Quicksort and Randomized Incremental Constructions

Instructor: Thomas Kesselheim and Kurt Mehlhorn

1 Quicksort

This is Section 5.4 in Mehlhorn/Sanders [DMS14, MS08]

Quicksort is a divide-and-conquer algorithm. Let S be the set to be sorted. We select a uniformly random element p from S and split S into three parts. The set $S_<$ of elements smaller than p, the set $S_=$ of elements equal to p and the set $S_>$ of elements larger than p. The element p is usually called the pivot. Then we apply the algorithm recursively to $S_<$ and $S_>$.

1.1 Analysis

To analyze the running time of quicksort for an input sequence $s = \{e_1, \ldots, e_n\}$, we focus on the number of element comparisons performed. We allow *three-way* comparisons here, with possible outcomes "smaller", "equal", and "larger". Other operations contribute only constant factors and small additive terms to the execution time.

Let $C(n)$ denote the worst-case number of comparisons needed for any input sequence of size n and any choice of pivots. The worst-case performance is easily determined. The subsequences $S_<$, $S_=$, and $S_>$ are formed by comparing the pivot with all other elements. This requires $n-1$ comparisons. If we use k and k' to denote the number of elements smaller, respectively larger, than the pivot, we obtain the following recurrence relation: $C(0) = C(1) = 0$ and

$$C(n) \leq n - 1 + \max\{C(k) + C(k') \mid 0 \leq k \leq n-1, 0 \leq k' < n - k\}.$$

It is easy to verify by induction that

$$C(n) \leq \frac{n(n-1)}{2} = \Theta(n^2).$$

The worst case occurs if all elements are different and we always pick the largest or smallest element as the pivot. Thus $C(n) = n(n-1)/2$.

The expected performance is much better. We first show a bound of $2n \ln n$ and then argue that the running time is sharply concentrated around the expectation. We concentrate on the case where all elements are different. Other cases are easier because a pivot that occurs several times results in a larger middle sequence $S_=$ that need not be processed any further.

Theorem 3.1. *The expected number of comparisons performed by quicksort is*

$$\bar{C}(n) \leq 2n \ln n \leq 1.39 n \log n.$$

Proof. Let $e'_1, \ldots, e'_n$ denote the elements of the input sequence in sorted order. Every comparison involves a pivot element. If an element is compared to a pivot, the pivot and the element end up in different subsequences. Hence any pair of elements is compared at most once, and we can therefore count comparisons by looking at the indicator random variables X_{ij}, $i < j$, where $X_{ij} = 1$ if e'_i and e'_j are compared and $X_{ij} = 0$ otherwise. We obtain

$$\bar{C}(n) = \mathbf{E}\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathbf{E}\left[X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathbf{Pr}\left[X_{ij} = 1\right].$$

The middle transformation follows from the linearity of expectations. The last equation uses the definition of the expectation of an indicator random variable $\mathbf{E}[X_{ij}] = \mathbf{Pr}[X_{ij} = 1]$. Before we can further simplify the expression for $\bar{C}(n)$, we need to determine the probability of X_{ij} being 1.

Lemma 3.2. *For any $i < j$, $\mathbf{Pr}[X_{ij} = 1] = \dfrac{2}{j - i + 1}$.*

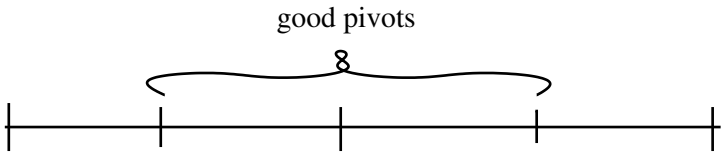

Figure 1: The line segment represents the current set in sorted order. If the pivot is chosen from the middle half, both subproblems have size at most 3/4 times the size of the current set.

Proof. Consider the $j-i+1$-element set $M = \{e'_i, \ldots, e'_j\}$. As long as no pivot from M is selected, e'_i and e'_j are not compared, but all elements from M are passed to the same recursive calls. Eventually, a pivot p from M is selected. Each element in M has the same chance $1/|M|$ of being selected. If $p = e'_i$ or $p = e'_j$, we have $X_{ij} = 1$. Otherwise, e'_i and e'_j are passed to different recursive calls, so that they will never be compared. Thus $\mathbf{Pr}[X_{ij} = 1] = 2/|M| = 2/(j-i+1)$. □ □

We can now complete the proof of Theorem 3.1 by a relatively simple calculation:

$$\begin{aligned}\bar{C}(n) &= \sum_{i=1}^{n}\sum_{j=i+1}^{n}\mathbf{Pr}[X_{ij}=1] = \sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{2}{j-i+1} = \sum_{i=1}^{n}\sum_{k=2}^{n-i+1}\frac{2}{k}\\ &\le \sum_{i=1}^{n}\sum_{k=2}^{n}\frac{2}{k} = 2n\sum_{k=2}^{n}\frac{1}{k} = 2n(H_n-1) \le 2n(1+\ln n-1) = 2n\ln n.\end{aligned}$$

For the last three steps, recall the properties of the n-th harmonic number $H_n := \sum_{k=1}^{n} 1/k \le 1+\ln n$. □

1.2 Sharp Concentration

Consider a fixed element e, and let X_e denote the total number of times e is compared with a pivot element. Note that we charge a comparison between an element and a pivot to the element. There is no charge to the pivot. Then $\sum_e X_e$ is the total number of comparisons. Whenever e is compared with a pivot element, it ends up in a smaller subproblem. Therefore, $X_e \le n-1$, and we have another proof for the quadratic upper bound.

Let us call a comparison "good" for e if e moves to a subproblem of at most three-quarters the size. Any e can be involved in at most $\lfloor \log_{4/3} n \rfloor$ good comparisons.[1] The probability that a pivot which is good for e is chosen, is at least $1/2$; this holds because any pivot chosen from the middle half of the set is good; see Figure 1. So[2] $\mathbf{E}[X_e] \le 2\log_{4/3} n$, and hence $\mathbf{E}[\sum_e X_e] \le 2n\log_{4/3} n$. Note that $\log_{4/3} n \le 3.5\ln n$ and $\log_{4/3} n \le 2.41\log n$.

This analysis can be sharpened to show that the number of comparisons in $O(n\log n)$ with high probability. Intuitively, the argument is as follows. We will see the precise argument in a later lecture. Let $k = \lfloor \log_{4/3} n \rfloor$. We consider a sequence of $m = 10\cdot k$ coin tosses. A coin comes up head with probability $1/2$. On average, we will see $m/2$ heads. The probability that we see at most k heads will be small. Indeed, let p_i be the probability that we see exactly i heads. Then[3][4]

$$\begin{aligned}p_k &= \binom{m}{k}2^{-m} \le \frac{m^k}{k!}2^{-11k} \le \frac{m}{(k/e)^k}2^{-10k} = \left(\frac{e\cdot 10\cdot k}{k\cdot 2^{10}}\right)^k \le \left(\frac{30}{1000}\right)^k \le \left(\frac{1}{32}\right)^k\\ &\le \left(\frac{1}{2}\right)^{5\cdot((\log_{4/3} n)-1)} \le 2^5\left(\frac{1}{n}\right)^{5/\log 4/3} \le 2^5\frac{1}{n^{10}}.\end{aligned}$$

and for $i < k$

$$\frac{p_i}{p_{i+1}} = \frac{\binom{m}{i}}{\binom{m}{i+1}} = \frac{m(m-1)\cdots(m-i+1)\cdot(i+1)!}{m(m-1)\cdots(m-i)\cdot i!} = \frac{i+1}{m-i} \le \frac{k}{m-k} \le \frac{1}{9}.$$

[1] After k good comparisons for e, the set containing e has cardinality at most $(2/3)^k n$. Thus $k \le (\log n)/\log(3/2)$.

[2] Formaly, we should define $X_e = \sum_{1\le j\le \log_{4/3} n} X_{ej}$, where X_{ej} is the number of comparisons after the $j-1$-th good comparison and up to and including the j-th good comparison. Define $X_{ej} = 0$ if the element e is already in a singleton set after the $j-1$-th good comparison. Since a comparison is good with probability at least $1/2$ we have $\mathbf{E}[X_{ej}] \le 2$ (compare the first lecture).

[3] We use $k^k/k! \le \sum_i k^i/i! = e^k$ and hence $k! \ge (k/e)^k$.

[4] $\log 3 \approx 1.58$.

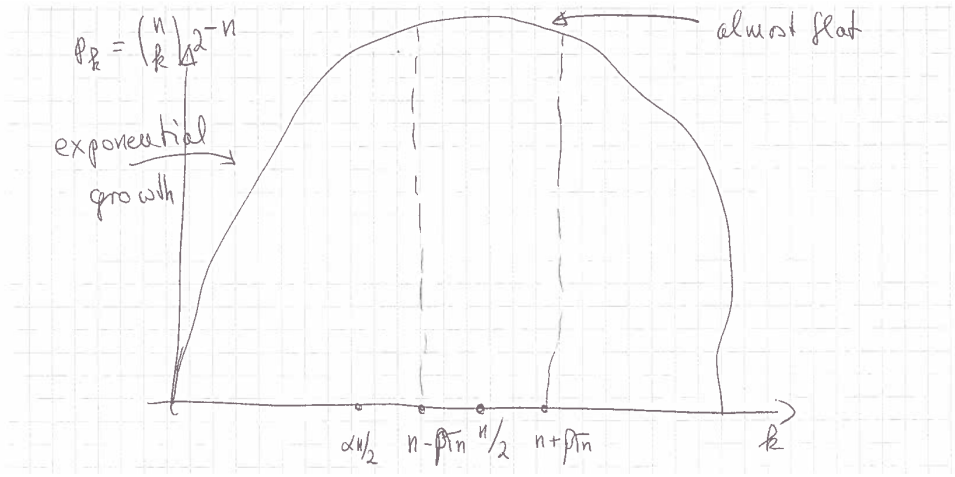

Figure 2: The binomial distribution.

Thus $\sum_{i \le k} p_i \le 2p_k \le 2^6/n^{10}$. We conclude that the probability that a particular element is involved in more than $10 \cdot k$ comparisons is $O(1/n^{10})$ and hence the probability that some element is involved in $10 \cdot k$ comparisons is $O(1/n^9)$.

Figure 2 illustrates the binomial distribution. Let $\alpha < 1$ be arbitrary. For $k \le \alpha n/2$, the p_k grow exponentially. Indeed,

$$\frac{p_k}{p_{k-1}} = \frac{n-k+1}{k} \ge \frac{(1-\alpha/2)n}{\alpha/2n} = \frac{2}{\alpha} - 1.$$

In the interval $n - \beta\sqrt{n}$ to $n + \beta\sqrt{n}$, where β is a constant, the distribution is essentially flat. Indeed, (for simplicity, I assume that n is even and $\beta \ge 1/4$ and $\beta \le \sqrt{n}/8$)

$$\begin{aligned}
\frac{p_{n/2-\beta\sqrt{n}}}{p_{n/2}} &= \frac{n(n-1)\cdots(n/2+\beta\sqrt{n}+1)}{(n/2-\beta\sqrt{n})!} \cdot \frac{(n/2)!}{n(n-1)\cdots n/2} \\
&= \frac{n/2(n/2-1)\cdots(n/2-\beta\sqrt{n}+1)}{(n/2+1)\cdots(n/2+\beta\sqrt{n})} \\
&\ge \left(\frac{n/2-\beta\sqrt{n}+1}{n/2+\beta\sqrt{n}}\right)^{\beta\sqrt{n}} \\
&= \left(\frac{1-\frac{2\beta}{\sqrt{n}}+\frac{2}{n}}{1+\frac{2\beta}{\sqrt{n}}}\right)^{\beta\sqrt{n}} = \left(1-\frac{\frac{4\beta}{\sqrt{n}}+\frac{2}{n}}{1+\frac{2\beta}{\sqrt{n}}}\right)^{\beta\sqrt{n}} \\
&\ge \left(1-\frac{4\beta}{\sqrt{n}}\right)^{\beta\sqrt{n}} && \text{true for } \beta \ge 1/4 \\
&= \left(\left(1-\frac{4\beta}{\sqrt{n}}\right)^{\sqrt{n}/(4\beta)}\right)^{4\beta^2} \\
&= \left(\left(1-\frac{4\beta}{\sqrt{n}}\right)\cdot\left(1-\frac{4\beta}{\sqrt{n}}\right)^{\sqrt{n}/(4\beta)-1}\right)^{4\beta^2} \\
&\ge \left(\left(1-\frac{4\beta}{\sqrt{n}}\right)\cdot\frac{1}{e}\right)^{4\beta^2} && \text{Fact 1.2 (b) of first lecture} \\
&\ge \cdot(2e)^{-4\beta^2} && \text{uses } 4\beta/\sqrt{n} \le 1/2
\end{aligned}$$

1.3 Backwards Analysis

Let π be a permutation of the integers 1 to n. We associate a run of quicksort on the integers 1 to n with π. Store the integers in sorted order in an array and color all elements white. We scan the permutation

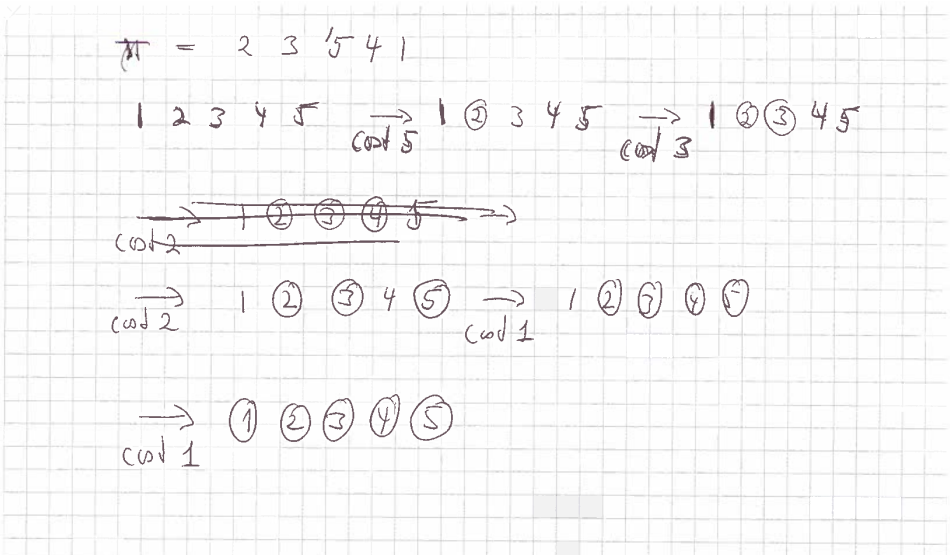

Figure 3: The execution of quicksort corresponding to the permutation $\pi = (2, 3, 5, 4, 1)$.

from left to right. When we encounter $\pi(i)$, we change the color of $\pi(i)$ from white to black and charge a cost equal to the size of the white interval containing $\pi(i)$, see Figure 3 for an example.

For the analysis, we run the execution backwards. Initially all elements are black and the permutation is completely undefined. As long as there is a black element, we choose a random black element, change its color from black to white and push the element to the front of the permutation. The cost of changing an element from black to white is the length of the white interval formed. Observe that we construct a random permutation in this way and executing quicksort with this permutation as described in the first paragraph is simply the reversal of the process described in the second paragraph.

For the analysis, consider the situation when there are still k black elements. They split the array into white intervals. The total length of the white intervals is $n - k$. When we change the color of a black element, the cost is the length of the white interval formed. Let B_k be the set of black elements and let X_k be the length of the white interval formed. Then

$$\begin{aligned}
\mathbf{E}[X_k] &= \frac{1}{k} \sum_{x \in B_k} \text{length of white interval formed when } x \text{ is recolored} \\
&= \frac{1}{k} \sum_{x \in B_k} (1 + \text{length of white interval left of } x \text{ (if any)} + \text{length of white interval right of } x \text{ (if any)}) \\
&\leq \frac{1}{k} (k + 2(n - k)) \\
&\leq \frac{2n}{k}.
\end{aligned}$$

The first inequality follows from the fact that each current white interval is counted at most twice, once for the black element following it and once for the black element preceding it. Thus

$$\mathbf{E}\left[\sum_{1 \leq k \leq n} X_k\right] \leq 2nH_n.$$

2 Randomized Incremental Constructions

Randomized incremental constructions (RICs) are a generalization of quicksort to the geometric setting. For many geometric problems, e.g., convex hulls in arbitrary dimension, Voronoi diagrams, Delaunay diagrams, the RIC paradigm leads to *simple* algorithms that at least match the running time of the best deterministic algorithms. RICs were introduced by Clarkson and Shor [CS89]. Our colleague Raimund

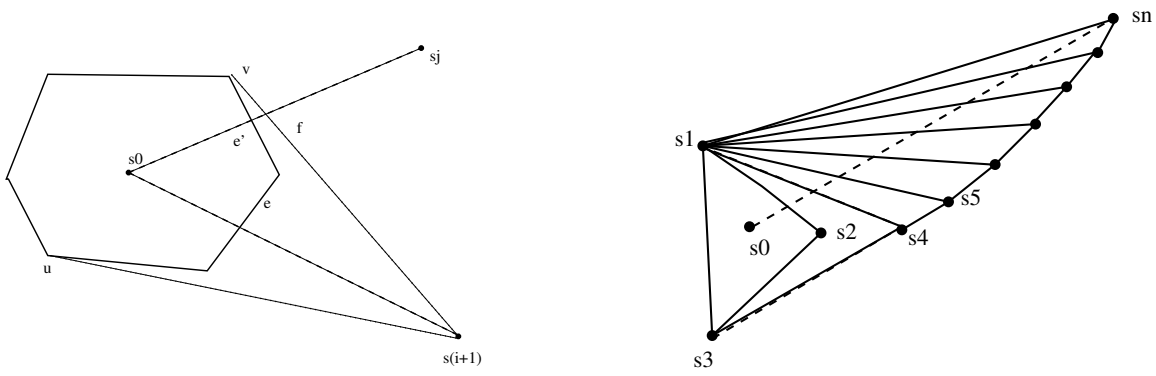

Figure 4: The left part shows the addition of s_{i+1} to S_i. The point s_j was associated with the edge e' of S_i and becomes associated with the edge f of S_{i+1}.
The right part shows an insertion sequence with quadratic cost. Note that the ray s_0s_n intersects the edge s_1s_2 of $conv(S_3)$, the edge s_1s_4 of $conv(S_4)$, ..., and the edge s_1s_{n-1} of $conv(S_{n-1})$. Generally, the ray s_0s_i intersects distinct hull edges of $conv(S_3)$ to $conv(S_{i-1})$.
When the points s_4 to s_n are inserted in random order, the association of s_n is changed only $O(\log n)$ times in expectation.

Seidel [Sei91] much simplified the analysis of RICs by introducing the technique of backward analysis. It is by now the standard method for analyzing RICs.

We exemplify RICs on the convex hull problem in the plane. We are given a set S of points in the plane and want to computer their convex hull $conv(S)$. We assume that the points in S are in general position, i.e., no three lie on a common line.

We consider the points in turn. Let S_i be the set formed by the first i points. The first three points form a triangle; let s_0 be a point in the interior of this triangle. We maintain the following data structures:

- The vertices of $conv(S_i)$ in counterclockwise order in a cyclic list.
- For each $s \in S \setminus S_i$, a pointer to the edge of $conv(S_i)$ that is intersected by the ray s_0s (if any). For every edge e of $conv(S_i)$, we keep the list of points $s \in S \setminus S_i$ for which e is intersected by the ray s_0s.

The data structure is readily initialized for $i = 3$. As s_0 we may choose the center of gravity of the points in S_3.

Let us next see how we can go from S_i to S_{i+1}. The point s_{i+1} lies inside or outside $conv(S_i)$. We distinguish these cases by inspecting the pointer associated with s_{i+1}. If the pointer is nil, s_{i+1} lies inside $conv(S_{i-1}$ and nothing needs to be done. Otherwise, it points to an edge e of $conv(S_i)$ that is intersected by the ray s_0s_{i+1}. Walking from e in both directions along the boundary of $conv(S_i)$, we can find the two vertices u and v of $conv(S_i)$ that together with s_{i+1} form new hull edges of $conv(S_{i+1})$.

We update our data structure as follows:

- We remove all vertices between u and v from the cyclic list of hull vertices and add s_{i+1} in their place.
- For each edge e that we removed from $conv(S_i)$ and each s in the list associated with s, we check whether the ray s_0s intersects one of the new hull edges. If so we store a pointer to the edge with s (otherwise we set the pointer to nil) and associate s with the edge.

This ends the description of the algorithm. You must agree that it is a simple algorithm.

We come to the analysis. What is the cost of adding $s = s_{i+1}$? If s lies inside $conv(S)$, the cost of adding s is one. If s lies outside, we first determine the edges that are to be removed from the hull. If there are k edges to be removed, we find them in time $O(k)$. We also add two new edges. Thus the total number of edges added is bounded by $2n$ and the number of edges removed can be no more. Thus the amortized cost of deleting and constructing hull edges is $O(1)$ be insertion.

Finally, for each edge removed, we need to look at all points associates with it. There can be many. In the worst-case all points still to be added are associated with the removed edges and hence the cost of the i-th insertion will be $\Omega(n - i)$. Figure 4 shows an example.

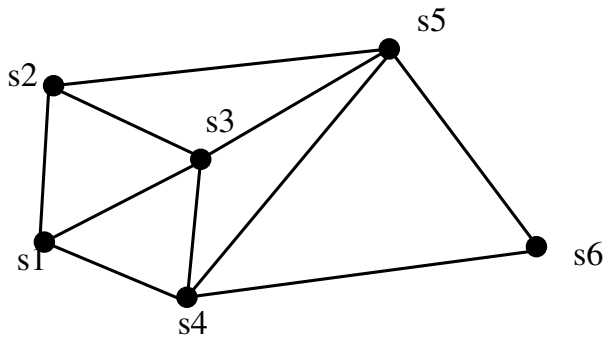

Figure 5: Higher-dimensional hull algorithm

For the probabilistic analysis we assume that the points are inserted in random order, i.e., any of the $n!$ ordering of the points are equally likely. We partition the insertion orders into classes according to the first three points that are inserted. For the insertion orders in a class, the point s_0 is fixed. Consider now a particular point $s \in S \setminus S_3$. How often does the hull edge intersecting the ray $s_0 s$ change? *Imagine that we run the construction backwards*, i.e., we start with $conv(S_n)$ and then remove the points one by one. Assume we are back to $conv(S_{i+1})$ and delete s_{i+1}. If s_{i+1} is not a vertex of the hull, nothing changes. If s_{i+1} is a vertex of the hull, two edges disappear and a chain of edges appears. How does this effect the ray $s_0 s$. The edge intersected by this ray changes only if s_{i+1} is equal to one of the endpoints of the hull edge intersected by the ray. The probability for this is $2/(i+1-3)$. Recall that we fixed S_3 and note that $|S_{i+1} \setminus S_3| = i+1-3$. Thus the expected number of distinct hull edges ever intersected by the ray $s_0 s$ is bounded by

$$1 + \sum_{3 \le i \le n-1} \frac{2}{i+1-3} \le 2H_n,$$

where the 1 counts the edge of the initial triangle intersected by the ray. Since s was arbitrary, the total expected cost of updating the edge-ray associations is $O(n \log n)$.

Theorem 3.3. *The algorithm of this section constructs the convex hull of n points in general position in the plane in expected time $O(n \log n)$.*

The theorem above is also true without the general position assumption [MN99, Section 10.1]. This is open for RICs in general.

The algorithm above extends to higher dimensions. The algorithm maintains a triangulation of $conv(S_i)$, i.e., a partition of $conv(S_i)$ into simplices. It starts with the simplex formed by the first $d+1$ points, where d is the dimension of the space. When a new point $s = s_{i+1}$ is to be inserted, we proceed as follows. Let again s_0 be a point in the first simplex.

1. We walk along the ray $s_0 s$ through the triangulation. We start the walk in the simplex containing s_0. Let C be the current simplex. We determine whether the ray leaves the simplex. If it does not leave the simplex, we are done. Otherwise, let f be the facet of C through which the ray leaves C. If f is a facet of $conv(S_i)$, we proceed to step 2. Otherwise, let C be the simplex on the other side of f.

2. Starting from f, we find the set F of all facets of $conv(S_i)$ that cease to be hull facets.

3. For each $f' \in F$, we add the simplex $S(f', s)$ to the triangulation.

See Figure 5 for an illustration and [CMS93] for the analysis.

We refer out students to the textbooks in computation geometry, for example [dBKOS97] and Motwani/Raghavan, for further applications of the paradigm.

References

[CMS93] K. Clarkson, K. Mehlhorn, and R. Seidel. Four Results on Randomized Incremental Constructions. *Computational Geometry: Theory and Applications*, 3:185–212, 1993.

[CS89] K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Journal of Discrete and Computational Geometry*, 4:387–421, 1989.

[dBKOS97] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 1997.

[DMS14] Martin Dietzfelbinger, Kurt Mehlhorn, and Peter Sanders. *Algorithmen und Datenstrukturen - die Grundwerkzeuge*. Springer, 2014. German translation of Mehlhorn/Sanders.

[MN99] K. Mehlhorn and S. Näher. *The LEDA Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[MS08] K. Mehlhorn and P. Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Springer, 2008. Translations into German, Greek, Japanese, and Chinese.

[Sei91] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete and Computational Geometry*, 6:423–434, 1991.