On tail estimates for Randomized Incremental Construction

Sandeep Sen
Department of CSE,
I.I.T. Delhi, India
ssen@cse.iitd.ac.in

August 8, 2018

Abstract

By combining several interesting applications of random sampling in geometric algorithms like point location, linear programming, segment intersections, binary space partitioning, Clarkson and Shor [4] developed a general framework of randomized incremental construction (RIC). The basic idea is to add objects in a random order and show that this approach yields efficient/optimal bounds on expected running time. Even quicksort can be viewed as a special case of this paradigm. However, unlike quicksort, for most of these problems, attempts to obtain sharper tail estimates on the running time had proved inconclusive. Barring some results by [14, 3, 19], the general question remains unresolved.

In this paper we present some general techniques to obtain tail estimates for RIC and provide applications to some fundamental problems like Delaunay triangulations and construction of Visibility maps of intersecting line segments. The main result of the paper centers around a new and careful application of Freedman’s [9] inequality for Martingale concentration that overcomes the bottleneck of the better known Azuma-Hoeffding inequality. Further, we show instances where an RIC based algorithm may not have inverse polynomial tail estimates. In particular, we show that the RIC time bounds for trapezoidal map can encounter a running time of $\Omega(n \log n \log \log n)$ with probability exceeding $\frac{1}{\sqrt{n}}$. This rules out inverse polynomial concentration bounds around the expected running time.

1 Introduction

One of the most natural and elegant paradigm for designing geometric algorithms is randomized incremental construction or RIC for short. It can be viewed as generalization of Quicksort and evolved over a sequence of papers [17, 2] eventually culminating in a very general framework of configuration space by Clarkson and Shor [4]. The basic procedure is described in Figure 1. Quicksort itself can be viewed through this paradigm as refinement of the current partially ordered set (partitions) by inserting the next splitter and updating the partitions. Some of the uninserted elements are further partitioned because of the latest insertion. Although the worst case deterministic behavior can be quite bad, the expected performance for a random insertion sequence (where are permutations are equally likely) is quite efficient, and often optimal.

A related but a somewhat distinct approach was developed in the work of Seidel [18, 19, 1] that maintains a solution inductively that is recomputed from scratch when the solution does not hold for the current insertion. The closest pair can also be computed in a similar manner [12]. Although our techniques can be applied to the latter work also, we will focus primarily on the Clarkson-Shor paradigm of a configuration space.
An abstract configuration space, that we will refer to as $\Pi(S)$ is defined by the given set $S$ of $n$ elements. A configuration $\sigma$ is defined by $O(1)$ objects of $S$ that we will denote by $d(\sigma)$. A configuration $\sigma$ is a subset of the Euclidean space and $\ell(\sigma) = \sigma \cap \{S - d(\sigma)\}$, i.e. the objects that intersect $\sigma$ not including $d(\sigma)$. $\Pi(1)(S) = \{\sigma : |\ell(\sigma)| = 1\}$ and $\Pi(S) = \bigcup_{i=0}^{n} \Pi(1)(S)$. For analyzing $RIC$, $\Pi(0)(R)$ where $R \subset S$ is a randomly chosen subset often turns out to be very important, that captures the uninserted elements of $S$ and how they interact with the current partially constructed structure, denoted by $H(R)$. For notational simplicity, for $\sigma \in \Pi(0)(R)$, $\ell(\sigma) = \sigma \cap S$ (instead of $\sigma \cap R$) which will be an important parameter in the analysis. The reader is referred to [14, 16] for further details regarding this framework.

When the next randomly chosen element $s \in S - R$ is included in $R$, $H(R)$ is updated and the cost of this contributes to the running time of $RIC$. In [4], the data-structure is maintained as a conflict graph that maintains relation between $\sigma \in \Pi(0)(R)$ and $S - R$ as a bipartite graph. Clearly configurations are created and destroyed but the amortized cost can be shown to be the cost of new configurations created and the ones destroyed can be charged to the cost of past creation. Although the initial analysis in [4] was somewhat intricate and complex, subsequent papers [11, 18] simplified the analysis using a clever technique called backward analysis. In this paper, we will appeal to the simpler analysis. Often the full conflict graph information can be replaced by simpler relations (see [10, 18]). However, the conflict graph approach is very general and works for diverse problems.

For analyzing $RIC$ based algorithms, Clarkson and Shor [4] derived many useful bounds based on properties of uniform random sampling that generalized the results of Haussler and Welzl [11] that essentially gave a bound on $\max_{\sigma \in \Pi(0)} |\ell(\sigma)|$. We will exploit such properties in the present paper - for proofs the reader can consult [4, 16]. Henceforth $\ell(\sigma)$ will also be used as a notation for $|\ell(\sigma)|$.

While the primary focus was on deriving bounds on the expected running time of $RIC$, it was felt that obtaining concentration bounds on the expected running time would make the $RIC$ more powerful and attractive. The conjecture is that the running times are concentrated around its expected value but to the best of our knowledge, there has been little progress in this direction barring some papers related to computing line segment intersections using $RIC$ [3, 14] and on fixed dimensional linear programming [19]. It is also known that for problems like planar hulls, high probability bounds can be proved based on linear ordering that do not extend to higher dimensions. Of course, by resampling $\Omega(\log n)$ times, we can obtain inverse polynomial concentration bounds at the expense of the increasing the running time by an $O(\log n)$ factor.

In this paper, we revisit the problem and present some general methods to obtain tail bounds for specific problems like Delaunay triangulation, 3-D convex hulls, and line segment intersections that are based on $RIC$. We obtain tail estimates of the form $2^{-\alpha}$ for an $\alpha$ factor deviation from the expected running times. This is similar to the bounds for resampling but it doesn’t involve independent restarts of the algorithm. For the case of finding intersection of line segments, our bounds are not only better than [14] but also distinctly less involved in terms of calculations.

---

1 We adopt some of the notations from [10] and for completeness, we include some formal definitions in the Appendix.
We also establish the tightness of such tail estimates by demonstrating a case of trapezoidal maps (based on maintaining conflict lists) for which inverse polynomial tail estimates can be obtained only for running times \( \Omega(n \log n \log \log n) \) and rules out concentration bounds within constant factor of expectation.

For analysis, we use Martingale inequalities based on the method of \textit{bounded variance} (as opposed to bounded difference) and the basic martingale set up follows that of \cite{14}.

\textbf{Remark} In RIC\textsuperscript{\textregistered} based algorithms, the term \textit{running time} is often interchangeably used with \textit{structural changes} caused by each insertion. The data structures are consciously kept minimal and simple (like lists) that enable the running times to be proportional to the structural changes that are explicitly handled. In this paper, we will use the term \textit{work} to denote structural changes and we will not attempt to analyze the precise running times.

1.1 Main techniques and organization

We begin by introducing a useful probabilistic inequality, viz., Freedman’s inequality \cite{9} for Martingales that will be used to model the running time of the generic RIC algorithms. In the following section, we illustrate the use of this analysis technique on the classical algorithm quicksort that can be also viewed through the lens of RIC. The application to quicksort doesn’t yield any better result but is a stepping stone to the more complex and general framework. In particular, even the more commonly used Azuma-Hoeffding bound is not known to be useful for quicksort concentration bounds because its dependence on the worst-case bound (sum of bounded differences) making it ineffective.

It is unlikely that the previously effective techniques for concentration bound of quicksort extend to generic RIC because the intermediate structures in RIC are more complex and can be bounded only in an \textit{expected} sense. Note that in sorting, the intermediate structures can be defined by exactly \( i \) intervals after \( i \) pivots are introduced. In the generic RIC the intermediate data-structures may become much larger which explains why similar concentration bounds are hard to obtain. The Freedman’s inequality is more effective since it uses variance (expectation of the second moment) for which better bounds can be obtained for the \( i \)-th step compared to the worst case.

Starting with quicksort in section \textsection 3 we tackle increasingly complex scenarios of RIC which can viewed as weaker bounds on the intermediate structures for which we are trying to obtain concentration bounds. In the case of Delaunay triangulation, in section \textsection 4 the number of triangles in the \( i \)-th step is fixed but the number of new triangles created in the \( i \)-th step can be bound only in expectation. In section \textsection 5 we consider the case of line segment intersections where even the intermediate structure can be bound only in an expected sense.

In the last section, we give concrete examples of RIC to show that inverse polynomial concentration bounds are not feasible without changing some basic structure of the algorithm.

2 Basic Tools

Let \( S = \{x_1, x_2, \ldots x_n\} \) be a set of \( n \) objects. A permutation \( \pi \) of \( S \) is a 1-1 function \( \pi(i) = j \) where \( i, j \in \{1, 2, \ldots n\} \) that produces a permutation \( x_{\pi(1)}, x_{\pi(2)} \ldots x_{\pi(n)} \). A \textit{random} permutation of \( S \) is one of the \( n! \) permutation function chosen uniformly at random. A \textit{k prefix} of a permutation \( \pi \) is the sequence of the first \( k \) objects and denoted by \( \pi^{(k)} \) consisting of \( x_{\pi^{-1}(1)} x_{\pi^{-1}(2)} \ldots x_{\pi^{-1}(k)} \). Note that the permutation \( x_3, x_1, x_2 \) is defined as \( \pi(1) = 2; \pi(2) = 3; \pi(3) = 1 \), so the permutation is \( x_{\pi^{-1}(1)} x_{\pi^{-1}(2)} x_{\pi^{-1}(3)} \).

Let \( X_1, X_2, \ldots X_n \) \( X_i \neq \{X_1, X_2, \ldots X_{i-1}\} \) where \( X_i = x_{\pi^{-1}(i)} \) corresponding to the random permutation \( \pi \). Further, let \( \tilde{X}^{(k)} \) to denote a sequence of \( k \) random variables.

Let \((\Omega, \mathcal{U})\) denote the space of all possible permutations of \( n \) objects and \( \mathcal{U} \) is the uniform probability distribution. For \( 0 \leq i \leq n \), let \( \mathcal{F}_i \) consist of all permutations with fixed prefixes of length \( i \) and set
\[ F_0 = \epsilon \text{ (empty prefix). Then, } F_i \text{ contains } \frac{\binom{n}{i}}{(n-i)!} \text{ blocks corresponding to each of the } i \text{ length prefixes. For example, if the set of objects is } \{x_1, x_2, x_3\}, \text{ then the collection of events are as follows.} \]

\[
F_0 : \{(x_1 x_2 x_3, x_1 x_3 x_1, x_2 x_1 x_3, x_2 x_3 x_1, x_3 x_1 x_2, x_3 x_2 x_1) \}
\]

\[
F_1 : \{(x_1 x_2 x_3, x_1 x_3 x_2), (x_2 x_1 x_3, x_2 x_3 x_1), (x_3 x_1 x_2, x_3 x_2 x_1) \}
\]

\[
F_2 : \{(x_1 x_2 x_3), (x_1 x_3 x_2), (x_2 x_1 x_3), (x_2 x_3 x_1), (x_3 x_1 x_2), (x_3 x_2 x_1) \}
\]

The blocks within each \( F_i \) are indicated by \( ( ) \). More precisely \( F_i \) is a sigma algebra of the corresponding events that can be enumerated explicitly, but omitted for brevity. It can be easily verified that \( F_{i+1} \) is a refinement of \( F_i \) where each block of \( F_i \) is partitioned into \( n - i \) subpartitions of \( F_{i+1} \). These nested subcollections of \( 2^{\Omega_i} \) define a filter denoted by \( F_0 \subseteq F_1 \subseteq \ldots \subseteq F_{i-1} \subseteq F_i \ldots \subseteq F_n \) that can be used to define a sequence of random variables \( Y_i \) where \( Y_i \)'s are functions of \( F_i \)'s. In particular, if the \( Y_i \)'s are \( F_i \) measurable, and \( \mathbb{E}[Y_{i+1}|F_i] = Y_i \), then \( Y_i \) is a martingale sequence [7, 8].

In this context, let us define \( Y_i = \mathbb{E}[Y|\bar{X}(i)] \) for any well-defined random variable \( Y \) over the probability space \((\Omega, \mathcal{U})\) where the conditioning is over the events in \( F_i \). Then, it can be verified that

\[
\mathbb{E}[Y_i] = \mathbb{E}[\mathbb{E}[Y|\bar{X}(i)]] = \mathbb{E}[\mathbb{E}[Y|\bar{X}(i-1)|X_i]] = \mathbb{E}[Y|\bar{X}(i-1)] = Y_{i-1}
\]

since \( F_i \) is a refinement of \( F_{i-1} \). The sequence \( Y_i \) defines a martingale sequence and is widely known as a Doob Martingale [6]. It is more intuitive to visualize the above filter as a tree, where the level \( i \) nodes correspond to \( F_i \) with arity \( n - i \) and each sub-block is connected to its parent block by an edge directed from the parent. Any node in the \( j \)-th level of this tree can be labelled by the (unique) sequence \( X(j) \) leading to it.

In the context of analyzing RIC, it may be useful to club prefixes that result in the same subset. For example, the length two prefixes of \([x_1, x_2, x_3], [x_2, x_1, x_3]\) lead to the subset \([x_1, x_2]\). Similar to the previous example, the blocks corresponding to distinct subsets can be enumerated as

\[
F'_0 : \{(x_1 x_2 x_3, x_1 x_3 x_1, x_2 x_1 x_3, x_2 x_3 x_1, x_3 x_1 x_2, x_3 x_2 x_1) \}
\]

\[
F'_1 : \{(x_1 x_2 x_3, x_1 x_3 x_2) : \{x_1\}, (x_2 x_1 x_3, x_2 x_3 x_1) : \{x_2\}, (x_3 x_1 x_2, x_3 x_2 x_1) : \{x_3\} \}
\]

\[
F'_2 : \{(x_1 x_2 x_3, x_2 x_1 x_3) : \{x_1, x_2\}, (x_1 x_3 x_2, x_3 x_1 x_2) : \{x_1, x_3\}, (x_2 x_1 x_3, x_2 x_3 x_1) : \{x_2, x_3\} \}
\]

\[
F'_3 : \{(x_1 x_2 x_3, x_1 x_3 x_1, x_2 x_1 x_3, x_2 x_3 x_1, x_3 x_1 x_2, x_3 x_2 x_1) : \{x_1, x_2, x_3\} \}
\]

The subsets corresponding to the blocks are indicated with curly brackets. The reader may notice that these blocks do not generate a nested sequence of sigma algebra - for instance the subset \([x_1, x_2]\) defined by the block \((x_1 x_2 x_3, x_1 x_3 x_2)\) in \( F'_0 \) come from different blocks of \( F'_i \) depending on whether \( x_1 \) or \( x_2 \) occurs before. Therefore, this collection of prefixes on the insertion sequence is not consistent with a martingale on collection of subsets.

Instead, let us interpret the sequences as deletion sequence, viz., starting from \([x_1, x_2, x_3]\), we will delete the elements according to a random permutation, finally leading to the empty say \( \phi \). Then \( F'_i \) denotes the subsets corresponding to deletion of \( x_1, x_2, x_3 \) respectively. This way, the blocks (of subsets) form a nested sequence. As we will see, this interpretation leads to running the RIC in the reverse direction with each step known in the literature as backward analysis which simplifies analysis considerably in many situations [20].

Let random variables \( X_1, X_2, \ldots \), denote the successive random choices in this tree starting from the root where the first \( i \) choices correspond to the blocks of \( F_i \) for \( 0 \leq i \leq n \). An edge is labelled by the (random) choice made at that level and also has an associated weight \( w(\bar{X}(i-1), \bar{X}(i)) \) that corresponds to the cost of the \( i \)-th incremental step. We will use \( W() \) to denote an upper bound of \( w() \) in the context of specific algorithms. Let \( Y = \sum_{j=0}^{n} w(\bar{X}(j-1), \bar{X}(j)) \) be a random variable that corresponds to the sum of the cost of the edges on a path that corresponds to the cost of the RIC. Let \( Y_i = \mathbb{E}[Y|F_i] = \mathbb{E}[Y|\bar{X}(i)] \).

---

1 For example if \( A, B \in \mathbb{F} \) then \( A \cup B \in \mathbb{F} \) etc.
As we have noted before, \( Y_t \) is a \textit{Doob’s martingale} based on the random variables \( X_t \) and \( Y_0 \) denotes the expected running time of the RIC. We would like to bound the deviation \(|Y_n - Y_0|\) for any run of the algorithm with high probability, which in the context of this paper will be inverse polynomial, unless otherwise mentioned. Likewise the random deletion sequence also defines a \textit{Doob’s martingale} on the subsets which will be referred to as the \textit{backward-sequence} martingale (BSM henceforth).

The following martingale tail bound is the basis of many later results in this paper which is distinct from Azuma’s inequality and referred to as the \textit{Method of bounded variance}.

**Theorem 1 (Freedman\cite{9})** Let \( X_1, X_2 \ldots X_n \) be a sequence of random variables and let \( Y_k \), a function of \( X_1 \ldots X_k \) be a martingale sequence, i.e., \( \mathbb{E}[Y_k|X_1 \ldots X_{k-1}] = Y_{k-1} \) such that \( \max_{1 \leq k \leq n} \{|Y_k - Y_{k-1}|\} \leq M_n. \) Let

\[
W_k = \sum_{j=1}^{k} \mathbb{E}[(Y_j - Y_{j-1})^2|X_1 \ldots X_{j-1}] = \sum_{j=1}^{k} \text{Var}(Y_j|X_1 \ldots X_{j-1})
\]

where \( \text{Var} \) is the variance using \( \mathbb{E}[Y_j] = Y_{j-1} \). Then for all \( \lambda \) and \( W_n \leq \Delta^2 \), \( \Delta^2 > 0 \),

\[
\Pr[|Y_n - Y_0| \geq \lambda] \leq 2 \exp \left( -\frac{\lambda^2}{2(\Delta^2 + M_n \cdot \lambda/3)} \right)
\]

Note that the term \( \Delta^2 \) can be bounded by \( \sum_{j=1}^{n} \max_{X_1, X_2 \ldots X_j} \text{Var}(Y_j|X_1 \ldots X_{j-1}) \) i.e., the worst case bounds over all choices of length \( j \) prefix \( X^{(j)} \). If the inner term can be bounded by some function of \( j \), say, \( \omega(j) \), then we may obtain an upper bound on the probability of deviation for any sequence \( X^{(n)} \) as \( \sum_{j=1}^{n} \omega(j) \) which can be viewed as a function of \( n \).

Further, we will actually use a minor variation of this result (see \cite{9}). Suppose \( \Pr[M_n \geq g(n)] \leq \frac{1}{f(n)} \) for some non-decreasing functions \( g,f \). Then the overall bound becomes

\[
\Pr[|Y_n - Y_0| \geq \lambda] \leq 2 \exp \left( -\frac{\lambda^2}{2(\Delta^2 + g(n)\lambda/3)} \right) + \frac{1}{f(n)}
\]

Similarly it can also be extended to the case where \( W_n \leq \Delta^2 \) holds with probability \( 1 - \frac{1}{f(n)} \).

Henceforth, in the remaining paper, we will appeal to this version of Freedman’s inequality where the bounds on \( M_n \) and \( W_n \) hold with high probability. Often the term \( \frac{1}{f(n)} \) will be the dominant term, so the final tail bound will effectively be \( O(\frac{1}{f(n)}) \).

### 3 Application to Quicksort and related problems

Let us consider quicksort in the RIC framework and without loss of generality, let the input elements be \( \{1, 2 \ldots n\} \). The \( j \)-th pivot, \( 1 \leq j \leq n \), partitions the input into \( j + 1 \) ordered sets, by splitting some existing partition \( P \). Any element \( x \in P \) is charged the cost of comparison with the pivot - any element \( x' \not\in P \) is not charged. The running time of the algorithm can be bounded by the cumulative charges accrued by each element. In this analysis we will bound the charge of each element with high probability (w.h.p.\footnote{The acronym w.h.p. will be used to denote probability exceeding \( 1 - 1/n^\alpha \) for some appropriate constant \( \alpha > 0 \)) and the overall running time bound follows from multiplying by \( n \).

The associated weight with each edge is either 1 or 0 depending on whether the latest random choice is one of the boundary elements of the interval containing \( x \). We define a random variable

\[
I^x_j = \begin{cases} 
1 & \text{if interval containing } x \text{ changes in step } j \\
0 & \text{otherwise} 
\end{cases}
\]

From backward analysis, the probability of this is at most \( \frac{2}{j} \) for a uniformly chosen child node.\footnote{Using a simple trick by considering a circular ordering (see \cite{20}), this probability can be made exactly equal to \( \frac{2}{j} \).}
completeness, we have included a detailed description of backward analysis in the appendix. We will also omit the superscript \( x \) and just use \( I_j \) since we will obtain a worst case bound over all choices of \( x \). The reader may note that the bound on \( \mathbb{E}[I_j] \) is only a function of \( j \) and not \( \bar{X}^{(j)} \) over all random choices of any prefix of \( j \) elements.

It will also help to focus on the BSM for quicksort. A random deletion sequence creates a nested sequence of random subsets starting from the all the elements and ending in the empty sequence. An edge of this tree \( (K, K - \{y\}) \) is given a value \( 1 \) for a subset \( K \) and an element \( y \in K \) if in the (forward) quicksort algorithm, selecting \( y \) as a pivot and leading to \( K \) (all the pivots selected) forces a comparison between \( y \) and \( x \). Clearly two edges from any subset will be given a value \( 1 \), so that the expected cost for a random deletion is \( \frac{2}{n-j} \) in the \( j \)-th level, \( n \geq j \geq 0 \). Figure 8 gives a depiction of this random variable in the quicksort process.

Consider a path \( \mathcal{P} = v_0v_1 \ldots v_n \) from root to a leaf-node in this tree. The cost of this path is given by \( w(\mathcal{P}) = \sum_{i=1}^{n} w(v_i, v_{i+1}) \). A random path corresponds to one where \( v_{i+1} \) is a child of \( v_i \) chosen uniformly at random among the \( n - i \) children. The expected cost of such a random path is given by

\[
\mathbb{E}_{\text{random}}[w(\mathcal{P})] = \mathbb{E}[\sum_{i=1}^{n} w(V_i, V_{i+1})] \quad \text{where } V_{i+1} \text{ is a random child of node } V_i
\]

Let \( \mathbb{E}_{j}[Z] \) denote \( \mathbb{E}[Z | \bar{X}^{(j)}] \) for some random variable \( Z \). Note that \( \bar{X}^{(j)} \) represents a fixed path from root to level \( j \) of this tree corresponding to the deletion sequence \( X_1X_2 \ldots X_j \), say node \( V_j \). Then,

\[
\mathbb{E}_j[Y] = Y_j = \sum_{k=0}^{j-1} w(X_k, X_{k+1}) + \mathbb{E}\left[\sum_{k=j}^{n-1} w(V_k, V_{k+1}) = \sum_{k=0}^{j-1} w(X_k, X_{k+1}) + \sum_{k=j+1}^{n} \mathbb{E}[I_k]\right]
\]

It follows that \( Y_0 = 2H_n \) and we want to obtain a tail estimate for \( Y_n - Y_0 \).

We can compute

\[
Y_j - Y_{j-1} = w(X_{j-1}, X_j) + \left( \sum_{k=j+1}^{n} \mathbb{E}[I'_k] \right) - \left( \sum_{k=j}^{n} \mathbb{E}[I_k] \right)
\]

\[
= I_j - E[I_j] \quad \text{assuming } I_j, I'_j \text{'s have the same distribution}
\]

So \( \mathbb{E}_j[(Y_j - Y_{j-1})^2] = \mathbb{E}[(I_j - E[I_j])^2] \) This shows that the value of \( Y_j \) differs from \( Y_{j-1} \) because of the specific choice random variable \( X_j \). The above bound can be extended to a more general situations of RIC but where a single change can affect multiple “intervals” (more precisely, configurations). More specifically, for \( W() \) not bounded by a constant we have the following generalization as long as \( W_j \)'s have the same distribution across all nodes in level \( j \) for a random choice of the next node.

\[
\mathbb{E}X_j[(Y_j - Y_{j-1})^2] \leq \mathbb{E}X_j[W_j^2] - \mathbb{E}^2[W_j] \leq \mathbb{E}X_j[W_j^2]
\]

(2)

In the case of quicksort, we can complete the analysis as follows.

\[
\mathbb{E}[(I_j - E[I_j])^2] = \mathbb{E}[I_j^2] - \mathbb{E}^2[I_j]
\]

\[
\leq \mathbb{E}[I_j^2] - \frac{4}{(n-j)^2}
\]

\[
= \mathbb{E}[I_j^2] - \frac{4}{(n-j)^2}
\]

\[
\leq \frac{2}{n-j} \quad \text{since } I_j^2 \text{ is also a 0-1 indicator rv}
\]
Figure 2: Tree corresponding to the Backward Sequence Martingale corresponding to the comparisons for a fixed element $x$. The root corresponds to $Y_0$ which denotes the expected running time. Every edge has cost 0 or 1 depending on whether $x$ and $X_i$ belong to the same interval and a path in this tree reveals the indicator variables $I_j$.

So $\sum_{j=1}^{n} E_{j-1}[(Y_j - Y_{j-1})^2] \leq \sum_{j=1}^{n} \frac{2}{n-j} \leq 2H_n$ where $H_n \leq \log n$. Plugging in $\lambda = 2c \log n$ for some constant $c$ and using Freedman’s theorem, we obtain

$$\Pr[|Y_n - Y_0| \geq c \log n] \leq \exp \left( -\frac{4c^2 \log^2 n}{2(\log n + c \log n/3)} \right) \leq \frac{1}{n^c}.$$ 

Note that $M_n = Y_i - Y_{i-1} \leq 1$.

This shows that a single element incurs at most $O(\log n)$ cost with high probability and therefore quicksort runs in $O(n \log n)$ time with high probability.

**Remark** A straightforward application of the classic Azuma-Hoeffding bound [15]

$$\Pr[|Y_n - Y_0| \geq t] \leq \exp \left( -\frac{t^2}{\sum_{i=1}^{n} c_i^2} \right)$$

would not have been effective since the bound $c_i = M_n = 1$ makes the denominator too large for an $O(\log n)$ deviation bound. In [20], the author obtained a similar bound by using Chernoff bounds for binomial distribution by assuming independence between $I_j$s across different levels. This assumption is an oversimplification since choices of pivots across different levels could affect $I_j$’s.

Also note that, there exists a superior bound of $O(n^{-\Omega(\log \log n)})$ for Quicksort obtained in [13].

The above argument can be directly extended to obtain a concentration bound on the dart throwing game that has many applications (Mulmuley [17]). Consider throwing $n$ darts randomly in $n$ ordered locations, say numbered $\{1, 2 \ldots n\}$. Let $S(i)$ be a random variable that denotes the smallest numbered location among the first $i$ randomly thrown darts. Let $Z(i) = 1$ if $S(i) \neq S(i-1)$ and $Z(1) = 1$. So $Z(i)$ is the number of times $S(i)$ changes among the first $i$ darts thrown. We are interested in $\mathbb{E}[Z(n)]$ which can be shown to be $\sum_{i=1}^{n} \frac{1}{i} = H_n$, the $n$-th harmonic. This follows from backward analysis by observing that among a set of $i$ randomly chosen numbers, the probability of picking the smallest number as the last number is $\frac{1}{i}$. This is related to many visibility problems in geometry as well as the analysis of Trieps. Using the Freedman’s inequality, we can easily show the following from the previous argument and looking at the changes in the leftmost interval induced by the darts.
Corollary 2

\[ \Pr[|Z(n) - H_n| \geq 0.9 \log n] \leq \exp(-0.7 \log n) \leq \frac{1}{n^{0.7}} \]

This implies that \( \Pr[0.1 \log n \leq Z(n) \leq 1.9 \log n] \geq 1 - n^{-0.7} \). The above result has been stated in a slightly weaker manner so that we can claim a lower bound on \( Z(n) \) that will be invoked later to show the limitations of \( RIC \).

The analysis in this section also extends to problems like constructing trapezoidal maps that can be used for point location (Seidel [18]). Since a trapezoid can be defined by at most 4 segments, the expected work for point location is \( \sum_{i=1}^{n} \frac{1}{4} \leq 4 \log n \). Using a straightforward extension of the previous arguments, the following result can be obtained.

Lemma 3 Given a set of \( n \) non-intersecting line segments, a trapezoidal map can be constructed using \( RIC \) such that for any query point \( q \), the number times the trapezoid containing \( q \) changes can be bounded by \( O(\log n) \) with inverse polynomial probability.

This result will turn out to be very useful for some later results.

3.1 Comparison with an earlier bound

We briefly recall the framework of Mehlhorn, Sharir and Welzl [14] to model the general RIC algorithm. A rooted \((n, r)\) tree \( T \) is either a single node for \( r = 0 \) or (for \( r > 0 \)) the tree has \( n \) children which are recursively defined \((n-1, r-1)\) subtrees. Each of the \( n \) edges has an associated weight \( d_i \) corresponding to the \( i \)-th child and \( \max_{i=1}^{n} d_i \leq d(n) \) and \( \sum_i d_i \leq M(n) \). The expected cost of a path in this recursively defined tree is \( A = \sum_{i=1}^{n-1} \frac{M(n-i)}{(n-i)} \). One of the main results in the paper is the following tail bound (Theorem 1 in [14]).

\[ \Pr(X \geq B) \leq \left( \frac{e}{1 + B/A} \right)^{B/d(n)} \text{ for all } B \geq 0 \]

Although this bound looks somewhat simpler to use, this is not directly comparable to Freedman’s bound except for some special cases like Lemma 3 and quicksort where the concentration results are similar. It may be noted that the authors [14] analyze the backward execution of the algorithm for these results. This bound becomes weaker if \( d(n) \) is not a constant - for some of the later applications \( d(n) \) may be larger than \( A \) in the worst case. The authors improve the bound for the specific problem of building visibility maps of line segments by using the expected value of \( M(n) \). However, there is no generalization given for other problems.

4 Incremental Delaunay Triangulation

We will now consider somewhat more complex scenarios like construction of Delaunay Triangulation and three dimensional convex hull (see Guibas Knuth and Sharir [10]). Broadly speaking these algorithms have two distinct components -

(i) Updating the (partial) structure of the points inserted thus far.
(ii) Updating the point-location data structure of the uninserted points.

For concreteness, we will address the problem of Delaunay Triangulation. The analysis corresponding to updating the point location structure is similar to the analysis of quicksort given above. For the update of structural complexity, it was shown in [10] that the expected cumulative structural change can be bound by \( O(n) \), whereas for the latter, the expected work over all the \( n \) (random) insertions sequence
$O(n \log n)$. In several places, the authors in \[10\] pose the problem of tail estimates as an important open problem. We will do a combined analysis since we are interested in obtaining tail estimates on the work including all data structural updates.

In the remaining part of the paper, we will be alluding to the BSM framework and make use of Equation 2 for deriving the tail estimates. To avoid any confusion, we will use stage/level $k$ to refer to the forward algorithm when $k$ objects have been added and do all calculations in this order. Although the martingale has been defined for the backward execution, substituting $n - k$ by $k$, consistently will not affect anything except the order of the summations. This will also help us use the random sampling bounds without having to restate them in the flipped order.

We will make use of the following result of \[4, 11\].

**Theorem 4** At any stage $i$ of the RIC of Delaunay triangulation, the $i$ randomly chosen points is a uniform random subset of the $n$ points. So the number of unsampled points within each triangle is bounded by $O(\frac{n}{i} \log n)$ with probability $1 - 1/n^c$ for any constant $c > 1$. Moreover, all the triangles that emerges in the course of edge flips also satisfy the above bounds.

**Remark:** All the triangle that show up in the course of edge flips belong to $\Pi^0(R)$. Although some of them are not delaunay triangles and therefore, only temporary, they can contribute to the running time, depending on if one maintains the intermediate partitions.

To apply Freedman’s bound, we will first bound the variance. Unlike the analysis of quicksort, we will consider the work done for all the $n$ points (actually $n - i$ uninserted points in stage $i$) together. Each edge flip involves four triangles - two old and two new and redistributes the points in the two new triangles. Since each triangle contains $O(\frac{n}{i} \log n)$ points w.h.p, each edge flip can be be done in $O(\frac{n}{i} \log n)$ w.h.p. Since the maximum degree of a Delaunay graph of $i$ points is $i$, the total number of edge flips in the $i$-th stage is bounded by $i$. Therefore we can claim

**Lemma 5** The work in stage $i$ of the algorithm, $i \leq n$ can be bounded by $O(n \log n)$ w.h.p.

Let $\Pi_s(R)$ denote the configurations in $\Pi^0(R \cup s)$ adjacent to $s$ (or defined by $s$). The following claims can be easily derived from some general random-sampling lemmas in \[4\]

**Lemma 6**

$$
\mathbb{E}\left[ \sum_{\sigma \in \Pi_s(R)} \ell(\sigma) \right] = O\left( \frac{n}{r} \right) \mathbb{E}[|\Pi_s(R)|]
$$

$$
\mathbb{E}\left[ \sum_{\sigma \in \Pi_s(R)} \ell^2(\sigma) \right] = O\left( \frac{n^2}{r^2} \right) \mathbb{E}[|\Pi_s(R)|]
$$

**Bounding Variance**

We will need the following result

**Lemma 7** For real numbers $x_i \ 1 \leq i \leq r$

$$
\left( \sum_{i=1}^{r} x_i \right)^2 \leq r \left( \sum_{i=1}^{r} x_i^2 \right)
$$

**Proof:** Using the convexity of the square function, from Jensens inequality it follows that

$$
\frac{\sum_{i=1}^{r} x_i^2}{r} \geq \left( \frac{\sum_{i=1}^{r} x_i}{r} \right)^2
$$
Multiplying both sides by $r^2$ yields the required result. $\square$

The work done when a degree $j$ vertex $v$ is picked is proportional to the number of points in the triangles adjoining the vertex. If $l(\sigma)$ is the number of points in a triangle $\sigma$, then the work is proportional to $T_k = \sum_{\sigma \in \Delta(v)} l(\sigma)$ where $\Delta(v)$ denotes triangles adjacent to $v$. Squaring $T_k$ and taking expectation

$$E[T_k^2] = \frac{1}{k} \sum_{v \in R^k} E\left[ \sum_{\sigma \in \Delta(v)} l(\sigma)^2 \right]$$

$$\leq \frac{1}{k} \sum_{v \in R^k} E\left[ \sum_{\sigma \in \Delta(v)} |\Delta(v)||l(\sigma)|^2 \right] \text{ from previous lemma}$$

$$= \frac{1}{k} \sum_{v \in R^k} \frac{n^2}{k^2} |\Delta(v)|^2$$

$$= O\left(\frac{n^2}{k}\right) \text{ as } \sum_{v} |\Delta(v)|^2 = O(k^2)$$

This yields

$$W_n \leq \sum_{k=1}^{k=n} E[T_k^2] \leq \sum_{k=1}^{k=n} O\left(\frac{n^2}{k}\right) = O(n^2 \log n)$$

Plugging the bound of $M_n = O(n \log n)$ from Lemma 5 in Freedman’s theorem, we obtain the following bound.

**Lemma 8** Let $T(n)$ denote the running time of ric based construction of Delaunay Triangulation and let $\lambda = cn \log n$ for a suitable constant $c$. Then

$$\Pr[T(n) \geq \alpha(n) \lambda] \leq \exp\left( -\frac{(\alpha(n)cn \log n)^2}{2(n^2 \log n + \alpha(n)c \cdot n^2 \log n/3)} \right) \leq \exp(-\alpha(n))$$

**Remark** The above Lemma gives high probability bound for $T(n)$ exceeding $\Omega(n \log^2 n)$ for $\alpha = \Omega(\log n)$. However, this bound is superior to the straightforward Markov’s bound applied on the expected work as well as preferable to restarting the original algorithm using independent random bits each time. There are also inputs (see Figure ??) for which the inverse polynomial high probability bounds do not seem possible without changing the algorithm.

This analysis can be extended to the three-dimensional convex hull algorithm presented in Mulmuley [16]. For fixed dimensional linear programming Seidel [19] proved a similar property and this can be extended to RIC algorithms like closest pair [12]. However, these bounds are not very interesting and the reason could be viewed as follows. In the steps that the RIC algorithm for LP and closest-pair re-builds the data structure, there is a high cost that gets subsumed in the expected bounds. But in Freedman’s inequality, it is easily seen that when $M_n = \Omega(\lambda(n))$, we are unable to obtain a inverse polynomial concentration bounds around $\lambda(n)$.

5 More generalized RIC: segment intersections

We now consider a more general scenario in RIC (Randomized Incremental Construction). Using a conflict graph update model of RIC, we obtain the following expression for expected work.
Expected work (#edges created in the conflict graph) =

\[ \sum_{\sigma \in \Pi^0(R \cup s)} l(\sigma) \cdot \Pr\{\sigma \in \Pi^0(R \cup s) - \Pi^0(R)\} \]

From \textit{backward analysis} this probability is the same as deleting a random element from \( R \cup s \) which is \( d(\sigma) \). By substituting this we obtain

\[ \sum_{\sigma \in \Pi^0(R \cup s)} l(\sigma) \cdot \frac{d(\sigma)}{r + 1} = \frac{d(\sigma)}{r + 1} \sum_{\sigma \in \Pi^0(R \cup s)} l(\sigma) \]

\[ = O\left(\frac{d(\sigma)}{r} \cdot \frac{n}{E[\Pi^0(R \cup s)]}\right) \]

Therefore the expected work over the sequence of random insertions is

\[ \sum_{r=1}^{n} O\left(\frac{d(\sigma)}{r} \cdot \frac{n}{E[\Pi^0(R \cup s)]}\right). \]

For the case of line segment intersections, it can be shown that \( E[\Pi^0(R \cup s)] = O(r + \frac{m \cdot n^2}{n^2}) \) from which it follows that the expected work is

\[ \sum_{r=1}^{n} O\left(\frac{d(\sigma)}{r} \cdot \frac{n}{E[\Pi^0(R \cup s)]}\right) = \sum_{r=1}^{n} \left(\frac{dn}{r} + \frac{dm}{n}\right) = O(n \log n + m). \]

Here \( d(\sigma) \leq 6 \) which the maximum number of segments that define a \( \sigma \) (trapezoid in this case).

Tail bounds for this problem has been elusive despite significant effort (see [14]). We will show that our previous techniques can be extended to obtain tail estimates on the work done.

Consider an arrangement of \( n \) segments with \( m \) intersections \( (0 \leq m \leq \binom{n}{2}) \). In the trapezoidal map \( T \) of the \( n \) segments (also known as a vertical visibility diagram), let us denote the set of trapezoids adjacent to segment \( s_i \) by \( T_i \). Any trapezoid \( \sigma \in T \) is defined by at most six segments. Since it is a planar map, and there are at most \( 2n + 2m \) vertices, it follows that \( \sum_{i} |T_i| = O(n + m) \). We would like to obtain a bound on \( \sum_{i} |T_i|^2 \). Let us denote by \( n_i \) and \( m_i \) respectively, the number of segments end-points and intersection points visible to segment \( s_i \). It follows that \( |T_i| = O(n_i + m_i) \) and

\[ \sum_{i} |T_i|^2 = O \left( \sum_{i} (n_i + m_i)^2 \right) = O \left( \sum_{i} n_i^2 + \sum_{i} n_i \cdot m_i + \sum_{i} m_i^2 \right) \]

where \( m_i \leq O(n) \) from the zone theorem bound. Moreover \( \sum_{i} n_i = O(n) \) and \( \sum_{i} m_i = O(m) \) as each point is visible from the closest segments above and below.

The first expression can be bounded by \( \sum_{i} (n_i + m_i)^2 = O(n^2) \) and the second expression by \( 2 \sum_{i} n_i \cdot m_i = O(n \cdot m) \) (Cauchy-Schwartz inequality). The third expression is less than \( \frac{m}{n} \cdot n^2 = mn \).

So, the overall expression can be bounded by \( O(m \cdot n + n^2) \).

For a uniformly chosen prefix \( S^k \) of size \( k \), the expected number of intersections in the sample is \( \frac{m \cdot k^2}{n^2} \), so the variance can be bounded by

\[ E[T^2_k | S^k] = \frac{1}{k} E \left[ \sum_{\sigma \in T_k} (\sum_{\sigma \in T_1} l(\sigma))^2 \right] \]

To simplify calculations, we recall (Theorem 4) that \( l(\sigma) \leq O\left(\frac{n \log n}{k}\right) \) with high probability. So, plugging this into the previous expression, and using the previous bound on \( \sum_{i} |T_i|^2 \), we obtain (w.h.p.)

\[ E[T^2_k | S^k] \leq \frac{1}{k} \cdot O\left(\frac{n^2 \log^2 n}{k^2} \cdot \frac{1}{k^2} \right) \cdot E[\frac{k^2 + k \cdot m_k}{k^2}] \]

11
where \( m \) is the number of intersections in \( S^k \). Taking expectation over all choices of \( S^k \), we obtain the unconditional expectation as
\[
\mathbb{E}[T_k^2] \leq \frac{n^2 \log^2 n}{k^3} \cdot \mathbb{E}[k^2 + km_k] \leq \frac{n^2 \log^2 n}{k^3} \cdot (k^2 + \frac{mk^3}{n^3}) \leq \frac{n^2 \log^2 n}{k} + m \log^2 n
\]

This uses the bound \( \mathbb{E}[m_k] = O(k + \frac{mk^2}{n^2}) \). This bound is relevant for the maintenance of conflict graphs.

In contrast, for an algorithm like Mulmuley [17], where only the trapezoids are maintained, the work done can be at most \( O(mn \log n) \) for some constant \( c > 0 \). Therefore, the expected running time exceeds the expected running time by a factor of \( \Omega(\log n) \).

For the specific case of \( T^2 \) for the algorithms of [17] and [4] that do not maintain conflict-graphs but only involves segment end-points. As observed before, the quantity \( W \) can be bounded by
\[
\sum_{k=1}^{n} O(k + \frac{mk^2}{n^2}) = O(n^2 + mn).
\]

To obtain high probability bounds using Freedman’s inequality, we want to bound this expression by \( \frac{\lambda^2}{\log n} \) where \( \lambda = c(n \log n + m) \). \( M_n \) can be bounded by \( O(n \log n \cdot \alpha(n)) \) and \( M_n \cdot \lambda \) by \( mn \alpha(n) \) with high probability. This follows from a bound of \( O(t \alpha(t)) \) on the zone of a segment that intersects \( t \) segments in an arrangement of \( n \) segments ([17]).

So from Freedman’s inequality we obtain a tail bound of \( \exp\left(-\frac{m^2}{mn \log n}\right) \) for \( m \geq n \log^2 n \).

**Theorem 9** Let \( T(n) \) represent the work done in the conflict-graph based segment intersection algorithm, then there exists constant \( \beta \), such that for \( m \geq \beta n \log^2 n \),
\[
\Pr[T(n) \geq m] \leq \exp\left(-\frac{m}{n \log^2 n}\right)
\]

To the best of our knowledge, no prior concentration bound was known for the conflict-graph based approach for segment intersection given by Clarkson and Shor [4]. The paper by [14] noted that their methods could not be extended to this algorithm.

For the specific case of \( m = 0 \), the bound can be improved by observing that the zone of a segment can be at most \( O(n) \) (instead of \( n \alpha(n) \)) as there are no intersections. Setting \( m = 0 \) in the previous bound for \( W_n \), we obtain the following:

**Corollary 10** For constructing the trapezoidal map of \( n \) non-intersecting line segments using RIC, the work done \( T(n) \) satisfies
\[
\Pr[T(n) \geq c \beta n \log^2 n] \leq n^{-\beta^2}
\]
for some constant \( c > 1 \).

This shows that we can obtain inverse polynomial concentration bounds around a running time that exceeds the expected running time by a factor of \( \Omega(\log n) \).

We now return to the algorithms of [17] and [4] that do not maintain conflict-graphs but only involves segment end-points. As observed before, the quantity \( W_n \) can be bounded by
\[
\sum_{k=1}^{n} O(k + \frac{mk^2}{n^2}) = O(n^2 + mn).
\]

We summarize as follows.

\[ ^a \text{there is some additional cost for point location that can be bounded using Lemma 3} \]
Lemma 11 In the segment intersection algorithms of \cite{T,F} that do not maintain conflict-graphs explicitly, the probability that the work exceeds $c(m + n \log n)$ can be bounded by

$$\exp \left( - \frac{\Omega(m^2 + mn \log n + n^2 \log^2 n)}{O(mn \alpha(n) + n^2 \alpha(n) \log n)} \right) \leq \exp \left( - \frac{\log n}{\alpha(n)} \right)$$

since $M = O(n \alpha(n))$.

For $m \geq n \log n$, the bound improves to $\exp \left( - \frac{m}{n \alpha(n)} \right)$.

Remark This bound is better than the results in \cite{M} where the authors show that for some constant $\delta > 0$,

$$\Pr[T(n) \geq Cm] \leq \exp \left( - \frac{\delta m}{n \log n} \right) \text{ for } m \geq n \log n \log \log \log n$$

6 Can we improve the tail bounds

Figure 3 shows an input of $n$ horizontal segments divided into two sets $\mathcal{U}, \mathcal{B}$ each of which has $n/2$ segments. In the top pile $\mathcal{U}$ of the $n/2$ horizontal segments, let us consider the lowest $\sqrt{n} \cdot c(n)$ segments for some function $c(n)$ that will determined in the analysis.

We will consider the RIC after the first $10\sqrt{n}$ insertions. The number of segments in $\mathcal{B}$ and $\mathcal{U}$ respectively is at least $\sqrt{n}$ with high probability. Notice that, if the lowest sampled segment is $s \in \mathcal{U}$ then all the segments below $s$ are visible to all the end-points in the lower pile $\mathcal{B}$ of the segments. Namely, if there are $m$ unsampled segments below $s$, then the size of the conflict graph is at least $\sqrt{n} \cdot m$. Consider the set of $\sqrt{n}c(n)$ segments and denote this by $T$.

Among the set of $10\sqrt{n}$ segments, what is the probability that none of them was sampled from the set $T$. This can be easily seen as

$$(1 - \frac{c(n)}{\sqrt{n}})^{10\sqrt{n}} = \Omega(4^{-10c(n)})$$

Every time $s$ changes, new edges are created in the conflict graph by the sampled edges in $\mathcal{B}$. Following the initial $10\sqrt{n}$ segments, let us consider the second phase where segments from $T$ may be
sampled. Within $T$, let us denote by $T'$ the lowest $c(n)\sqrt{n}$ segments for some constant $c < 1$ and let $T''$ denote the remaining segments. What is the probability that among the first $\log n$ segments sampled from $T$, none are from $T'$? This can be calculated as $(1 - c)^{\log n} = \Omega(4^{-c\log n})$.

It follows from backward analysis that the lowest sampled segment in $T''$ changes about $\Omega(\log \log n)$ times (Corollary 2). So, at least $\Omega(c(n) \sqrt{n} \log n)$ edges are created with probability $\Omega(\exp(-20c(n)))$. For example, for $c(n) = \log n/40$, then the probability is at least $\frac{1}{\sqrt{n}}$ that the total number of edges created in the conflict graph is $\Omega(n \log n \log \log n)$. More specifically, this holds with probability $\frac{1}{\log n} \cdot \frac{1}{n^\epsilon} \cdot \frac{1}{\sqrt{n}} \cdot (1 - \frac{1}{n})$ for some constants $0 < \alpha, \beta < 1$ which is $\Omega(\frac{1}{\sqrt{n}})$. The multiplicative factors (very close to 1) help us uncondition the probability that

(i) Adequate samples - about $\sqrt{n}$ are chosen from sets $U, B$ and
(ii) There are $\theta(\log \log n)$ changes in $s$.

The above argument can be easily generalized as follows

**Theorem 12** There exists inputs for which the conflict-graph based RIC algorithm for constructing vertical visibility maps (segment intersections with no intersections) encounters $\Omega(c(n) \log n \log \log n)$ structural changes with probability $\Omega(e^{-20c(n)})$ for $c(n) = o(\sqrt{n})$.

In particular, by choosing $c(n) = \log n/40$, the conflict graph based RIC algorithm may encounter $\Omega(n \log \log n)$ changes with probability $\Omega(\frac{1}{\sqrt{n}})$ that rules out inverse polynomial bounds for a total work of $O(n \log n)$. Note that $\frac{1}{\sqrt{n}}$ can be easily increased to $\frac{1}{n^\epsilon}$ for any $\epsilon > 0$.

By comparing this result with Corollary 10, we have nearly tight bounds for tail estimates, modulo some constant factor in the work done.

**References**

[1] P. Chew. The simplest voronoi diagram algorithm takes linear expected time. In *Manuscript*, 1988.

[2] Kenneth L. Clarkson. Applications of random sampling in computational geometry, II. In *Symposium on Computational Geometry*, pages 1–11. ACM, 1988.

[3] Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental constructions. In *STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, France, February 13-15, 1992, Proceedings*, pages 463–474, 1992.

[4] Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[5] J.L. Doob. Regularity properties of certain families of chance variables. *Transactions of the American Mathematical Society*, 47 (3):455–486, 1940.

[6] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[7] W. Feller. *An Introduction to Probability Theory, Vol. 1*. Wiley, New York, NY, third edition, 1968.

[8] W. Feller. *An Introduction to Probability Theory and Its Applications, Vol. 2*. Wiley, New York, NY, second edition, 1971.

[9] David Freedman. On tail probabilities on martingales. In *The Annals of Probability*, volume 3(1), pages 100–118, 1975.
7 Appendix

We provide a brief description of the notations and definitions that capture the framework of RIC and its analysis in very general setting.

Given a set $S$ of $n$ elements (like points, segments, lines etc.), a configuration $\sigma$ is defined by at most $d$ objects where $d = O(1)$. The set of objects is denoted by $d(\sigma)$ and the number of configurations is bounded by $n^d$ if there are no more than $O(1)$ configurations associated each subset of $d$ elements (there can be more than one configuration associated with the same $d(\sigma)$ elements).

Let $\ell(\sigma) = S \cap \sigma - d(\sigma)$ be the elements that intersect with $\sigma$. With a slight overloading of notation we will also use $\ell(\sigma)$ to denote the set of the intersecting elements with $\sigma$ also. Let $\Pi^i(S)$ denote the set of configurations $\sigma$ with $\ell(\sigma) = i$. We use $\Pi(S) = \bigcup_i \Pi^i(S)$ to denote all configurations. For any subset $R \subset S$, we use $\Pi(R)$ to denote the configurations defined by elements of $R$ and the conflict list of any configuration $d(\sigma) \subset R$ as $\sigma \cap S$, i.e., all the elements and not just the elements in $R$.

A conflict graph represents the relation between the configurations in $\Pi^0(R)$ and the corresponding conflict list, which is a bipartite graph with configurations in $\Pi^0(R)$ on one side and the uninserted elements on the other side. Randomized Incremental construction can be thought of as maintaining
and update of the conflict graph starting with $R = \emptyset$ and successively adding a random (uninserted) element $e \in S - R$ into $R$. This introduces $\sigma \in \Pi^0(R \cup e) - \Pi^0(R)$ requiring appropriate changes in the conflict graph. To illustrate this framework on quicksort, we define the configurations as intervals defined by a pair of elements $[x_i, x_j]$ where $x_i < x_j$. Initially there is the hypothetical configuration $(-\infty, +\infty)$. As we introduce more pivots, we maintain the ordered set of intervals induced by the elements chosen as pivots. As we introduce a pivot, some interval is split. Eventually we have the sorted set defined by consecutive intervals. When an interval $[x_i, x_j]$ splits because of a pivot element $y$ such that $x_i < y < x_j$, the elements in $\ell([x_i, x_j]) \cap S$ is reassigned to $\ell([x_i, y])$ and $\ell([y, x_j])$ appropriately. The number of comparisons required is roughly $|\ell([x_i, x_j]) \cap S|$ (the cardinality).

The analysis of quicksort in this framework can be done using the technique of \textit{backward analysis} which is very elegant. Let us assign an indicator random variable $X_k$ associated with an element $x$, such that

$$
X_k = \begin{cases} 
1 & \text{if } x \text{ is compared for the } k\text{-th pivot} \\
0 & \text{otherwise}
\end{cases}
$$

The number of comparisons involving $x$ is given by $\sum_{k=1}^{n} X_k$. Therefore

$$
\mathbb{E}[\sum_{k=1}^{n} X_k] = \sum_{k=1}^{n} \mathbb{E}[X_k] = \sum_{k=1}^{n} p_k(x)
$$

where $p_k(x)$ is the probability that element $x$ is involved in the partitioning of the $k$-th pivot insertion.

To compute the probability, we observe that $X_k = 1$ iff the $k$-th pivot $y$ is one of the two elements that bound the interval containing $x$ after $k$ pivots are chosen randomly. For a fixed choice of $k$ initial pivots, the probability that $y$ is one of the two bounding elements is at most $\frac{2(k-1)!}{k!} = \frac{2}{k}$. The numerator represents the number of permutations with one of the bounding elements being the last pivot. Although this is the probability conditioned on the choice of the first $k$ pivots, clearly unconditioning would also give us the same probability. Therefore the expected number of comparisons involving $x$ is $\sum_{k=1}^{n} \frac{n}{k} = O(\log n)$. Further the total expected number of comparisons is $O(n \log n)$ by summing over all elements.