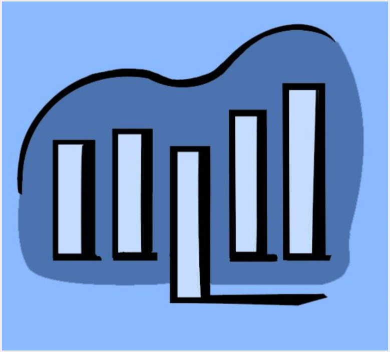
Randomized Algorithm to find median

using $1.5n + o[n]$ comparisons

Submitted to:
Prof. Surender Baswana

Submitted by:
Prashant Kumar (210750)

May 7, 2024



Input: Set S of size n

Aim: To design a randomized algorithm that computes the median of set S in $1.5n + o(n)$ comparisons with less error probability.

Idea and Intuition

The proposed methodology involves sampling a random subset, denoted as A, comprising a small fraction of the elements from the set S, typically with a size approximation of $|A| \approx n^{3/4}$. Given this size, sorting A can be accomplished in sub-linear time, denoted as $o(n)$.

For some t, the algorithm scrutinizes two elements, denoted as a and b, within A, where a represents the $(k/2 - t)^{th}$ element and b represents the $(k/2 + t + 1)^{th}$ element. We establish that with high probability a and b capture the median between them, and we find all the elements between a and b in S and accurately report the median from this subset of S.

Summary of Idea

- The median of S lies inclusively between a and b.
- The number of elements of S residing between a and b remains negligibly small, denoted as $o(n)$.

Rough Sketch of Algorithm

Sketch of Algorithm

Algorithm 1 Randomized Algorithm to Find Exact Median

Require: Set S
Ensure: Median of set S

$n = |S|, k = n^{3/4}, p = k/n$ and Fix some $t < k/2$
A = Random sample by picking each element of S independently with probability p.
Sort(A)
$(a, b) = (k/2 - t)^{th}$ and $(k/2 + t + 1)^{th}$ element of A respectively
(R_a, R_b) = Rank of $a \& b$ in set S
Q = Set of elements from S lying between a and b
if $R_a < n/2$ and $R_b > n/2$ **then**
    Sort(Q)
    return $Q[n/2 - R_a]$
end if

Inference about Parameters

Randomly select each element from set S with probability p, uniformly and independently.

Let us fix $\mathbb{E}(|A|) = n^{3/4} = k$ (say).

This implies our sampling probability p should be $n^{-1/4}$.

Upper bound on size of $|A|$

Lemma: $\mathbb{P}(|A| \geq 2k) \leq e^{-k/4}$.

Proof Let X_i be a random variable defined as follows:

$$X_i = \left\{ \begin{array}{ll} 1, & S_i \in A \\ 0, & S_i \notin A \end{array} \right\}, \quad \forall i = 1, 2, \ldots, n$$

Observe that $|A| = \sum_{i=1}^{n} X_i$. X_i's are independent Bernoulli random variables with probability of success being $p = n^{-1/4}$. By Chernoff bound, we have

$$\mathbb{P}(|A| \geq 2k) = \mathbb{P}\Big(\sum_{i=1}^{n} X_i \geq (1+1)k \Big) \leq e^{-k/4}$$

Define event B : a and b fails to capture median between them.

$$\begin{aligned}\mathbb{P}(B) &= \mathbb{P}(\text{Both } a \text{ and } b \text{ lies same side of median of S}) \\ &= \mathbb{P}(\text{Both } a \text{ and } b \text{ lies right side of median}) \\ &\quad + \mathbb{P}(\text{Both } a \text{ and } b \text{ lies left side of median})\end{aligned}$$

Define:
L := Subset of elements in set S, not surpassing the median
U := Subset of elements in set S, surpassing the median.

Let $Y = |L \cap A|$. Note that a ranks $k/2 - t$ in a random sample A. a must belongs to L. If the random sample has less than $k/2 - t$ elements, selected from L, then $a \in U$, i.e. if $Y < k/2 - t$, then $a \in U$ and a and b both lie right off the median. Similarly, if $Y > k/2 + t$, then $b \in L$ and a and b lie left of the median.

$$\mathbb{P}(B) = \mathbb{P}(Y < k/2 - t) + \mathbb{P}(Y > k/2 + t)$$

Lemma $\mathbb{P}(Y < k/2 - t) \leq e^{-t^2/k}$ and $\mathbb{P}(Y > k/2 + t) \leq e^{-2t^2/3k}$.
Proof Let us define RV Y_i as following:

$$Y_i = \left\{ \begin{array}{ll} 1, & L_i \in A \\ 0, & L_i \notin A \end{array} \right\}, \quad \forall i = 1, 2, .., n/2$$

Observe that $Y = \sum_{i=1}^{n/2} Y_i$. Since each element in S (so in L) is selected randomly uniformly Independently, We can use the result from Chernoff's bound here. By Chernoff's bound (Lower tail), we get

$$\mathbb{P}(Y < k/2 - t) = \mathbb{P}\left(\sum_{i=1}^{n/2} Y_i < \left(1 - \frac{2t}{k}\right) k/2 \right) \leq e^{-t^2/k}$$

$\mathbb{P}$(FAILING TO CAPTURE THE MEDIAN BETWEEN a AND b)

By Chernoff's bound (Upper tail) and observing that $2t/k < 1$, we get

$$\mathbb{P}(Y < k/2 + t) = \mathbb{P}\left(\sum_{i=1}^{n/2} Y_i < \left(1 + \frac{2t}{k}\right)k/2\right) \leq e^{-2t^2/3k}$$

We get,

$$\mathbb{P}(B) \leq e^{-t^2/k} + e^{-2t^2/3k} < 2e^{-2t^2/3k} = \frac{2}{n^c}$$

for $t = \sqrt{1.5ck\log(n)}$.

Note : It is important to note that we can not choose c as large as we want because...

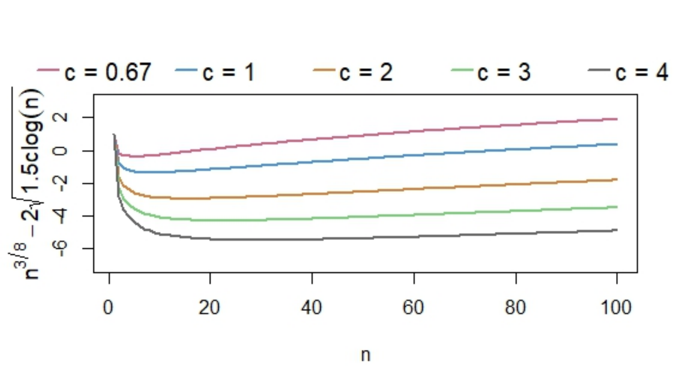


Figure: Constraint on parameter c

Did a and b are close enough?

We need to ensure the upper bound on the size of Q. As mentioned, the expected value of $|Q|$ is $2t/p$. We need to *ABORT* the algorithm if $|Q|$ deviates significantly from its expected size. We show that with high probability, the ranks, R_a and R_b are close to $n/2$.

Lemma: $\mathbb{P}(R_a < n/2 - \mathbb{E}(|Q|)) \leq exp\big(-t^2/2(k-4t\big)$ and $\mathbb{P}(R_b > n/2 + \mathbb{E}(|Q|)) \leq exp\big(-t^2/2(k-4t\big)$.

Proof we will establish the fact that $R_b - R_a = |Q| = o(n)$ with high probability. Let T_s be the set of small elements with a rank of less than $n/2 - \mathbb{E}(|Q|)$ in set S if sorted. Define

$$Y_s := |T_s \cap A|$$

Finding the probability $R_a < n/2 - \mathbb{E}(|Q|)$ is equivalent to finding the probability that the number of elements sampled from T_s is greater than or equal to $k/2 - t$, i.e. $Y_s > k/2 - t$. We get

$$\mathbb{P}(R_a < n/2 - \mathbb{E}(|Q|)) = \mathbb{P}(Y_s \geq k/2 - t)$$

Let us define RV $Y_s^{(i)}$ as follows:

$$Y_s^{(i)} = \left\{ \begin{array}{ll} 1, & T_{s_i} \in A \\ 0, & T_{s_i} \notin A \end{array} \right\}, \quad \forall i = 1, 2, \ldots, n/2 - \mathbb{E}(|Q|)$$

Observe that $Y_s = \sum_{i=1}^{n/2-\mathbb{E}(|Q|)} Y_s^{(i)}$.

By linearity of expectation, we have

$$\mathbb{E}(Y_s) = \mathbb{E}\left(\sum_{i=1}^{n/2-\mathbb{E}(|Q|)} Y_s^{(i)}\right) = p(n/2 - \mathbb{E}(|Q|)) = p(n/2 - 2t/p) = k/2 - 2t$$

By applying Chernoff's bound on Y_s, we have

$$\begin{aligned}
\mathbb{P}(Y_s \geq k/2 - t) &= \mathbb{P}\left(\sum_{i=1}^{n/2-\mathbb{E}(|Q|)} Y_s^{(i)} \geq (k/2 - 2t)\left(1 + \frac{2t}{k-4t}\right)\right) \\
&\leq exp\left(-\frac{\left(\frac{2t}{k-4t}\right)^2 (k/2 - 2t)}{4}\right) \\
&= exp\left(-\frac{t^2}{2(k-4t)}\right)
\end{aligned}$$

Error Probability

$$
\begin{aligned}
\text{Error Probability} &= \mathbb{P}(\text{Algorithm gone through bad random sample}) \\
&< \mathbb{P}(|A| > 2k) + \mathbb{P}(R_a > n/2 \text{ or} R_a < n/2 - 2t/p) \\
&\qquad + \mathbb{P}(R_b < n/2 \text{ or } R_b > n/2 + 2t/p) \\
&< \mathbb{P}(|A| > 2k) + \mathbb{P}(R_a > n/2) + \mathbb{P}(R_a < n/2 - 2t/p) \\
&\qquad + \mathbb{P}(R_b < n/2) + \mathbb{P}(R_b > n/2 + 2t/p) \\
&\leq e^{-k/4} + e^{-t^2/k} + e^{-t^2/2(k-4t)} + e^{-2t^2/3k} + e^{-t^2/2(k-4t)} \\
&< e^{-k/4} + e^{-t^2/k} + e^{-t^2/2k} + e^{-2t^2/3k} + e^{-t^2/2k} \\
&< e^{-k/4} + 4e^{-t^2/k} \\
&= e^{-k/4} + \frac{4}{n^{2/3}} = O(1/n^{O(1)})
\end{aligned}
$$

Final Algorithm

Final Algorithm

Algorithm 2 Randomized Algorithm to Find Exact Median

Require: Set S
Ensure: Median of set S

$n = |S|, p = n^{-1/4}$, $k = n^{3/4}$ and $t = \sqrt{k \log(n)}$.
if $(n < 10)$ **then**
  return median of S by sorting S
end if
A = Random sample by picking each element of S independently with probability p.
if $(|A| > 2k)$ **then** ▷ $O(1)$ comparisons
  return STOP, we have gone through bad sample
end if
Sort(A) ▷ $O(k \log(k)) = o(n)$ comparisons
$a = (k/2 - t)^{th}$ element of A
$b = (k/2 + t + 1)^{th}$ element of A

$(R_a, S') =$ (Rank of a in set S, Set of elements from S greater than s) ▷ n comparisons

if $(R_a > n/2$ or $R_a < n/2 - 2t/p)$ **then** ▷ $O(1)$ comparisons

  return STOP, we have gone through bad sample

end if

$(R_b, Q) =$ (Rank of b in set S,Set of elements from S' smaller than b) ▷ $0.5n + o(n)$ comparisons

if $(R_b < n/2$ or $R_b > n/2 + 2t/p)$ **then** ▷ $O(1)$ comparisons

  return STOP, we have gone through bad sample

end if

Sort(Q) ▷ $O\left(\frac{4t}{p}\log\left(\frac{4t}{p}\right)\right) = o(n)$ comparisons

Q = Set of elements from S lying between a and b

return $Q[n/2 - R_a]$

Total Comparisons

$$
\begin{aligned}
\text{No. of comaprison} &= O(1) + O(k\log(k)) + n + O(1) + 0.5 + O(4t/p) \\
&\qquad + O(1) + O\Big(\frac{4t}{p}\log\big(\frac{4t}{p}\big)\Big) \\
&= O(n^{0.75}\log(n^{0.75})) + 1.5n + O(n^{5/8}\log(n^{5/8})) \\
&= 1.5n + O(n^{0.75}\log(n)) \\
&= 1.5n + o(n)
\end{aligned}
$$

Experiments and Results

Experiment

To test the consistency and error probability of the algorithm, we ran it on the random inputs of size $n = 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8$ each 2000 times.

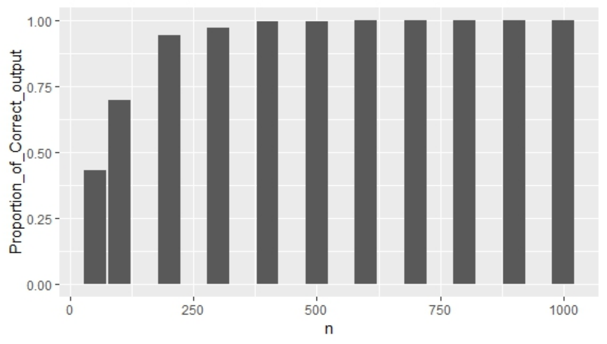


Figure: Proportion of correct output vs n

Table: Input Size vs Accuracy

Input size (n)	Number of correct runs out of 2000	Proportion
50	858	0.4290
100	1394	0.6970
200	1888	0.9440
300	1946	0.9730
400	1991	0.9955
500	1996	0.9980
600	1999	0.9995
700	2000	1.0000
800	1999	0.9995
900	2000	1.0000
1000	2000	1.0000

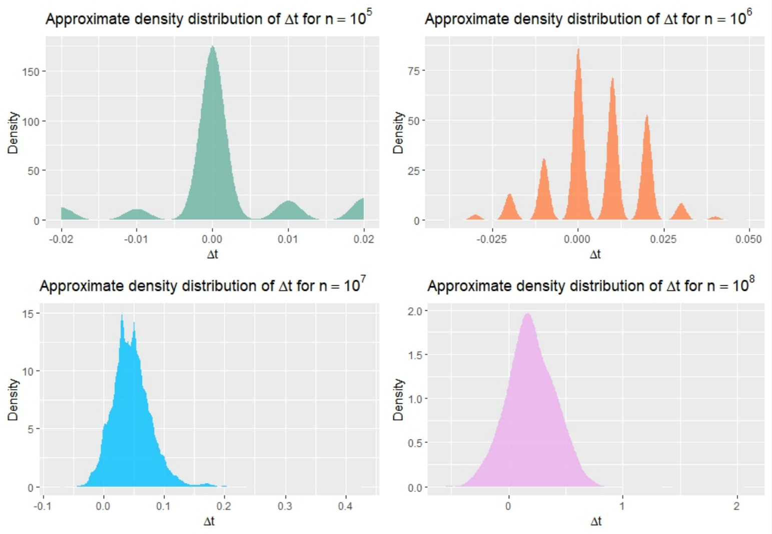
Define $\Delta t = T_{\text{randomized algorithm}} - T_{built-in}$.

Table: Statistics of Δt from samples

n	mean of Δt	median of Δt	mode of Δt	SD of Δt
50	0.000070	0.00	0.00	0.001896549
100	0.000115	0.00	0.00	0.002596179
10^3	0.000185	0.00	0.00	0.003524484
10^4	0.000270	0.00	0.00	0.003799252
10^5	0.001125	0.00	0.00	0.008355793
10^6	0.005370	0.01	0.00	0.012591361
10^7	0.046355	0.04	0.03	0.034781732
10^8	0.194850	0.19	0.18	0.217733280

Density of Δt

Figure: Approximate Density function for Δt



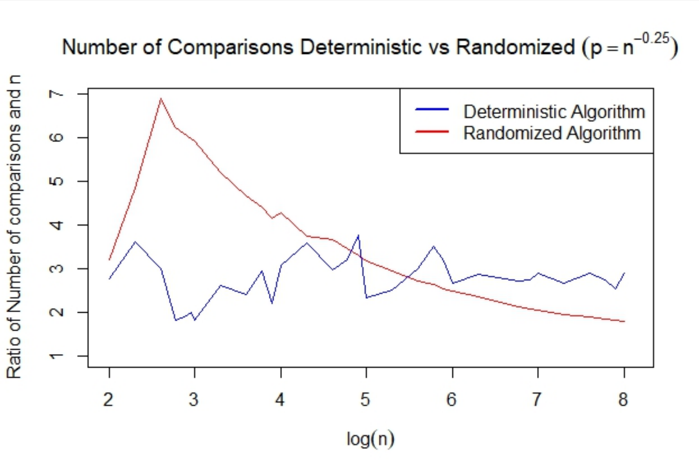

Number of Comparisons Deterministic vs Randomized (p = n^−0.25)
Deterministic Algorithm
Randomized Algorithm
Ratio of Number of comparisons and n
1
2
3
4
5
6
7
log(n)
2
3
4
5
6
7
8

Figure: $p = n^{-1/4}$

Number of Comparisons

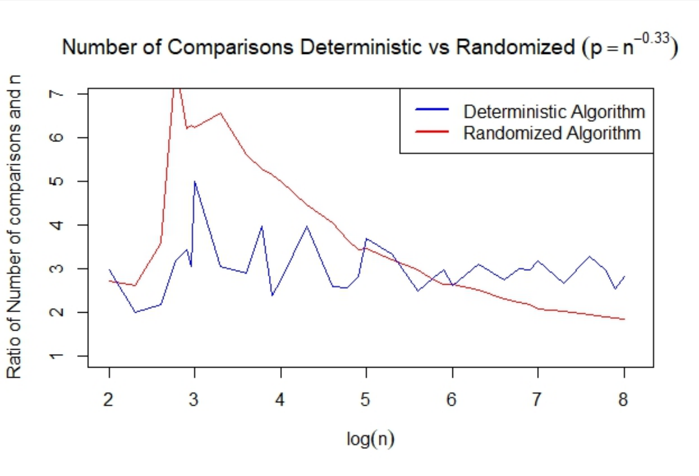


Figure: $p = n^{-1/3}$

n	$(p = n^{-0.25})$	$(p = n^{-0.25})$	Deterministic
100	322	273	277
200	971	524	725
400	2753	1438	1201
600	3737	4613	1090
800	4841	4963	1553
900	5368	5664	1798
1000	5927	6249	1825
10000	42877	49877	30830
100000	318251	345892	234259
1000000	2481772	2652757	2675274
10000000	20502494	20927133	28961496
100000000	179674957	185461673	289289681

Table: Number of Comparisons

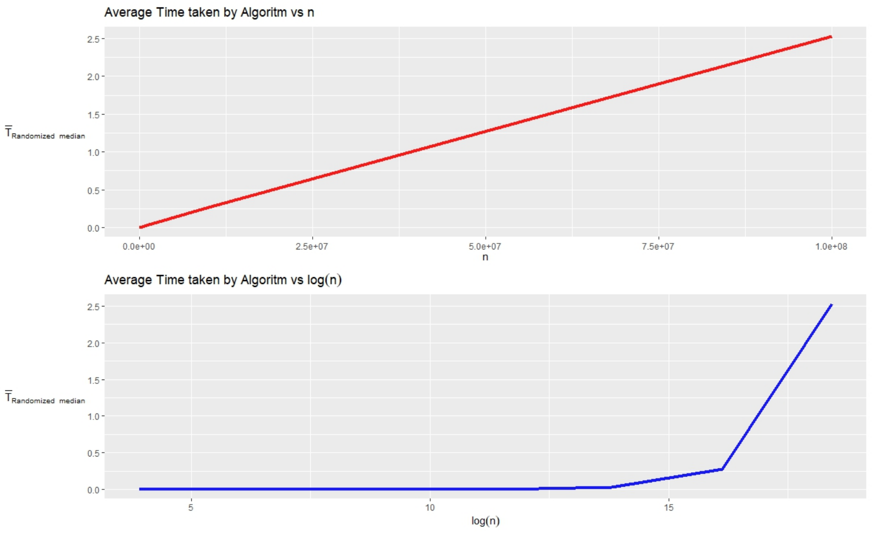


Figure: Average time elapsed vs input size

Conclusion

Conclusion

In conclusion

- This project has delved into the realm of randomized algorithms.
- Through rigorous analysis and virtual exploration, we have demonstrated the viability of leveraging random subsets to approximate the median of a given set efficiently.

References

- Lecture Slides CS648
- Crucial hints and multiple discussions from the Instructor ☺

☺ Thank You!

Questions?