Randomized Median Selection

A Sampling-Based Approach

Joseph Chuang-Chieh Lin

Department of Computer Science & Engineering,
National Taiwan Ocean University

Spring 2026


Outline

Outline

Learning goals

- Understand the sampling-based median algorithm.
- See why one run always takes linear time.
- Prove that one run fails with probability at most $n^{-1/4}$.
- Interpret the algorithm as a Monte Carlo algorithm, and see how repetition gives a Las Vegas algorithm.

Median selection problem

Given a set S of n distinct elements from a totally ordered universe, a median is an element $m \in S$ such that

$$\#\{x \in S : x \leq m\} \geq \lceil n/2 \rceil \quad \text{and} \quad \#\{x \in S : x \geq m\} \geq \lceil n/2 \rceil.$$

Consider the simplifying assumptions:

- n is odd;
- all elements of S are distinct.

Hence m is exactly the $\left(\frac{n+1}{2}\right)$th element in sorted order.

Deterministic baselines

Two classical approaches

- Sort all of S: $O(n \log n)$ time.
- A deterministic $O(n)$ time selection algorithm, but more complicated. [Blum, Floyd, Pratt, Rivest, Tarjan; JCSS 1973]

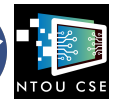
Deterministic baselines

Two classical approaches

- Sort all of S: $O(n \log n)$ time.
- A deterministic $O(n)$ time selection algorithm, but more complicated. [Blum, Floyd, Pratt, Rivest, Tarjan; JCSS 1973]

Our Objective

Design a *simpler* randomized algorithm that still runs in linear time per attempt and succeeds with high probability.

Why randomness helps here

The algorithm does *not* try to locate the exact median immediately.

Why randomness helps here

The algorithm does *not* try to locate the exact median immediately. Instead, it tries to find two elements $d, u \in S$ such that

1. $d \leq m \leq u$;
2. the interval

$$C := \{x \in S : d \leq x \leq u\}$$

is small enough to sort quickly.

Why randomness helps here

The algorithm does *not* try to locate the exact median immediately. Instead, it tries to find two elements $d, u \in S$ such that

1. $d \leq m \leq u$;
2. the interval

$$C := \{x \in S : d \leq x \leq u\}$$

is small enough to sort quickly.

If we can find such a narrow interval around the true median, then the remaining work is easy.

Desired sandwich interval

We want

$$d \leq m \leq u \quad \text{and} \quad |C| = O(n^{3/4}).$$

Desired sandwich interval

We want

$$d \leq m \leq u \quad \text{and} \quad |C| = O(n^{3/4}).$$

Why is this enough?


Desired sandwich interval

We want

$$d \le m \le u \quad \text{and} \quad |C| = O(n^{3/4}).$$

Why is this enough?

- Scan S once to count $\ell_d := \#\{x \in S : x < d\}$.
- Sort only the much smaller set C.


Desired sandwich interval

We want

$$d \le m \le u \quad \text{and} \quad |C| = O(n^{3/4}).$$

Why is this enough?

- Scan S once to count $\ell_d := \#\{x \in S : x < d\}$.
- Sort only the much smaller set C.
  - Since $|C| = O(n^{3/4})$, sorting costs

$$O(|C| \log |C|) = O(n^{3/4} \log n) = o(n).$$


Desired sandwich interval

We want

$$d \le m \le u \quad \text{and} \quad |C| = O(n^{3/4}).$$

Why is this enough?

- Scan S once to count $\ell_d := \#\{x \in S : x < d\}$.
- Sort only the much smaller set C.
  - Since $|C| = O(n^{3/4})$, sorting costs

$$O(|C| \log |C|) = O(n^{3/4} \log n) = o(n).$$

- Output the $\big(\lceil n/2 \rceil - \ell_d + 1\big)$th smallest element of C.

Desired sandwich interval

We want

$$d \le m \le u \quad \text{and} \quad |C| = O(n^{3/4}).$$

Why is this enough?

- Scan S once to count $\ell_d := \#\{x \in S : x < d\}$.
- Sort only the much smaller set C.
  - Since $|C| = O(n^{3/4})$, sorting costs

$$O(|C| \log |C|) = O(n^{3/4} \log n) = o(n).$$

- Output the $\big(\lceil n/2 \rceil - \ell_d + 1\big)$th smallest element of C.

Key idea

Shrink the search space from all of S down to a small candidate interval C.


Why is the total cost is still linear?

Suppose we manage to ensure

$$|C| = O(n^{3/4}).$$

Why is the total cost is still linear?

Suppose we manage to ensure

$$|C| = O(n^{3/4}).$$

Then an optimal comparison-based sorting algorithm sorts C in time

$$O(|C| \log |C|) = o(n).$$

Why is the total cost is still linear?

Suppose we manage to ensure

$$|C| = O(n^{3/4}).$$

Then an optimal comparison-based sorting algorithm sorts C in time

$$O(|C| \log |C|) = o(n).$$

So the total cost becomes

$$\underbrace{O(n)}_{\text{scan } S} + \underbrace{o(n)}_{\text{sort } C} = O(n).$$


Why is the total cost is still linear?

Suppose we manage to ensure

$$|C| = O(n^{3/4}).$$

Then an optimal comparison-based sorting algorithm sorts C in time

$$O(|C| \log |C|) = o(n).$$

So the total cost becomes

$$\underbrace{O(n)}_{\text{scan } S} + \underbrace{o(n)}_{\text{sort } C} = O(n).$$

Main challenge

How to find d and u quickly, without knowing the median?


Outline

Sampling strategy

Take a random multiset R of size

$$|R| = n^{3/4}$$

by sampling *with replacement* uniformly from S.

Sampling strategy

Take a random multiset R of size

$$|R| = n^{3/4}$$

by sampling *with replacement* uniformly from S.

Intuition:

- the median of R should be near the median of S;
- so elements slightly below and slightly above the sample median should bracket the true median of S.

Sampling strategy

Take a random multiset R of size

$$|R| = n^{3/4}$$

by sampling *with replacement* uniformly from S.

Intuition:

- the median of R should be near the median of S;
- so elements slightly below and slightly above the sample median should bracket the true median of S.

Why with replacement?

- easier to implement;
- easier to analyze because the samples are independent.

How d and u are chosen?

After sorting R, define:

$$d = \text{the } \left(\frac{1}{2}n^{3/4} - \sqrt{n}\right)\text{th smallest element of } R,$$

$$u = \text{the } \left(\frac{1}{2}n^{3/4} + \sqrt{n}\right)\text{th smallest element of } R.$$

d and u are symmetrically around the sample median, with a buffer of $\sqrt{n}$ sample positions on each side.

How d and u are chosen?

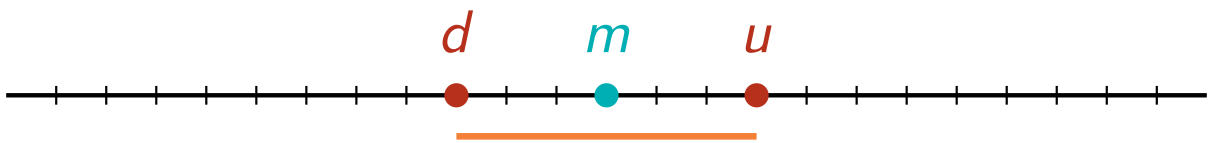
After sorting R, define:

$$d = \text{the } \left(\frac{1}{2}n^{3/4} - \sqrt{n}\right)\text{th smallest element of } R,$$

$$u = \text{the } \left(\frac{1}{2}n^{3/4} + \sqrt{n}\right)\text{th smallest element of } R.$$

d and u are symmetrically around the sample median, with a buffer of $\sqrt{n}$ sample positions on each side.

- This buffer is what makes the true median fall inside $[d, u]$ *with high probability*.



$$C = \{x \in S : d \leq x \leq u\}$$

Goal: m lies in a "short" interval that can be sorted cheaply.

- d and u are selected from the sample R.
- C is defined in the original set S.

Algorithm: Randomized Median Algorithm (RMA)

Input / output

Input: a set S of n elements over a totally ordered universe.
Output: the median element of S, denoted by m.

1. Pick a multiset R of $n^{3/4}$ elements from S, independently and uniformly at random with replacement.
2. Sort R.
3. Let d be the $\left(\frac{1}{2}n^{3/4} - \sqrt{n}\right)$th smallest element of R.
4. Let u be the $\left(\frac{1}{2}n^{3/4} + \sqrt{n}\right)$th smallest element of R.
5. Compare every element of S with d and u to compute
   $$C = \{x \in S : d \le x \le u\}, \quad \ell_d = \#\{x \in S : x < d\}, \quad \ell_u = \#\{x \in S : x > u\}.$$
6. If $\ell_d > n/2$ or $\ell_u > n/2$, output FAIL.
7. If $|C| \le 4n^{3/4}$, sort C; otherwise output FAIL.
8. Output the $\left(\lceil n/2 \rceil - \ell_d + 1\right)$th smallest element of C.

Purposes of each step

Step	Purpose
1–4	Use a random sample to guess a narrow interval around the true median.
5	Translate the sample interval $[d, u]$ back to the full set S.
6	Check whether the true median could have escaped outside $[d, u]$.
7	Guarantee the sorting work remains sublinear.
8	Recover the exact rank of the median inside C.

Numerical scale of the parameters

Take $n = 10^4$ as a sanity check.

$$n^{3/4} = 1{,}000, \qquad \sqrt{n} = 100.$$

So the algorithm:

- samples only 1,000 items from 10,000;
- chooses d and u at sample ranks $500 - 100 = 400$ and $500 + 100 = 600$;
- accepts only if $|C| \leq 4n^{3/4} = 4{,}000$.

We pay linear cost to isolate a candidate interval that is much smaller than the whole input and likely to contain the true median.


Outline

The Main Theorem

Theorem

Algorithm RMA terminates in linear time, and if it does not output FAIL, *then it outputs the correct median element of the input set S.*

This theorem separates the analysis into two parts:

1. **deterministic correctness conditioned on no failure**;
2. **probability of failure**.

Why step 6 protects correctness

Suppose the true median m is *not* in C. Then either:


Why step 6 protects correctness

Suppose the true median m is *not* in C. Then either:

- $m < d \Rightarrow$ more than $n/2$ elements are smaller than d

Why step 6 protects correctness

Suppose the true median m is *not* in C. Then either:

- $m < d \Rightarrow$ more than $n/2$ elements are smaller than $d \Rightarrow \ell_d > n/2$;

Why step 6 protects correctness

Suppose the true median m is *not* in C. Then either:

- $m < d \Rightarrow$ more than $n/2$ elements are smaller than $d \Rightarrow \ell_d > n/2$;
- or $m > u \Rightarrow$ more than $n/2$ elements are larger than u

Why step 6 protects correctness

Suppose the true median m is *not* in C. Then either:

- $m < d \Rightarrow$ more than $n/2$ elements are smaller than $d \Rightarrow \ell_d > n/2$;
- or $m > u \Rightarrow$ more than $n/2$ elements are larger than $u \Rightarrow \ell_u > n/2$.

Why step 6 protects correctness

Suppose the true median m is *not* in C. Then either:

- $m < d \Rightarrow$ more than $n/2$ elements are smaller than $d \Rightarrow \ell_d > n/2$;
- or $m > u \Rightarrow$ more than $n/2$ elements are larger than $u \Rightarrow \ell_u > n/2$.

Therefore, if $m \notin C$, step 6 forces the algorithm to output FAIL.

So if the algorithm reaches step 8, the true median must already lie in C.

Why step 8 returns the median once $m \in C$

Assume $m \in C$.

- There are exactly ℓ_d elements of S smaller than d, and all remaining elements smaller than m that are not among those ℓ_d lie inside C.
- Hence inside the sorted order of C, the median sits at position

$$\lceil n/2 \rceil - \ell_d + 1.$$

Under the event $m \in C$, step 8 returns the exact median of S.

Why one run is always linear time

The cost of a single run is:

- sample R:

Why one run is always linear time

The cost of a single run is:

- sample R: $O(n^{3/4})$;
- sort R:

Why one run is always linear time

The cost of a single run is:

- sample R: $O(n^{3/4})$;
- sort R: $O(n^{3/4} \log n)$;
- scan all of S to form C, ℓ_d, ℓ_u:

Why one run is always linear time

The cost of a single run is:

- sample R: $O(n^{3/4})$;
- sort R: $O(n^{3/4} \log n)$;
- scan all of S to form C, ℓ_d, ℓ_u: $O(n)$;
- sort C only if $|C| \leq 4n^{3/4}$, which costs

Why one run is always linear time

The cost of a single run is:

- sample R: $O(n^{3/4})$;
- sort R: $O(n^{3/4} \log n)$;
- scan all of S to form C, ℓ_d, ℓ_u: $O(n)$;
- sort C only if $|C| \leq 4n^{3/4}$, which costs $O(n^{3/4} \log n)$.

Why one run is always linear time

The cost of a single run is:

- sample R: $O(n^{3/4})$;
- sort R: $O(n^{3/4} \log n)$;
- scan all of S to form C, ℓ_d, ℓ_u: $O(n)$;
- sort C only if $|C| \leq 4n^{3/4}$, which costs $O(n^{3/4} \log n)$.

$\because n^{3/4} \log n = o(n)$, the total is

$$O(n) + o(n) = O(n).$$

Why one run is always linear time

The cost of a single run is:

- sample R: $O(n^{3/4})$;
- sort R: $O(n^{3/4} \log n)$;
- scan all of S to form C, ℓ_d, ℓ_u: $O(n)$;
- sort C only if $|C| \le 4n^{3/4}$, which costs $O(n^{3/4} \log n)$.

$\because n^{3/4} \log n = o(n)$, the total is

$$O(n) + o(n) = O(n).$$

If $|C|$ is too large, step 7 stops immediately with FAIL instead of spending superlinear time.

Deterministic summary so far

What is already proved without probability?

1. Every execution halts in $O(n)$ time.
2. Any non-FAIL output is correct.

What remains?

Bound the probability that the algorithm outputs FAIL.

Outline

Three bad events

Define

$$E_1 : \ Y_1 := \#\{r \in R : r \le m\} < \frac{1}{2}n^{3/4} - \sqrt{n},$$

$$E_2 : \ Y_2 := \#\{r \in R : r \ge m\} < \frac{1}{2}n^{3/4} - \sqrt{n},$$

$$E_3 : \ |C| > 4n^{3/4}.$$

Intuitive meanings:

- E_1: too few sampled points land at or below the true median;
- E_2: too few sampled points land at or above the true median;
- E_3: the candidate interval in the full set is still too large.

Failure is exactly these bad events

Lemma

Algorithm RMA fails if and only if at least one of E_1, E_2, E_3 occurs.

Failure is exactly these bad events

Lemma

Algorithm RMA fails if and only if at least one of E_1, E_2, E_3 occurs.

- Step 7 fails exactly when $|C| > 4n^{3/4}$, namely event E_3.

Failure is exactly these bad events

Lemma

Algorithm RMA fails if and only if at least one of E_1, E_2, E_3 occurs.

- Step 7 fails exactly when $|C| > 4n^{3/4}$, namely event E_3.
- Step 6 fails when $\ell_d > n/2$ or $\ell_u > n/2$.

Failure is exactly these bad events

Lemma

Algorithm RMA fails if and only if at least one of E_1, E_2, E_3 occurs.

- Step 7 fails exactly when $|C| > 4n^{3/4}$, namely event E_3.
- Step 6 fails when $\ell_d > n/2$ or $\ell_u > n/2$.
- Those two conditions are equivalent to E_1 and E_2, respectively.

Failure is exactly these bad events

Lemma

Algorithm RMA fails if and only if at least one of E_1, E_2, E_3 occurs.

- Step 7 fails exactly when $|C| > 4n^{3/4}$, namely event E_3.
- Step 6 fails when $\ell_d > n/2$ or $\ell_u > n/2$.
- Those two conditions are equivalent to E_1 and E_2, respectively.

Proof Idea

Bound $\Pr[E_1]$, $\Pr[E_2]$, and $\Pr[E_3]$ separately, then apply the union bound.

Failure is exactly these bad events

Lemma

Algorithm RMA fails if and only if at least one of E_1, E_2, E_3 occurs.

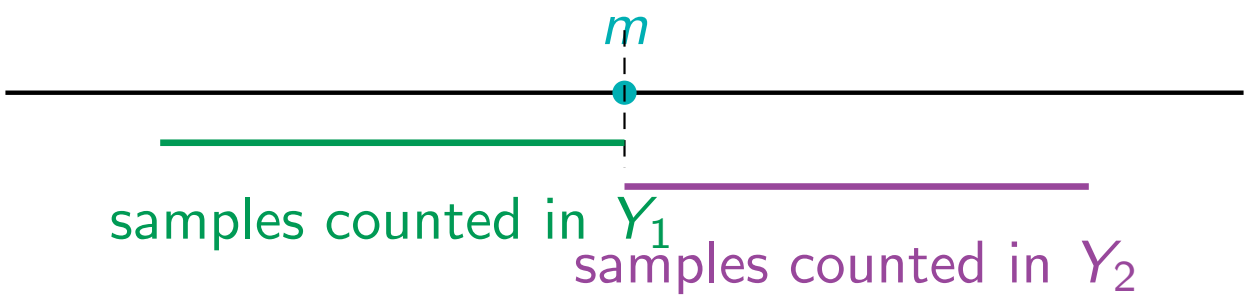
- Step 7 fails exactly when $|C| > 4n^{3/4}$, namely event E_3.
- Step 6 fails when $\ell_d > n/2$ or $\ell_u > n/2$.
- Those two conditions are equivalent to E_1 and E_2, respectively.

Proof Idea

Bound $\Pr[E_1]$, $\Pr[E_2]$, and $\Pr[E_3]$ separately, then apply the union bound.

$$\Pr[E_1 \cup E_2 \cup E_3] \leq \Pr[E_1] + \Pr[E_2] + \Pr[E_3].$$

Visual intuition for E_1 and E_2



If Y_1 is too small, then d ends up to the *right* of m.
If Y_2 is too small, then u ends up to the *left* of m.

Bounding $\Pr[E_1]$: define indicators

For each sampled point $r_i \in R$, define

$$X_i = \begin{cases} 1, & r_i \leq m, \\ 0, & r_i > m. \end{cases}$$

Then

$$Y_1 = \sum_{i=1}^{n^{3/4}} X_i.$$

Bounding $\Pr[E_1]$: define indicators

For each sampled point $r_i \in R$, define

$$X_i = \begin{cases} 1, & r_i \leq m, \\ 0, & r_i > m. \end{cases}$$

Then

$$Y_1 = \sum_{i=1}^{n^{3/4}} X_i.$$

Because the sample is taken with replacement, $X_1, X_2, \ldots, X_{n^{3/4}}$ are independent Bernoulli trials.

Expectation of Y_1

Since m is the median and all elements are distinct,

$$\Pr[X_i = 1] = \frac{(n-1)/2 + 1}{n} = \frac{1}{2} + \frac{1}{2n}.$$

Expectation of Y_1

Since m is the median and all elements are distinct,

$$\Pr[X_i = 1] = \frac{(n-1)/2+1}{n} = \frac{1}{2} + \frac{1}{2n}.$$

Therefore

$$\mathbb{E}[Y_1] = n^{3/4}\left(\frac{1}{2} + \frac{1}{2n}\right) = \frac{1}{2}n^{3/4} + \frac{1}{2n^{1/4}}.$$

Expectation of Y_1

Since m is the median and all elements are distinct,

$$\Pr[X_i = 1] = \frac{(n-1)/2 + 1}{n} = \frac{1}{2} + \frac{1}{2n}.$$

Therefore

$$\mathbb{E}[Y_1] = n^{3/4}\left(\frac{1}{2} + \frac{1}{2n}\right) = \frac{1}{2}n^{3/4} + \frac{1}{2n^{1/4}}.$$

Comparison with the bad threshold

Event E_1 asks whether Y_1 drops below

$$\frac{1}{2}n^{3/4} - \sqrt{n},$$

which is roughly $\sqrt{n}$ below its mean.

Variance of Y_1

- Because Y_1 is binomial with parameters $n^{3/4}$ and $\frac{1}{2} + \frac{1}{2n}$,

$$\text{Var}[Y_1] = n^{3/4}\left(\frac{1}{2} + \frac{1}{2n}\right)\left(\frac{1}{2} - \frac{1}{2n}\right) = \frac{1}{4}n^{3/4} - \frac{1}{4n^{5/4}} < \frac{1}{4}n^{3/4}.$$

Variance of Y_1

- Because Y_1 is binomial with parameters $n^{3/4}$ and $\frac{1}{2} + \frac{1}{2n}$,

$$\text{Var}[Y_1] = n^{3/4}\left(\frac{1}{2} + \frac{1}{2n}\right)\left(\frac{1}{2} - \frac{1}{2n}\right) = \frac{1}{4}n^{3/4} - \frac{1}{4n^{5/4}} < \frac{1}{4}n^{3/4}.$$

- So the standard deviation scale is about $n^{3/8}$, while the buffer in the algorithm is $\sqrt{n} = n^{1/2}$.

Chebyshev bound for E_1

Using Chebyshev's inequality,

$$\begin{aligned}
\Pr[E_1] &= \Pr\left[Y_1 < \frac{1}{2}n^{3/4} - \sqrt{n}\right] \\
&\leq \Pr\left[|Y_1 - \mathbb{E}[Y_1]| > \sqrt{n}\right] \\
&\leq \frac{\mathrm{Var}[Y_1]}{(\sqrt{n})^2} \\
&< \frac{\frac{1}{4}n^{3/4}}{n} = \frac{1}{4}n^{-1/4}.
\end{aligned}$$

Chebyshev bound for E_1

Using Chebyshev's inequality,

$$\begin{aligned}
\Pr[E_1] &= \Pr\left[Y_1 < \frac{1}{2}n^{3/4} - \sqrt{n}\right] \\
&\leq \Pr\left[|Y_1 - \mathbb{E}[Y_1]| > \sqrt{n}\right] \\
&\leq \frac{\mathrm{Var}[Y_1]}{(\sqrt{n})^2} \\
&< \frac{\frac{1}{4}n^{3/4}}{n} = \frac{1}{4}n^{-1/4}.
\end{aligned}$$

Lemma

$$\Pr[E_1] \leq \frac{1}{4}n^{-1/4}.$$

The bound for E_2

The event E_2 is symmetric to E_1.

The bound for E_2

The event E_2 is symmetric to E_1.

Cumulative progress

So far,

$$\Pr[E_1] + \Pr[E_2] \leq \frac{1}{2} n^{-1/4}.$$

Why E_3 is harder

Event E_3 is different:

$$E_3 : \ |C| > 4n^{3/4}.$$

- Even if the true median lies inside $[d, u]$, the interval could still be too wide.

Why E_3 is harder

Event E_3 is different:

$$E_3 : |C| > 4n^{3/4}.$$

- Even if the true median lies inside $[d, u]$, the interval could still be too wide.
- We must show that the chosen sample ranks make the resulting interval in S *unlikely to contain too many elements*.

Split E_3 into two one-sided events

- If $|C| > 4n^{3/4}$, then at least one of the following must happen:

Split E_3 into two one-sided events

- If $|C| > 4n^{3/4}$, then at least one of the following must happen:

  $E_{3,1}$: at least $2n^{3/4}$ elements of C are **greater** than m,

  $E_{3,2}$: at least $2n^{3/4}$ elements of C are **smaller** than m.

Split E_3 into two one-sided events

- If $|C| > 4n^{3/4}$, then at least one of the following must happen:

  $E_{3,1}$: at least $2n^{3/4}$ elements of C are **greater** than m,

  $E_{3,2}$: at least $2n^{3/4}$ elements of C are **smaller** than m.

- Thus,

$$\Pr[E_3] = \Pr[E_{3,1} \cup E_{3,2}] \leq \Pr[E_{3,1}] + \Pr[E_{3,2}].$$

Split E_3 into two one-sided events

- If $|C| > 4n^{3/4}$, then at least one of the following must happen:

  $E_{3,1}$: at least $2n^{3/4}$ elements of C are **greater** than m,

  $E_{3,2}$: at least $2n^{3/4}$ elements of C are **smaller** than m.

- Thus,

$$\Pr[E_3] = \Pr[E_{3,1} \cup E_{3,2}] \leq \Pr[E_{3,1}] + \Pr[E_{3,2}].$$

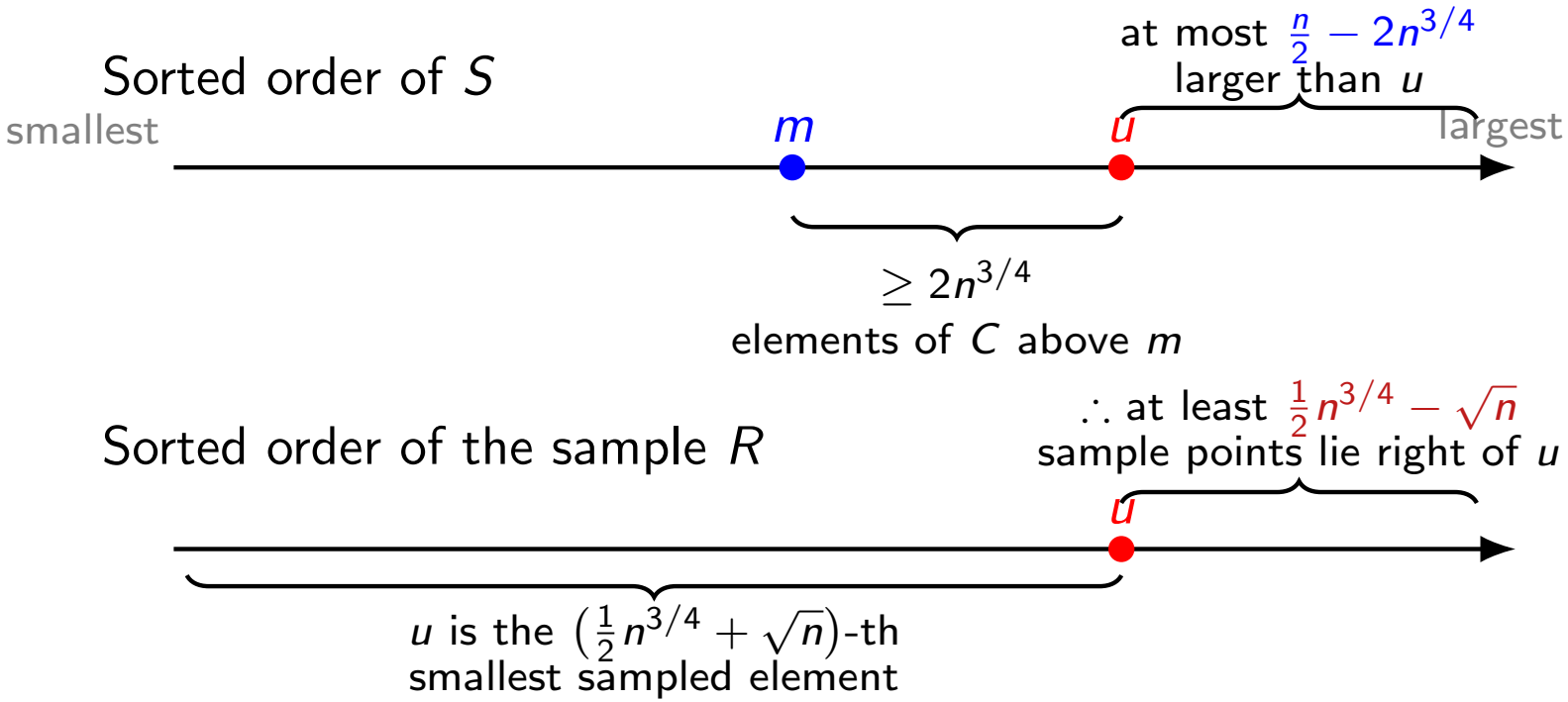
- By symmetry, it is enough to bound $\Pr[E_{3,1}]$.

What does $E_{3,1}$ imply? (1/3)

$E_{3,1}$: at least $2n^{3/4}$ elements of C are **greater** than m.

- If $E_{3,1}$ occurs, then u is too far above the median in the sorted order of S.

What does $E_{3,1}$ imply? (2/3)



$$E_{3,1} \implies \text{rank}_S(u) \geq \frac{n}{2} + 2n^{3/4}.$$

What does $E_{3,1}$ imply? (3/3)

- The rank of u in S is at least

$$\frac{1}{2}n + 2n^{3/4}.$$

- Among the sample points in R, at least $\frac{1}{2}n^{3/4} - \sqrt{n}$ samples must come from the $\frac{1}{2}n - 2n^{3/4}$ largest elements of S.

Define the variable for $E_{3,1}$

- Let X be the number of sampled points falling among the largest

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

elements of S.

Define the variable for $E_{3,1}$

- Let X be the number of sampled points falling among the largest

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

elements of S.

- Write $X = \sum_{i=1}^{n^{3/4}} X_i$, where

$$X_i = \begin{cases} 1, & \text{the } i\text{th sample lies in that top set,} \\ 0, & \text{otherwise.} \end{cases}$$

Define the variable for $E_{3,1}$

- Let X be the number of sampled points falling among the largest

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

elements of S.

- Write $X = \sum_{i=1}^{n^{3/4}} X_i$, where

$$X_i = \begin{cases} 1, & \text{the } i\text{th sample lies in that top set,} \\ 0, & \text{otherwise.} \end{cases}$$

- Then $E_{3,1}$ implies $X \geq \frac{1}{2}n^{3/4} - \sqrt{n}$.

Expectation and variance of X

The success probability for each X_i is

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

Expectation and variance of X

The success probability for each X_i is

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

Hence

Expectation and variance of X

The success probability for each X_i is

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

Hence

$$\mathbb{E}[X] = n^{3/4}\left(\frac{1}{2} - 2n^{-1/4}\right) = \frac{1}{2}n^{3/4} - 2\sqrt{n}.$$

Expectation and variance of X

The success probability for each X_i is

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

Hence

$$\mathbb{E}[X] = n^{3/4}\left(\frac{1}{2} - 2n^{-1/4}\right) = \frac{1}{2}n^{3/4} - 2\sqrt{n}.$$

Also,

Expectation and variance of X

The success probability for each X_i is

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

Hence

$$\mathbb{E}[X] = n^{3/4}\left(\frac{1}{2} - 2n^{-1/4}\right) = \frac{1}{2}n^{3/4} - 2\sqrt{n}.$$

Also,

$$\begin{aligned}\mathrm{Var}[X] &= n^{3/4}\left(\frac{1}{2} - 2n^{-1/4}\right)\left(\frac{1}{2} + 2n^{-1/4}\right) \\ &= \frac{1}{4}n^{3/4} - 4n^{1/4} < \frac{1}{4}n^{3/4}.\end{aligned}$$

Chebyshev bound for $E_{3,1}$

Now apply Chebyshev again:

$$\begin{aligned}
\Pr[E_{3,1}] &\leq \Pr\left[X \geq \frac{1}{2}n^{3/4} - \sqrt{n}\right] \\
&\leq \Pr\left[|X - \mathbb{E}[X]| \geq \sqrt{n}\right] \\
&\leq \frac{\mathrm{Var}[X]}{(\sqrt{n})^2} \\
&< \frac{\frac{1}{4}n^{3/4}}{n} = \frac{1}{4}n^{-1/4}.
\end{aligned}$$

By symmetry, $\Pr[E_{3,2}] \leq \frac{1}{4}n^{-1/4}$.

Chebyshev bound for $E_{3,1}$

Now apply Chebyshev again:

$$\begin{aligned}
\Pr[E_{3,1}] &\leq \Pr\left[X \geq \frac{1}{2}n^{3/4} - \sqrt{n}\right] \\
&\leq \Pr\left[|X - \mathbb{E}[X]| \geq \sqrt{n}\right] \\
&\leq \frac{\mathrm{Var}[X]}{(\sqrt{n})^2} \\
&< \frac{\frac{1}{4}n^{3/4}}{n} = \frac{1}{4}n^{-1/4}.
\end{aligned}$$

By symmetry, $\Pr[E_{3,2}] \leq \frac{1}{4}n^{-1/4}$. Therefore, $\Pr[E_3] \leq \frac{1}{2}n^{-1/4}$.

Finally, using the union bound

Combining the three bounds,

$$\Pr[E_1] + \Pr[E_2] + \Pr[E_3] \leq \frac{1}{4}n^{-1/4} + \frac{1}{4}n^{-1/4} + \frac{1}{2}n^{-1/4} = n^{-1/4}.$$

Theorem

The probability that Algorithm RMA fails is at most $n^{-1/4}$.

Equivalent success guarantee

A single run succeeds with probability at least

$$1 - n^{-1/4}.$$


What was the role of Chebyshev's inequality?

For each bad event, we:

1. express a sample count as a sum of independent Bernoulli indicators;
2. compute its mean and variance;
3. use Chebyshev to control deviations of size $\sqrt{n}$.

The algorithm is a concrete application of variance calculations and Chebyshev bounds, not yet of Chernoff bounds.

Outline

One-run guarantee

For one execution:

- running time is always $O(n)$;
- output is either the correct median or FAIL;
- failure probability is at most $n^{-1/4}$.

This is a *Monte Carlo* algorithm: bounded running time, but a small probability of failure.

From Monte Carlo to Las Vegas

Repeat Algorithm RMA until it succeeds.

Because the samples used in different runs are independent,

- each run succeeds with probability at least $1 - n^{-1/4}$;
- the number of runs until success is geometric.

From Monte Carlo to Las Vegas

Repeat Algorithm RMA until it succeeds.

Because the samples used in different runs are independent,

- each run succeeds with probability at least $1 - n^{-1/4}$;
- the number of runs until success is geometric.

Result

The repeated version always returns the correct median.

From Monte Carlo to Las Vegas

Repeat Algorithm RMA until it succeeds.

Because the samples used in different runs are independent,

- each run succeeds with probability at least $1 - n^{-1/4}$;
- the number of runs until success is geometric.

Result

The repeated version always returns the correct median.

So repetition converts the Monte Carlo algorithm into a *Las Vegas* algorithm.

Expected running time after repetition

Let $p \geq 1 - n^{-1/4}$ be the success probability of one run. For a geometric random variable with success probability p,

$$\mathbb{E}[\text{number of runs}] = \frac{1}{p}.$$

Expected running time after repetition

Let $p \geq 1 - n^{-1/4}$ be the success probability of one run. For a geometric random variable with success probability p,

$$\mathbb{E}[\text{number of runs}] = \frac{1}{p}.$$

Hence

$$\mathbb{E}[\text{total time}] =$$

Expected running time after repetition

Let $p \geq 1 - n^{-1/4}$ be the success probability of one run. For a geometric random variable with success probability p,

$$\mathbb{E}[\text{number of runs}] = \frac{1}{p}.$$

Hence

$$\mathbb{E}[\text{total time}] = O(n) \cdot \frac{1}{p} \leq$$

Expected running time after repetition

Let $p \geq 1 - n^{-1/4}$ be the success probability of one run. For a geometric random variable with success probability p,

$$\mathbb{E}[\text{number of runs}] = \frac{1}{p}.$$

Hence

$$\mathbb{E}[\text{total time}] = O(n) \cdot \frac{1}{p} \leq O(n) \cdot \frac{1}{1 - n^{-1/4}}$$

Expected running time after repetition

Let $p \geq 1 - n^{-1/4}$ be the success probability of one run. For a geometric random variable with success probability p,

$$\mathbb{E}[\text{number of runs}] = \frac{1}{p}.$$

Hence

$$\mathbb{E}[\text{total time}] = O(n) \cdot \frac{1}{p} \leq O(n) \cdot \frac{1}{1 - n^{-1/4}} \leq O(n) \cdot 2 \ \ (\text{for } n \geq 16).$$

Hence for sufficiently large n, this is still $O(n)$.

Expected running time after repetition

Let $p \geq 1 - n^{-1/4}$ be the success probability of one run. For a geometric random variable with success probability p,

$$\mathbb{E}[\text{number of runs}] = \frac{1}{p}.$$

Hence

$$\mathbb{E}[\text{total time}] = O(n) \cdot \frac{1}{p} \leq O(n) \cdot \frac{1}{1 - n^{-1/4}} \leq O(n) \cdot 2 \ \ (\text{for } n \geq 16).$$

Hence for sufficiently large n, this is still $O(n)$.

Conclusion

The repeated algorithm is Las Vegas with linear expected running time.


Why these parameters are balanced

There are three competing design goals:

1. sample small enough so sorting R is cheap;
2. buffer large enough so m lies between d and u with high probability;
3. candidate set C small enough so sorting C is cheap.

Why these parameters are balanced

There are three competing design goals:

1. sample small enough so sorting R is cheap;
2. buffer large enough so m lies between d and u with high probability;
3. candidate set C small enough so sorting C is cheap.

The algorithm chooses

$$|R| = n^{3/4}, \quad \text{buffer} = \sqrt{n}, \quad \text{accept if } |C| \leq 4n^{3/4},$$

which makes all three constraints fit together cleanly with Chebyshev bounds.

Big-picture summary

1. Sample $n^{3/4}$ elements.
2. Use the sample to define a narrow interval $[d, u]$ around the sample median.
3. Scan the original input to form the candidate set C.
4. Abort if C is too large or if the median appears to fall outside the interval.
5. Otherwise sort only C and recover the exact median.

Main theorem chain

One run takes $O(n)$ and satisfies $\Pr[\textsc{Fail}] \leq n^{-1/4}$. Hence repeated independent restarts give a Las Vegas algorithm with linear expected running time.

Takeaways

- Guess a structure cheaply;
- Verify it deterministically;
- Retry if needed.

Discussions