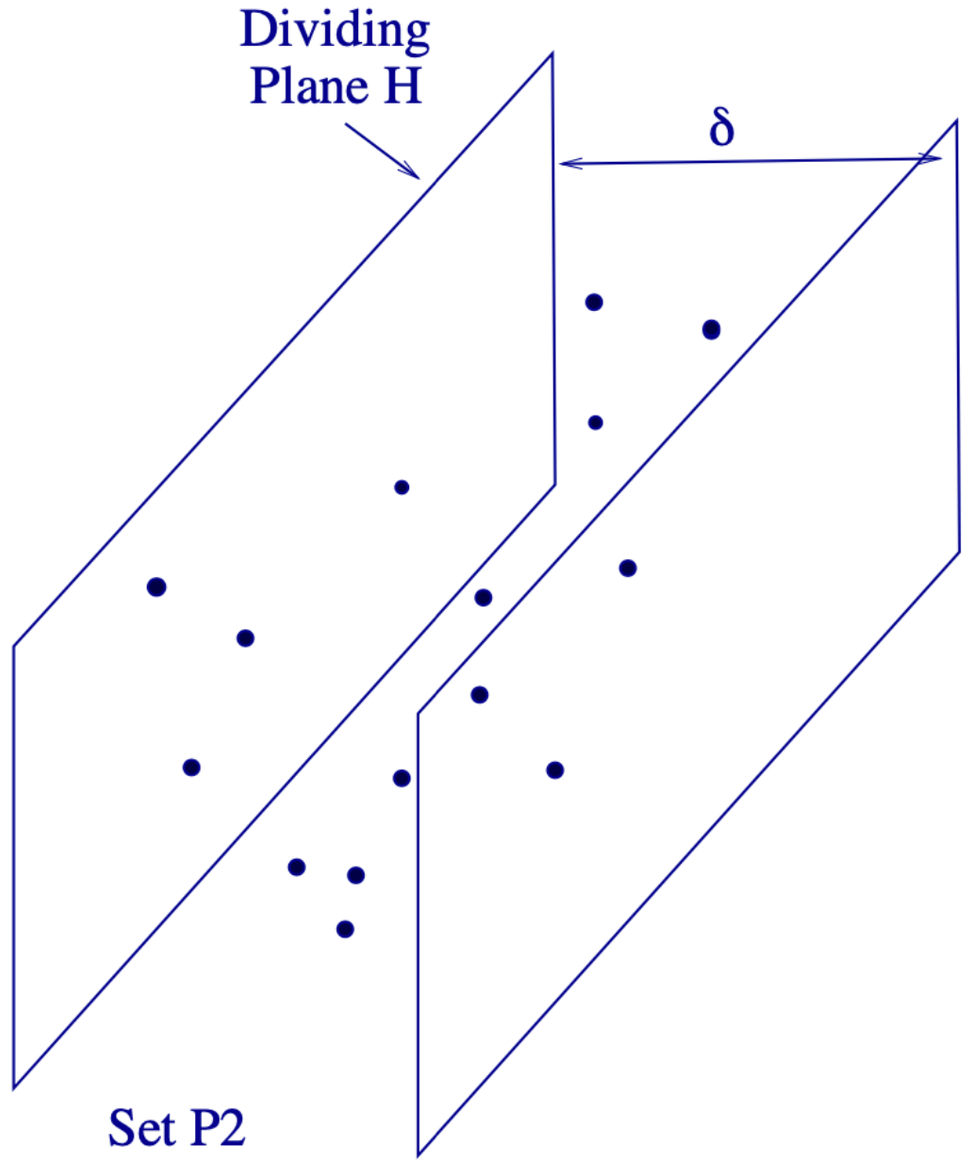
- Thus: Scan linear sequence on one side.
- Keep track of potential neighbors on other side.
- Sorting takes $\mathcal{O}(n \log n)$; total time $\mathcal{O}(n \log^2 n)$.
- After global presorting: $\mathcal{O}(n)$; total time $\mathcal{O}(n \log n)$.

- Same idea: Divide-and-conquer.



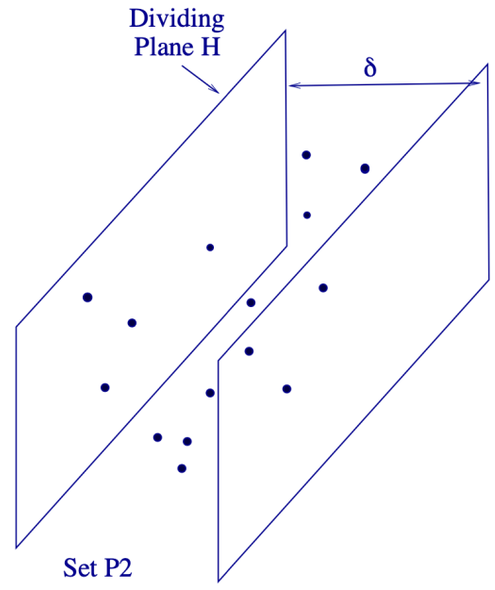
Subhash Suri

- Same idea: Divide-and-conquer.
- Merge step: one dimension lower:



Higher dimensions

- Same idea: Divide-and-conquer.
- Merge step: one dimension lower:



$$
\begin{aligned}
T(n,d) &= 2T(n/2,d) + U(n,d-1) + O(n) \\
&= 2T(n/2,d) + O(n(\log n)^{d-2}) + O(n) \\
&= O(n(\log n)^{d-1}).
\end{aligned}
$$

- **If we could show that the problem size in the conquer step is $m \leq n/(\log n)^{d-2}$, then $U(m, d-1) = O(m(\log m)^{d-2}) = O(n)$.**

Theorem 5. (Existence of a cut-plane in k-space.)

Given a sparse collection of N points in k-space, there exists a cut-plane P perpendicular to one of the original coordinate axes with the following properties: (1) Both of the subcollections A and B induced by P contain at least N/4k of the points. (2) There are at most $kcN^{1-1/k}$ points within distance δ of P.

- **If we could show that the problem size in the conquer step is $m \leq n/(\log n)^{d-2}$, then $U(m, d-1) = O(m(\log m)^{d-2}) = O(n)$.**

- **Theorem: Given a set S with δ-sparsity, there exists a hyperplane H normal to some axis such that**

1. $|S_1|, |S_2| \geq n/4d$.
2. **Number of points within δ of H is $O(\frac{n}{(\log n)^{d-2}})$.**
3. **H can be found in $O(n)$ time.**

- **If we could show that the**
  the conquer step is $m \leq n/$
  $U(m, d-1) = O(m(\log m)^{d-2})$

- **Theorem: Given a set** S **with**
  there exists a hyperplane H **n**
  some axis such that

1. $|S_1|, |S_2| \geq n/4d$.
2. **Number of points within** δ **of** H **is** $O(\frac{n}{(\log n)^{d-2}})$.
3. H **can be found in** $O(n)$ **time.**

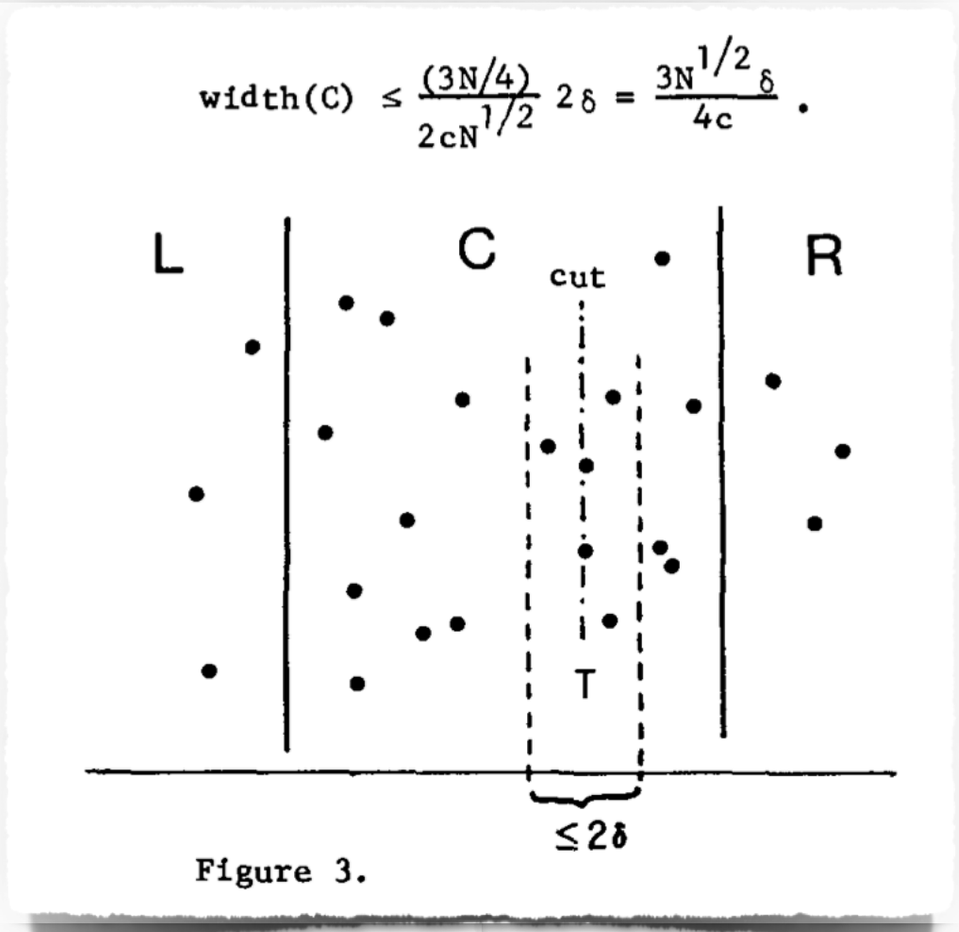


Figure 3.

- **The divide and conquer algorithm now satisfies the recurrence** $T(n,d) = 2T(n/2,d) + U(m,d-1) + O(n)$.
- **By new sparsity claim,** $m \leq n/(\log n)^{d-2}$, **and so** $U(m,d-1) = O(m(\log m)^{d-2}) = O(n)$.
- **Thus,** $T(n,d) = 2T(n/2,d) + O(n) + O(n)$, **which solves to** $O(n \log n)$.
- **Solves the Closest Pair problem in fixed** d **in optimal** $O(n \log n)$ **time.**

- **If we could show that the problem size in the conquer step is $m \leq n/(\log n)^{d-2}$, then $U(m, d-1) = O(m(\log m)^{d-2}) = O(n)$.**

Theorem 8. $P(N,k) \leq O(N \log N)$.

Thank you for today!