Course MiniProject

Krish, Siddharth

IIT Kanpur

June 14, 2024

Outline

Outline

Problem

Given an array with n elements. Design a randomized Las Vegas algorithm that computes its median by performing only $1.5n$ comparisons with high probability.

Known Algorithms for median finding

Known Algorithms for median finding

- Deterministic Algorithm

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies 2.95n$ comparisons.
  - Any deterministic algorithm $\implies 2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect:

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies 2.95n$ comparisons.
  - Any deterministic algorithm $\implies 2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect: very simple recursive algorithm

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect: very simple recursive algorithm
    Can show: takes $< 3.5n$ comparisons on expectation.

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect: very simple recursive algorithm
    Can show: takes $< 3.5n$ comparisons on expectation.
- QuickSelect involves:

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect: very simple recursive algorithm
    Can show: takes $< 3.5n$ comparisons on expectation.
- QuickSelect involves:
  - Selecting pivot randomly

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect: very simple recursive algorithm
    Can show: takes $< 3.5n$ comparisons on expectation.
- QuickSelect involves:
  - Selecting pivot randomly
  - Partitioning array based on pivot.

Known Algorithms for median finding

- Deterministic Algorithm
  - Median of medians: $\implies \sim 20n$ comparisons
  - Best deterministic algorithm $\implies$ $2.95n$ comparisons.
  - Any deterministic algorithm $\implies$ $2n + o(n)$ comparisons in the worst case.
- Las Vegas Algorithm
  - QuickSelect: very simple recursive algorithm
    Can show: takes $< 3.5n$ comparisons on expectation.
- QuickSelect involves:
  - Selecting pivot randomly
  - Partitioning array based on pivot.
  - Recursive call on part containing median.

Inspiration from QuickSelect

Inspiration from QuickSelect

- Partition

Inspiration from QuickSelect

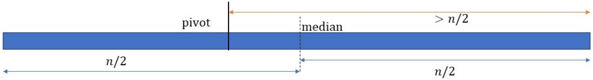
- Partition $\implies$ n comparisons in first recursive call.

Inspiration from QuickSelect

- Partition $\implies n$ comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.

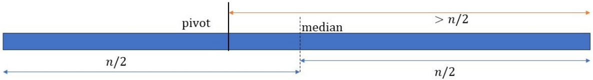
Inspiration from QuickSelect

- Partition $\implies n$ comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



Inspiration from QuickSelect

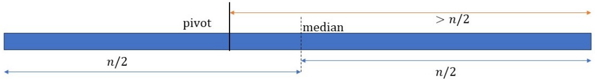
- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.

Inspiration from QuickSelect

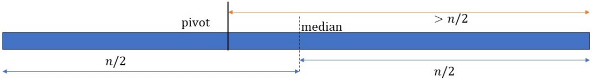
- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.

Inspiration from QuickSelect

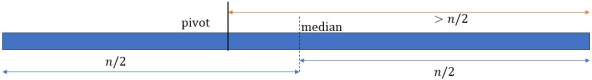
- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots.

Inspiration from QuickSelect

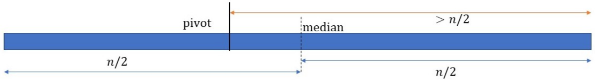
- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.

Inspiration from QuickSelect

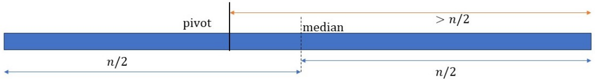
- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

Inspiration from QuickSelect

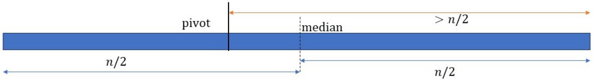
- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

Inspiration from QuickSelect

- Partition $\implies n$ comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.

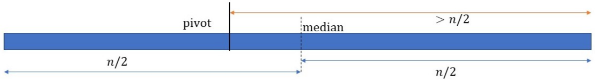


- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

  - Number of comparisons: $2n - x_1$

Inspiration from QuickSelect

- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.

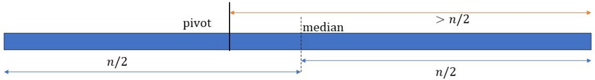


- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

  - Number of comparisons: $2n - x_1 \implies x_1 = n/2 - o(n)$

Inspiration from QuickSelect

- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.

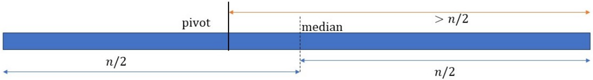


- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

  - Number of comparisons: $2n - x_1 \implies x_1 = n/2 - o(n)$
  - $1.5n + o(n)$ comparisons consumed

Inspiration from QuickSelect

- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.

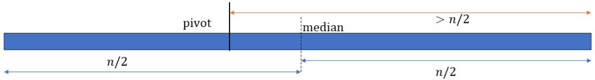


- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

  - Number of comparisons: $2n - x_1 \implies x_1 = n/2 - o(n)$
  - $1.5n + o(n)$ comparisons consumed $\implies x_2 - x_1 = o(n)$

Inspiration from QuickSelect

- Partition $\implies$ n comparisons in first recursive call.
- Left with $0.5n$ comparisons, but array of size $> n/2$.



- $\implies$ Should have an algorithm to find ith order statistic in $0.5n$ comparisons for size $> n/2$.
- Reason of failure: array size left after first recursive call.
- Solution: two pivots. Median lies between the pivots.
- Comparisons to be made with two pivots in partition.

  - Number of comparisons: $2n - x_1 \implies x_1 = n/2 - o(n)$
  - $1.5n + o(n)$ comparisons consumed $\implies x_2 - x_1 = o(n)$
- Pivots on either side of median with difference $o(n)$

How to select the pivots ?

How to select the pivots ?

- Cannot select pivots deterministically in $o(n)$ comparisons.

How to select the pivots ?

- Cannot select pivots deterministically in $o(n)$ comparisons.
  - Need to look (and compare) $\Theta(n)$ elements.

How to select the pivots ?

- Cannot select pivots deterministically in $o(n)$ comparisons.
  - Need to look (and compare) $\Theta(n)$ elements.
- Use the idea from randomized approximate median algorithm.

How to select the pivots ?

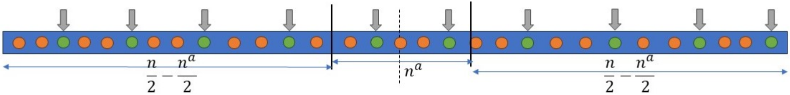
- Cannot select pivots deterministically in $o(n)$ comparisons.
  - Need to look (and compare) $\Theta(n)$ elements.
- Use the idea from randomized approximate median algorithm. But how?

How to select the pivots ?

- Cannot select pivots deterministically in $o(n)$ comparisons.
  - Need to look (and compare) $\Theta(n)$ elements.
- Use the idea from randomized approximate median algorithm. But how?
- Suppose we want the pivots at $\frac{n}{2} \pm \frac{n^a}{2}$ ranks, $a < 1$

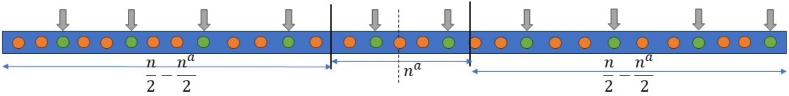
How to select the pivots ?

- Cannot select pivots deterministically in $o(n)$ comparisons.
  - Need to look (and compare) $\Theta(n)$ elements.
- Use the idea from randomized approximate median algorithm. But how?
- Suppose we want the pivots at $\frac{n}{2} \pm \frac{n^a}{2}$ ranks, $a < 1$



How to select the pivots ?

- Cannot select pivots deterministically in $o(n)$ comparisons.
  - Need to look (and compare) $\Theta(n)$ elements.
- Use the idea from randomized approximate median algorithm. But how?
- Suppose we want the pivots at $\frac{n}{2} \pm \frac{n^a}{2}$ ranks, $a < 1$



- Choose k elements **r.u.i** elements, sort them and select 2 elements equidistant from the middle element that approximate pivots.

Designing the algorithm

Designing the algorithm

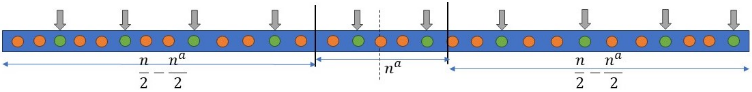
- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.

Designing the algorithm

- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.
- Need to find rank of pivots in B.

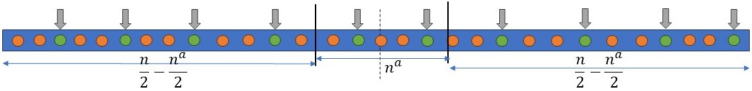
Designing the algorithm

- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.
- Need to find rank of pivots in B.



Designing the algorithm

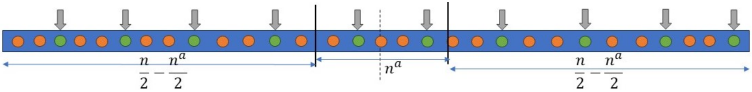
- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.
- Need to find rank of pivots in B.



- Expected number of elements in set B with $< \frac{n}{2} - \frac{n^a}{2}$ in $A = n^b \cdot Pr(\text{rank} < \frac{n}{2} - \frac{n^a}{2}) = n^b \cdot \frac{1}{n} \cdot (\frac{n}{2} - \frac{n^a}{2}) = \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$.

Designing the algorithm

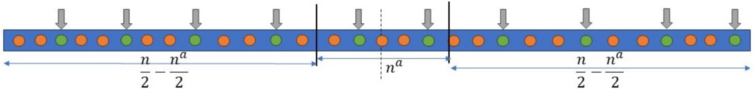
- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.
- Need to find rank of pivots in B.



- Expected number of elements in set B with $< \frac{n}{2} - \frac{n^a}{2}$ in $A = n^b \cdot Pr(\text{rank} < \frac{n}{2} - \frac{n^a}{2}) = n^b \cdot \frac{1}{n} \cdot (\frac{n}{2} - \frac{n^a}{2}) = \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$.
- Hence, we will select pivots to be $(\frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}})^{th}$ ranked elements in B.

Designing the algorithm

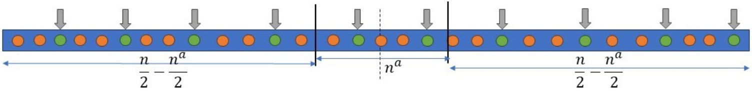
- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.
- Need to find rank of pivots in B.



- Expected number of elements in set B with $< \frac{n}{2} - \frac{n^a}{2}$ in $A = n^b \cdot Pr(\text{rank} < \frac{n}{2} - \frac{n^a}{2}) = n^b \cdot \frac{1}{n} \cdot (\frac{n}{2} - \frac{n^a}{2}) = \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$.
- Hence, we will select pivots to be $(\frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}})^{th}$ ranked elements in B.
- Partition function: will calculate ranks of both approximate pivots in A and give sub-array C of A between the pivots.

Designing the algorithm

- Since we need to select and sort k elements (name this set B), let $k = n^b, b < 1$.
- Need to find rank of pivots in B.



- Expected number of elements in set B with $< \frac{n}{2} - \frac{n^a}{2}$ in $A = n^b \cdot Pr(\text{rank} < \frac{n}{2} - \frac{n^a}{2}) = n^b \cdot \frac{1}{n} \cdot (\frac{n}{2} - \frac{n^a}{2}) = \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$.
- Hence, we will select pivots to be $(\frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}})^{th}$ ranked elements in B.
- Partition function: will calculate ranks of both approximate pivots in A and give sub-array C of A between the pivots.
- Sort C to find median: $C[n/2 - x_1 + 1]$

Outline

Final Algorithm

Final Algorithm

Input: Array A with n elements
Output: Median of A
$k \leftarrow n^b$;
$t \leftarrow \frac{k}{2n^{1-a}}$;
Select a multi set B of k elements from A r.u.i.;
Sort B;
$p_1 \leftarrow B[\frac{k}{2} - t]$;
$p_2 \leftarrow B[\frac{k}{2} + t]$;
$(A_{\text{new}}, x_1, x_2) \leftarrow \text{partition}(A, p_1, p_2)$; //$x_1$, x_2 ranks of pivots
$C \leftarrow A_{\text{new}}[x_1 : x_2]$;
Sort C;
if $x_1 \leq \frac{n}{2} \leq x_2$ **then**
    return $C[\frac{n}{2} - x_1 + 1]$;
else
    Median $\leftarrow$ Compute the exact median by $O(n)$ deterministic algorithm;
    return *Median*;
end

Analyzing the probability of failure

Analyzing the probability of failure

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability.

Analyzing the probability of failure

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability.
- Define Events
  - E_1 : Event that the median doesn't lie in array B.

Analyzing the probability of failure

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability.
- Define Events
  - E_1 : Event that the median doesn't lie in array B.
  - E_2 : Event that the size of B is too large, say, $\text{size}(B) \geq 2n^a$.

Analyzing the probability of failure

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability.
- Define Events
  - E_1 : Event that the median doesn't lie in array B.
  - E_2 : Event that the size of B is too large, say, $\text{size}(B) \geq 2n^a$.
- Will show, if neither E_1 nor E_2 occurs $\implies 1.5n + o(n)$ comparisons.

Analyzing the probability of failure

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability.
- Define Events
  - E_1 : Event that the median doesn't lie in array B.
  - E_2 : Event that the size of B is too large, say, $\text{size}(B) \geq 2n^a$.
- Will show, if neither E_1 nor E_2 occurs $\implies 1.5n + o(n)$ comparisons. First need to show bound on $E = E_1 \cup E_2$

Analyzing the probability of failure

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability.
- Define Events
  - E_1 : Event that the median doesn't lie in array B.
  - E_2 : Event that the size of B is too large, say, $\text{size}(B) \geq 2n^a$.
- Will show, if neither E_1 nor E_2 occurs $\implies 1.5n + o(n)$ comparisons. First need to show bound on $E = E_1 \cup E_2$
- By Union Theorem,

$$P(E_1 \cup E_2) \leq P(E_1) + P(E_2)$$

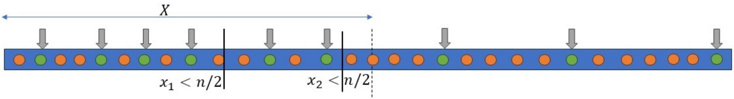
Estimating an upper bound on $P(E_1)$

Estimating an upper bound on $P(E_1)$

Define RV X : Number of elements from B with rank $< \frac{n}{2}$ in A.

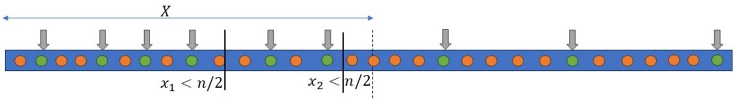
Estimating an upper bound on $P(E_1)$

Define RV X : Number of elements from B with rank $< \frac{n}{2}$ in A.



Estimating an upper bound on $P(E_1)$

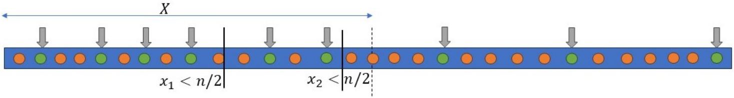
Define RV X : Number of elements from B with rank $< \frac{n}{2}$ in A.



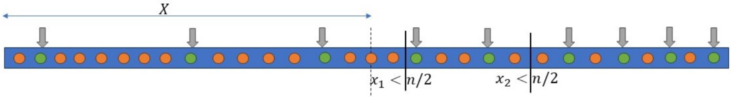
Occurs when: $X > \text{rank}(p_2)$ in B.

Estimating an upper bound on $P(E_1)$

Define RV X : Number of elements from B with rank $< \frac{n}{2}$ in A.

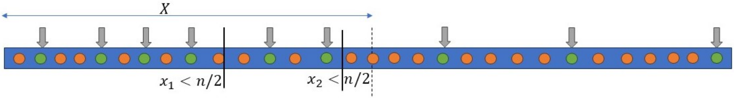


Occurs when: $X > \text{rank}(p_2)$ in B.

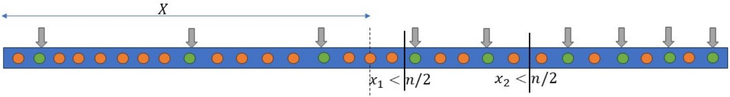


Estimating an upper bound on $P(E_1)$

Define RV X : Number of elements from B with rank $< \frac{n}{2}$ in A.



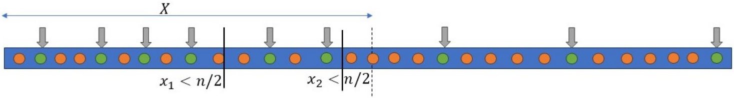
Occurs when: $X > \text{rank}(p_2)$ in B.



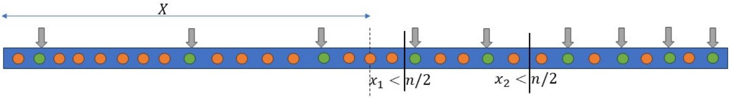
Occurs when: $X < \text{rank}(p_1)$ in B.

Estimating an upper bound on $P(E_1)$

Define RV X : Number of elements from B with rank $< \frac{n}{2}$ in A.



Occurs when: $X > \text{rank}(p_2)$ in B.



Occurs when: $X < \text{rank}(p_1)$ in B.

$$\implies E_1 = \left(X \geq \frac{n^b}{2} + \frac{n^b}{2n^{1-a}}\right) \cup \left(X \leq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}\right).$$

Clearly, $E[X] = \frac{n^b}{2} \implies E_1 = |X - E[X]| \geq d, d = \frac{n^b}{2n^{1-a}}$

Estimating an upper bound on $P(E_1)$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank } < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies X$ is a Bernoulli RV.

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies X$ is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies$ X is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies$ X is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

$$P(E_1) = P\left(|X - E[X]| \geq d\right) \leq 2e^{-\left(\frac{d}{E[X]}\right)^2 \frac{E[X]}{3}}.$$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies X$ is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

$$P(E_1) = P\left(|X - E[X]| \geq d\right) \leq 2e^{-\left(\frac{d}{E[X]}\right)^2 \frac{E[X]}{3}}.$$

$d = \frac{n^b}{2n^{1-a}}$, $E[X] = \frac{n^b}{2}$,

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies X$ is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

$$P(E_1) = P\left(|X - E[X]| \geq d\right) \leq 2e^{-\left(\frac{d}{E[X]}\right)^2 \frac{E[X]}{3}}.$$

$d = \frac{n^b}{2n^{1-a}}$, $E[X] = \frac{n^b}{2}$, Hence, $\frac{d}{E[X]} = \frac{1}{n^{1-a}} = n^{a-1}$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies$ X is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies$ X is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

$$P(E_1) = P\left(|X - E[X]| \geq d\right) \leq 2e^{-\left(\frac{d}{E[X]}\right)^2 \frac{E[X]}{3}}.$$

$d = \frac{n^b}{2n^{1-a}}$, $E[X] = \frac{n^b}{2}$, Hence, $\frac{d}{E[X]} = \frac{1}{n^{1-a}} = n^{a-1}$

$$\implies P(E_1) \leq 2e^{-n^{2a-2} \times \frac{n^b}{6}} = 2e^{-\frac{n^{2a+b-2}}{6}}$$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies X$ is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

$$P(E_1) = P\left(|X - E[X]| \geq d\right) \leq 2e^{-\left(\frac{d}{E[X]}\right)^2 \frac{E[X]}{3}}.$$

$d = \frac{n^b}{2n^{1-a}}$, $E[X] = \frac{n^b}{2}$, Hence, $\frac{d}{E[X]} = \frac{1}{n^{1-a}} = n^{a-1}$

$$\implies P(E_1) \leq 2e^{-n^{2a-2} \times \frac{n^b}{6}} = 2e^{-\frac{n^{2a+b-2}}{6}} = 2e^{-g(n)} \text{ (say)}$$

Estimating an upper bound on $P(E_1)$

$$X_i = \begin{cases} 1 & \text{if the } i\text{th element in } B \text{ has a rank} < \frac{n}{2} \text{ in } A \\ 0 & \text{otherwise} \end{cases}$$

$X = \sum X_i$ and X_i's are independent random variables
$\implies X$ is sum of k independent binomial RV's with $p = \frac{1}{2}$
$\implies X$ is a Bernoulli RV.
We can use Chernoff's bound to find $P(|X - E[X]| \geq d)$

$$P(E_1) = P\left(|X - E[X]| \geq d\right) \leq 2e^{-\left(\frac{d}{E[X]}\right)^2 \frac{E[X]}{3}}.$$

$d = \frac{n^b}{2n^{1-a}}$, $E[X] = \frac{n^b}{2}$, Hence, $\frac{d}{E[X]} = \frac{1}{n^{1-a}} = n^{a-1}$

$$\implies P(E_1) \leq 2e^{-n^{2a-2} \times \frac{n^b}{6}} = 2e^{-\frac{n^{2a+b-2}}{6}} = 2e^{-g(n)} \text{ (say)}$$

Will select a and b s.t. $2a + b > 2$.

Estimating an upper bound on $P(E_2)$

Estimating an upper bound on $P(E_2)$

- Define events ($X_1(X_2) =$ rank of first (second) pivot in A)
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$

Estimating an upper bound on $P(E_2)$

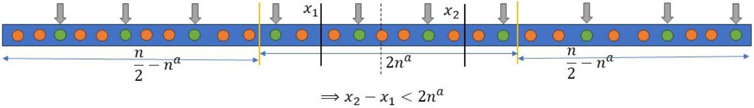
- Define events ($X_1(X_2)$ = rank of first (second) pivot in A)
  - $E_{2,1} : X_1 \le \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \ge \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen.

Estimating an upper bound on $P(E_2)$

- Define events ($X_1(X_2)$ = rank of first (second) pivot in A)
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen. Reason:

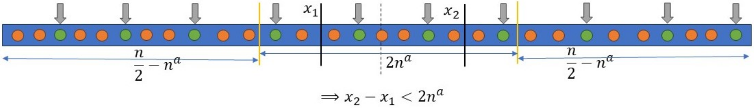
Estimating an upper bound on $P(E_2)$

- Define events $(X_1(X_2)$ = rank of first (second) pivot in $A)$
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen. Reason:



Estimating an upper bound on $P(E_2)$

- Define events $(X_1(X_2)$ = rank of first (second) pivot in $A)$
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen. Reason:



$$\implies E_2 \subseteq E_{2,1} \cup E_{2,2} \implies P(E_2) \leq P(E_{2,1} \cup E_{2,2})$$
$$\implies P(E_2) \leq P(E_{2,1}) + P(E_{2,2}) \text{ by Union theorem.}$$

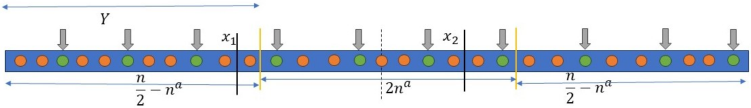
Estimating an upper bound on $P(E_{2,1})$

Estimating an upper bound on $P(E_{2,1})$

- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.

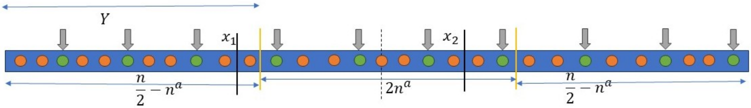
Estimating an upper bound on $P(E_{2,1})$

- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



Estimating an upper bound on $P(E_{2,1})$

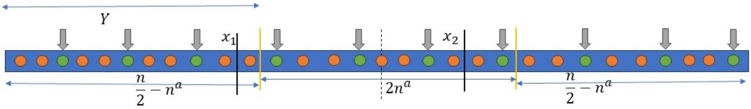
- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in B

Estimating an upper bound on $P(E_{2,1})$

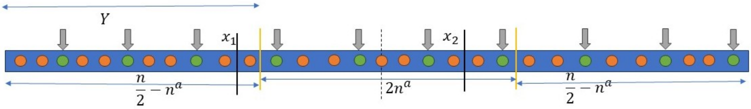
- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in $B$$= Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$

Estimating an upper bound on $P(E_{2,1})$

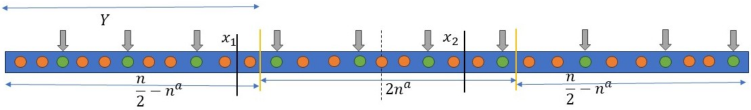
- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in $B = Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Y is a Bernoulli RV with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$

Estimating an upper bound on $P(E_{2,1})$

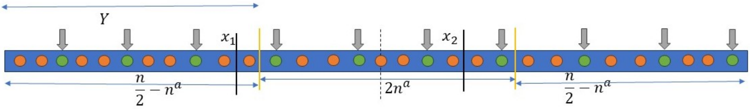
- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in B= $Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Y is a Bernoulli RV with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- Therefore, using chernoff bound :

Estimating an upper bound on $P(E_{2,1})$

- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



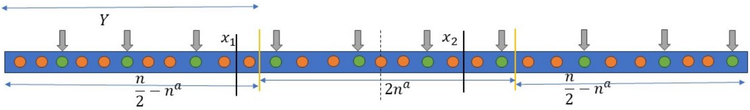
- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in $B$$= Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Y is a Bernoulli RV with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- Therefore, using chernoff bound :

$$P\left[Y \geq (1+\delta)E[Y]\right] \leq e^{-\frac{\delta^2 E[Y]}{3}}$$

Substituting $(1+\delta)E[Y]$ as $\frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$

Estimating an upper bound on $P(E_{2,1})$

- Y : RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A.



- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in B= $Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Y is a Bernoulli RV with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- Therefore, using chernoff bound :

$$P\left[Y \geq (1+\delta)E[Y]\right] \leq e^{-\frac{\delta^2 E[Y]}{3}}$$

Substituting $(1+\delta)E[Y]$ as $\frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$

$$\delta = \frac{1}{n^{1-a} - 2} \geq \frac{1}{n^{1-a}}$$

Estimating the final bound on E

Estimating the final bound on E

$$\implies \delta E[Y] = \frac{n^b}{2n^{1-a}}$$

Substituting this and lower bound of δ in $\delta\frac{E[Y]\delta}{3}$, we get

$$P\left[E_{2,1}\right] \leq e^{-\left(\frac{1}{n^{1-a}}\right)\frac{n^b}{6n^{1-a}}} = e^{-\frac{n^{2a+b-2}}{6}} = e^{-g(n)}$$

Estimating the final bound on E

$$\implies \delta E[Y] = \frac{n^b}{2n^{1-a}}$$

Substituting this and lower bound of δ in $\delta\frac{E[Y]\delta}{3}$, we get

$$P\left[E_{2,1}\right] \leq e^{-\left(\frac{1}{n^{1-a}}\right)\frac{n^b}{6n^{1-a}}} = e^{-\frac{n^{2a+b-2}}{6}} = e^{-g(n)}$$

By symmetry, $P(E_{2,2}) \leq e^{-g(n)}$

$$P(E_2) \leq P(E_{2,1}) + P(E_{2,2}) \leq 2e^{-g(n)}$$

Estimating the final bound on E

$$\implies \delta E[Y] = \frac{n^b}{2n^{1-a}}$$

Substituting this and lower bound of δ in $\delta\frac{E[Y]\delta}{3}$, we get

$$P\left[E_{2,1}\right] \leq e^{-\left(\frac{1}{n^{1-a}}\right)\frac{n^b}{6n^{1-a}}} = e^{-\frac{n^{2a+b-2}}{6}} = e^{-g(n)}$$

By symmetry, $P(E_{2,2}) \leq e^{-g(n)}$

$$P(E_2) \leq P(E_{2,1}) + P(E_{2,2}) \leq 2e^{-g(n)}$$

$$\implies \boxed{P(E) \leq P(E_1) + P(E_2) \leq 4e^{-g(n)}}$$

Results

Results

- With probability more than $1 - 4e^{-g(n)}$,

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}}$$

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}} + \underbrace{n^b \log n^b}_{\text{Sorting multiset B}}$$

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}} + \underbrace{n^b \log n^b}_{\text{Sorting multiset B}} + \underbrace{2n^a \log 2n^a}_{\text{Sorting sub-array C}}$$

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}} + \underbrace{n^b \log n^b}_{\text{Sorting multiset B}} + \underbrace{2n^a \log 2n^a}_{\text{Sorting sub-array C}}$$

$$= 1.5n + o(n)$$

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}} + \underbrace{n^b \log n^b}_{\text{Sorting multiset B}} + \underbrace{2n^a \log 2n^a}_{\text{Sorting sub-array C}}$$

$$= 1.5n + o(n)$$

- With probability $< 4e^{-g(n)}$,

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}} + \underbrace{n^b \log n^b}_{\text{Sorting multiset B}} + \underbrace{2n^a \log 2n^a}_{\text{Sorting sub-array C}}$$

$$= 1.5n + o(n)$$

- With probability $< 4e^{-g(n)}$, the number of comparisons performed by the algorithm are cn

Results

- With probability more than $1 - 4e^{-g(n)}$, the number of comparisons performed by the algorithm are

$$\leq \underbrace{1 \times (\frac{n}{2} - n^a) + 2 \times (\frac{n}{2} + n^a)}_{\text{Partition}} + \underbrace{n^b \log n^b}_{\text{Sorting multiset B}} + \underbrace{2n^a \log 2n^a}_{\text{Sorting sub-array C}}$$

$$= 1.5n + o(n)$$

- With probability $< 4e^{-g(n)}$, the number of comparisons performed by the algorithm are cn for some c, where $g(n) = \frac{n^{2a+b-2}}{6}$.