Some Randomized Algorithms

Mohamed Syaheer Altaf

September 23, 2025

Randomized algorithms are algorithms that employ a degree of *randomness* in their logic or procedure. Unlike deterministic algorithms that produce the same outputs for the same inputs and terminate at a deterministic time, randomized algorithms may sacrifice accuracy (i.e., provide correct outputs with high probability) in exchange for efficiency (the runtime is low and bounded), or they can always provide correct outputs but with randomized runtime and average efficiency. Therefore, there are two types of randomized algorithms:

- *Las Vegas Algorithms*: these algorithms always produce the correct output, but their running time is a random variable—the expected running time is often better than deterministic alternatives in practice.
- *Monte Carlo Algorithms*: these algorithms have a chance of producing incorrect outputs, but the runtime is guaranteed and oftentimes efficient—hence the probability of error can be reduced significantly by repeating the algorithm multiple times.

Now, we will look at some randomized algorithms.

1 Randomized Quicksort

Algorithm 1 Randomized Quicksort

Quicksort(S)

Require: A list $S = \{s_1, \ldots, s_n\}$ of n distinct elements from a totally ordered universe.

1: **if** $|S| \leq 1$ **then**
2: **return** S
3: Choose a pivot $s \in S$ uniformly at random.
4: Partition the remaining elements into
   (a) $S_1 = \{x \in S \setminus \{s\} : x < s\}$,
   (b) $S_2 = \{x \in S \setminus \{s\} : x > s\}$.
5: **call** Quicksort(S_1); **call** Quicksort(S_2).
6: **return** (S_1, s, S_2).

Analysis. We will now analyze the correctness and the runtime of the algorithm:

Correctness. The crucial invariant is that, after each partition around a chosen pivot s_i, the list is *locally sorted* with respect to s_i: every element in

$$S_1 = \{x \in S \setminus \{s_i\} \mid x < s_i\}$$

is less than s_i, and every element in

$$S_2 = \{x \in S \setminus \{s_i\} \mid x > s_i\}$$

is greater than s_i. Thus the pivot immediately occupies its final position in the overall order. It remains only to sort S_1 and S_2 relative to their own pivots. Applying QUICKSORT recursively to each sublist enforces the same invariant on progressively smaller subsets, until the recursion reaches a single element or the empty set, both of which are already sorted. Because every element is eventually placed in its correct position relative to every other element, the entire list is sorted when the algorithm terminates.

Runtime. We measure runtime by the number of *comparisons* performed before the algorithm terminates as the input size grows. The **worst case** occurs when every pair of elements is compared, yielding $\binom{n}{2} = \frac{n(n-1)}{2}$; hence Algorithm **??** has worst-case complexity $O(n^2)$. This, however, is rare, so we look instead at the **average** number of comparisons as a practical performance metric.

We first observe that the expected number of comparisons is the same for all permutations of the input S; this is because the pivot at each step is chosen uniformly at random. Hence, without loss of generality, we may assume that S is already sorted—i.e., $\mathrm{ord}(s_i) < \mathrm{ord}(s_j)$ for every $i < j$.

For $i < j$, let $\mathbb{I}_{ij}$ be the indicator random variable that equals 1 if s_i and s_j are ever compared during the algorithm. Then

$$E[\#\text{ comparisons}] \;=\; \sum_{i<j} E[\mathbb{I}_{ij}] \;=\; \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr(\mathbb{I}_{ij} = 1).$$

Elements s_i and s_j are compared *iff* one of them is chosen as the pivot for the sublist $\tilde{S} = \{s_i, s_{i+1}, \ldots, s_j\}$. This sublist can occur only if every earlier pivot s satisfies $s < s_i$ or $s > s_j$; if instead a pivot with $s_i < s < s_j$ were chosen earlier, s_i and s_j would be separated into different sublists and could never be compared. Therefore

$$\Pr(\mathbb{I}_{ij} = 1) \;=\; \frac{1}{|\tilde{S}|} + \frac{1}{|\tilde{S}|} \;=\; \frac{2}{j-i+1}.$$

Summing over all possible sizes of $|\tilde{S}|$ gives

$$\begin{aligned}
E[\#\text{ comparisons}] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= 2\Big[\Big(\tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{n}\Big) + \Big(\tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{n-1}\Big) + \cdots + \tfrac{1}{2}\Big] \\
&= 2\Big[n\Big(\tfrac{1}{2}\Big) + (n-1)\Big(\tfrac{1}{3}\Big) + \cdots + 1\Big(\tfrac{1}{n}\Big)\Big] \\
&< 2n\Big(1 + \tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{n}\Big) \;=\; 2nH(n).
\end{aligned}$$

Hence the **expected** runtime is $O(n \log n)$, making randomized quicksort highly practical. Because the algorithm always produces a correct output (by the correctness proof) while its runtime is random, it is a *Las Vegas* randomized algorithm. □

2 Randomized Median

Algorithm 2 Randomized Median

Median(S)

Require: An unsorted list S of n elements drawn from a totally ordered universe.

1: Obtain a (multi)set R of $\lceil n^{3/4} \rceil$ elements from S, chosen independently and uniformly at random with replacement.
2: Sort R.
3: Let d be the $\left\lfloor \frac{n^{3/4}}{2} - \sqrt{n} \right\rfloor$-th element in sorted R.
4: Let u be the $\left\lfloor \frac{n^{3/4}}{2} + \sqrt{n} \right\rfloor$-th element in sorted R.
5: Compute the following by scanning S:
   (a) $\mathcal{C} = \{s \in S \mid d \leq s \leq u\}$,
   (b) $\mathcal{L}_d = \{s \in S \mid s < d\}$,
   (c) $\mathcal{L}_u = \{s \in S \mid s > u\}$.
6: **if** $|\mathcal{L}_d| > \frac{n}{2}$ or $|\mathcal{L}_u| > \frac{n}{2}$ **then**
7: **return** FAIL
8: **if** $|\mathcal{C}| \leq 4n^{3/4}$ **then**
9: Sort $\mathcal{C}$.
10: **else**
11: **return** FAIL
12: **return** $(\lfloor \frac{n}{2} \rfloor - |\mathcal{L}_d| + 1)$-st element in sorted $\mathcal{C}$.

Analysis. We will now analyze the correctness, the probability of returning FAIL, and the runtime of the algorithm.

Correctness. We would like to show that if the algorithm does not return FAIL, then it returns the *correct* median. Also, we assume that n is odd for convenience. First, note that the median m_S must lie in $\mathcal{C}$ to begin with if the algorithm is to return it. If $m_S \notin \mathcal{C}$, assume $m_S \in \mathcal{L}_d$. Suppose we construct the set $\mathcal{D}_S = \{s \in S \mid s \leq m_S\}$; then $|\mathcal{D}_S| = \lfloor \frac{n}{2} \rfloor + 1$ because m_S is the median. For the median to lie in $\mathcal{L}_d$, we must have $d > m_S$, which implies $|\mathcal{L}_d| > |\mathcal{D}_S|$—a condition that causes the algorithm to return FAIL in Step 6 of the algorithm. A similar argument applies if $m_S \in \mathcal{L}_u$. Now if S were sorted, the portion of the sorted $\mathcal{C}$ containing m_S would match exactly; therefore, to find the offset, we first determine the position of m_S in sorted S, which is $\lfloor \frac{n}{2} \rfloor + 1$, and because the portion of $\mathcal{C}$ starts at position $|\mathcal{L}_d| + 1$ in sorted S, the median must lie at the $(\lfloor \frac{n}{2} \rfloor - |\mathcal{L}_d| + 1)$-st element of sorted $\mathcal{C}$ (from $\left(\lfloor \frac{n}{2} \rfloor + 1\right) - \left(|\mathcal{L}_d| + 1\right) + 1$).

Probability of FAIL. As stated, the algorithm fails under the three conditions specified as the following events:

1. $\mathcal{E}_1 : |\mathcal{L}_d| > \frac{n}{2}$,
2. $\mathcal{E}_2 : |\mathcal{L}_u| > \frac{n}{2}$,
3. $\mathcal{E}_3 : |\mathcal{C}| > 4n^{3/4}$.

Due to symmetry, we know that $\Pr(\mathcal{E}_1) = \Pr(\mathcal{E}_2)$—thus proving one is sufficient. We know that if $m_S \in \mathcal{L}_d$, then $d > m_S$ and d is the $\left\lfloor \frac{n^{3/4}}{2} - \sqrt{n} \right\rfloor$-th element of sorted R. This implies that $|\{r \in R \mid r \leq m_S\}| < \frac{1}{2}n^{3/4} - \sqrt{n}$, which is the position of d in sorted

R. Let the cardinality of this set be the random variable X. We therefore compute the probability of this event (i.e., when $X < \frac{1}{2}n^{3/4} - \sqrt{n}$), which is exactly $\Pr(\mathcal{E}_1)$. We define an indicator random variable $\mathbb{I}_i$ that takes the value 1 if the ith sample is less than or equal to m_S. Because the sampling is done with replacement, each $\mathbb{I}_i$ is independent. Since there are $\frac{n-1}{2}+1$ such elements in S (because $\lfloor \frac{n}{2} \rfloor = \frac{n-1}{2}$ for odd n), it follows that

$$\Pr(\mathbb{I}_i = 1) = \frac{(n-1)/2 + 1}{n} = \frac{1}{2} + \frac{1}{2n}.$$

This means that $X \sim \mathsf{binomial}(n^{3/4}, \frac{1}{2} + \frac{1}{2n})$ (because $X = \sum_i \mathbb{I}_i$). The variance of this binomial random variable is then computed:

$$\mathsf{Var}[X] = n^{3/4}\Big(\tfrac{1}{2} + \tfrac{1}{2n}\Big)\Big(\tfrac{1}{2} - \tfrac{1}{2n}\Big) < \frac{1}{4}n^{3/4}.$$

Meanwhile, $E[X] = \frac{1}{2}n^{3/4} + O(n^{-1/4})$. Then, using Chebyshev's inequality, we obtain

$$\begin{aligned}
\Pr(\mathcal{E}_1) &= \Pr\Big(X < \tfrac{1}{2}n^{3/4} - \sqrt{n}\Big) \\
&= \Pr\Big(-\Big[X - \Big(\tfrac{1}{2}n^{3/4} + O(n^{-1/4})\Big)\Big] > \sqrt{n} + O(n^{-1/4})\Big) \\
&\leq \Pr\Big(|X - E[X]| > \sqrt{n} + O(n^{-1/4})\Big) \\
&\leq \frac{\mathsf{Var}[X]}{\Big(\sqrt{n} + O(n^{-1/4})\Big)^2} \\
&< \frac{\mathsf{Var}[X]}{n} \\
&< \tfrac{1}{4}n^{-1/4}.
\end{aligned}$$

Now it remains to derive the bound for $\Pr(\mathcal{E}_3)$. Unlike events $\mathcal{E}_1$ and $\mathcal{E}_2$, which are necessary conditions to return the *correct* median when they do not occur, $\mathcal{E}_3$ was set in place so that the algorithm terminates in *linear time*—i.e., set $\mathcal{C}$ is not too big for sorting in Step 9 of Algorithm **??**. When $|\mathcal{C}| > 4n^{3/4}$, **at least** one of the following two events must have occurred:

1. $\mathcal{E}_3'$: at least $2n^{3/4}$ elements of $\mathcal{C} < m_S$,
2. $\mathcal{E}_3''$: at least $2n^{3/4}$ elements of $\mathcal{C} > m_S$.

To prove this fact (i.e., $|\mathcal{C}| > 4n^{3/4} \Rightarrow \mathcal{E}_3' \cup \mathcal{E}_3''$), we employ the *contrapositive*. That is, when $\overline{\mathcal{E}_3'} \cap \overline{\mathcal{E}_3''}$, then

$$\begin{aligned}
|\mathcal{C}| &= \Big|\{\, c \in \mathcal{C} \mid c < m_S \,\}\Big| + \Big|\{\, c \in \mathcal{C} \mid c > m_S \,\}\Big| + \Big|\{\, c \in \mathcal{C} \mid c = m_S \,\}\Big| \\
&< 2n^{3/4} + 2n^{3/4} + 1 \\
&\leq 4n^{3/4}.
\end{aligned}$$

Again, due to symmetry, $\Pr(\mathcal{E}_3') = \Pr(\mathcal{E}_3'')$—so proving one of them is sufficient (as an exercise, we will prove the latter). To clarify, when $|\{c \in \mathcal{C} \mid c > m_S\}| \geq 2n^{3/4}$, this means that u was at least the $\Big(\frac{1}{2}n + 2n^{3/4}\Big)$-th element in sorted S and hence the set R contains **at least** $n^{3/4} - \Big(\frac{1}{2}n^{3/4} + \sqrt{n}\Big) + 1 = \frac{1}{2}n^{3/4} - \sqrt{n} + 1$ samples among the $\frac{1}{2}n - 2n^{3/4} + 1$

largest elements in S (this number comes from sorted S with $n - \left(\frac{1}{2}n + 2n^{3/4}\right) + 1$). For convenience, we drop the constant $+1$ terms from our computation. Let X be the number of samples among the $\frac{1}{2}n - 2n^{3/4}$ largest elements in S; then X is a binomial random variable with $n^{3/4}$ total samples, each sampling that portion of S independently with probability

$$p = \frac{\frac{1}{2}n - 2n^{3/4}}{n} = \frac{1}{2} - 2n^{-1/4}.$$

Again, by computing $E[X]$ and $\mathsf{Var}[X]$ and then applying Chebyshev's inequality, we obtain

$$\Pr(\mathcal{E}_3'') \leq \frac{1}{4}n^{-1/4}.$$

A similar argument and bound can be derived for $\Pr(\mathcal{E}_3')$. By the *union bound*, we get $\Pr(\mathcal{E}_3) = \Pr(\mathcal{E}_3' \cup \mathcal{E}_3'') \leq O(n^{-1/4})$. Again, by the *union bound*, we obtain

$$\Pr(\text{FAIL} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3) \leq O(n^{-1/4}).$$

Runtime. When Algorithm **??** is run *once*, we ensure it terminates in linear time (due to the size condition on $\mathcal{C}$), and it returns the correct median **with high probability** (i.e., the correct median is returned with probability $1 - O(n^{-1/4})$)—therefore the randomized algorithm is a *Monte Carlo* algorithm in this case. However, we can run this algorithm multiple times until the first success to make the algorithm *Las Vegas*—this makes the number of required trials a geometric random variable; thus, the expected number of trials is $\frac{1}{1-O(n^{-1/4})} = O\left(\frac{n^{1/4}}{n^{1/4}-c}\right)$ for some constant $c > 0$, and since each trial has a linear runtime, this means that the average runtime in the *Las Vegas* version is therefore

$$O\left(\frac{n^{5/4}}{n^{1/4} - c}\right).$$

□

3 Metropolis-Hastings Algorithm

The goal of this randomized algorithm is to sample from a complex target distribution via the construction of an *ergodic*, *time-reversible* Markov chain whose stationary distribution *is* the target distribution. Recall that an ergodic Markov chain is a chain whose states are *ergodic*; that is, they are aperiodic, irreducible, and positive recurrent. We then introduce the definition of a time-reversible Markov chain with some important lemmas to make sense of the algorithm.

Definition 1 (Time-reversible Markov chain)**.** *Let $\vec{P}$ be a transition matrix and let $\overline{\pi} = (\pi_0, \ldots, \pi_n)$ be a stationary distribution with $\sum_i \pi_i = 1$. A Markov chain is said to be* ***time-reversible*** *when, for any pair of states i, j,*

$$\pi_i P_{i,j} = \pi_j P_{j,i}.$$

To understand why this property is interesting, we must first look at a general reversed sequence of a Markov chain.

Lemma 1. *We consider a finite Markov chain on n states with $\overline{\pi}$ and $\vec{P}$ as stationary distribution and transition matrix, respectively. Suppose we have obtained a sequence of $m+1$ steps, $X_0, \ldots, X_{m-1}, X_m$, and the chain was started in stationary distribution (i.e., $X_0 \sim \overline{\pi}$). Then, we reverse the sequence to get $X_m, X_{m-1}, \ldots, X_0$. The reversed sequence is Markovian (with respect to $\overline{\pi}$) with the transition probability*

$$Q_{i,j} = \frac{\pi_j P_{j,i}}{\pi_i}.$$

Note that if our Markov chain is time-reversible, it is not difficult to see that $Q_{i,j} = P_{i,j}$.

Proof. First, we prove that the reverse chain is Markovian. We can do this by induction. Start with the base case (**Notation.** x_t indicates the realization of X_t):

$$\begin{aligned}
\Pr(x_{m-2} \mid x_{m-1}, x_m) &= \frac{\Pr(x_{m-1}, x_{m-2}) \Pr(x_m \mid x_{m-1}, x_{m-2})}{\Pr(x_{m-1}) \Pr(x_m \mid x_{m-1})} \\
&= \frac{\Pr(x_{m-1}, x_{m-2}) \Pr(x_m \mid x_{m-1})}{\Pr(x_{m-1}) \Pr(x_m \mid x_{m-1})} \qquad \text{(forward-time is Markovian)} \\
&= \frac{\Pr(x_{m-1}, x_{m-2})}{\Pr(x_{m-1})} \\
&= \Pr(x_{m-2} \mid x_{m-1}).
\end{aligned}$$

Suppose the claim holds true for all k states. Then,

$$\begin{aligned}
\Pr(x_{k-1} \mid x_k, x_{k+1}, \ldots, x_m) &= \frac{\Pr(x_{k-1}, \ldots, x_m)}{\Pr(x_k, \ldots, x_m)} \\
&= \frac{\Pr(x_m) \Pr(x_{m-1} \mid x_m) \Pr(x_{m-2} \mid x_{m-1}) \ldots \Pr(x_k \mid x_{k+1}) \Pr(x_{k-1} \mid x_k)}{\Pr(x_m) \Pr(x_{m-1} \mid x_m) \Pr(x_{m-2} \mid x_{m-1}) \ldots \Pr(x_k \mid x_{k+1})} \\
&= \Pr(x_{k-1} \mid x_k).
\end{aligned}$$

The second equality comes from the induction hypothesis and the multiplication rule. Thus, we establish that the reversed-sequence chain is indeed Markovian, as it has the memoryless property. Furthermore, we can compute the transition probability by the definition of conditional probability (when the chain started in *stationarity*):

$$\begin{aligned}
Q_{i,j} &= \Pr(X_{t-1} = j \mid X_t = i) \\
&= \frac{\Pr(X_t = i \mid X_{t-1} = j) \Pr(X_{t-1} = j)}{\Pr(X_t = i)} \\
&= \frac{P_{j,i}\, \pi_j}{\pi_i}.
\end{aligned}$$

□

Lemma 2. *Suppose we have a Markov chain with transition matrix $\vec{P}$ and stationary distribution $\overline{\pi}$. Suppose also that the chain is time-reversible; in that case, the reversed chain also has the same stationary distribution $\overline{\pi}$.*

Proof. From Definition **??**, we know that $\pi_i P_{i,j} = \pi_j P_{j,i}$. Consequently,

$$\begin{aligned}
\sum_i \pi_i P_{i,j} &= \pi_j \sum_i P_{j,i} \\
&= \pi_j.
\end{aligned}$$

This holds true for any j-th entry of $\overline{\pi} \cdot \vec{P}$. This proves that $\overline{\pi} = \overline{\pi}\vec{P}$. This fact, coupled with the result in Lemma **??**, shows why $\vec{Q} = \vec{P}$ where $\vec{Q}$ is the reversed chain's transition matrix. □

From Lemma **??** and Lemma **??**, we know that if a Markov chain is time reversible, then the transition matrix and the stationary distribution are the same for forward-time and reverse-time. Let us use this to our advantage in constructing an algorithm. **Note.** On a finite state space, *ergodic* usually means *irreducible and aperiodic*. Irreducibility implies existence and uniqueness of a stationary distribution; aperiodicity ensures convergence to it from any initial distribution (see Theorem 19 of `My Notes for Discrete Probability: Detailed Explanations on Concepts and Proofs`).

Algorithm 3 METROPOLIS-HASTINGS ALGORITHM

METROPOLIS

1: Select an arbitrary state $X_0 = x$; compute π_x.
2: **for** all i **do**
3: Propose $X_{i+1} = y$ such that $y \in N(x)$; compute π_y. ▷ $N(x)$ refers to the neighbor of x
4: **if** $U[0,1] < \pi_y/\pi_x$ **then**
5: $x \leftarrow y$; $\pi_x \leftarrow \pi_y$. ▷ assume all $\pi_x > 0$
6: **else**
7: Set $X_{i+1} = x$.
8: **return**

Analysis. By the construction of the algorithm, every state is reachable, which implies irreducibility. Moreover, Step 6 of the algorithm ensures the probability of a self-loop, which then implies aperiodicity. Thus, the chain is ergodic. Notice that Step 4 states that the state y is accepted with probability $\min(1, \pi_y/\pi_x)$.

1. If $\pi_x \leq \pi_y$, then $P_{x,y} = 1$ and $P_{y,x} = \pi_x/\pi_y$. This implies that $\pi_x P_{x,y} = \pi_y P_{y,x}$.
2. If $\pi_x > \pi_y$, then $P_{x,y} = \pi_y/\pi_x$ and $P_{y,x} = 1$. This implies that $\pi_x P_{x,y} = \pi_y P_{y,x}$.

So, either way, the chain is time-reversible. Hence, the stationary distribution is uniquely determined by the values of π_x, by Lemma **??**. Also, π_x can be interpreted as a *score* of how close x is to our target distribution. □

Acknowledgement

I would like to acknowledge *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*, 2nd edition, by Michael Mitzenmacher and Eli Upfal, as the primary source of reference for these notes.