Course MiniProject

Krish, Siddharth

IIT Kanpur

March 31, 2024

Outline

Outline

1. Introduction
2. Algorithm and Proofs

Problem and Inspiration

Problem and Inspiration

- To find median of an array

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array
  - Partitioning array based on pivot

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array
  - Partitioning array based on pivot
  - Recursive call on part containing the median

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array
  - Partitioning array based on pivot
  - Recursive call on part containing the median
- Partition function: compares pivot with every element

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array
  - Partitioning array based on pivot
  - Recursive call on part containing the median
- Partition function: compares pivot with every element
  - Places pivot at its correct position

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array
  - Partitioning array based on pivot
  - Recursive call on part containing the median
- Partition function: compares pivot with every element
  - Places pivot at its correct position
  - Ensures smaller elements to left, greater to right of pivot

Problem and Inspiration

- To find median of an array
- Should takes $1.5n$ comparisons with high probability
- Our algorithm draws inspiration from QuickSelect
- QuickSelect: simple Las Vegas recursive algorithm
  - Selecting pivot randomly from the array
  - Partitioning array based on pivot
  - Recursive call on part containing the median
- Partition function: compares pivot with every element
  - Places pivot at its correct position
  - Ensures smaller elements to left, greater to right of pivot
- Let us improve partition function for our problem

Improving Partition function

Improving Partition function

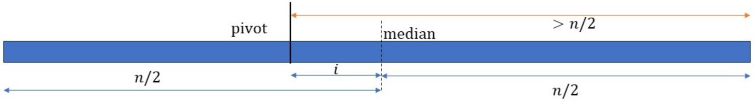
- Takes n comparisons in first recursive call

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

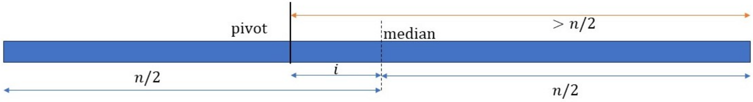
Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



Improving Partition function

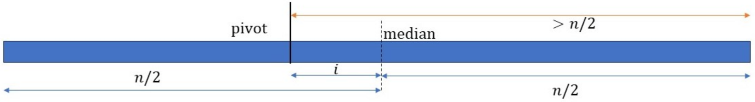
- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



- Have to find ith element in $0.5n$ comparisons for size $> n/2$

Improving Partition function

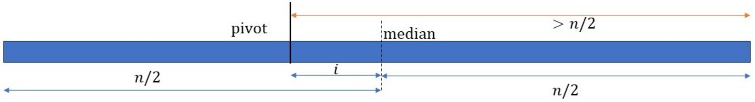
- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call

Improving Partition function

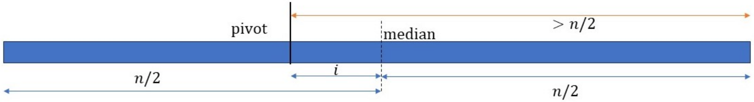
- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots.

Improving Partition function

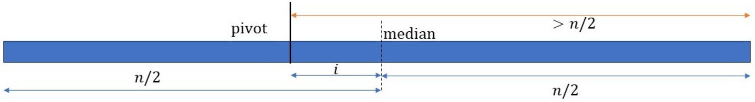
- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



- Have to find ith element 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

Improving Partition function

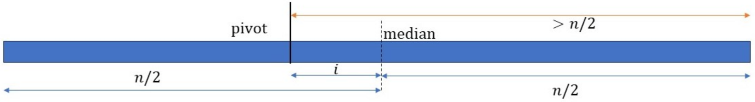
- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



- Have to find ith element 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

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

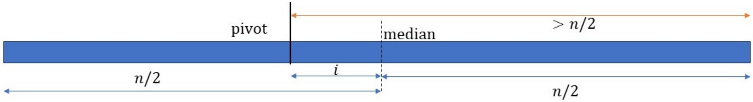


- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots. Median lies between the pivots

- Partition will give subarray of elements between pivots

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

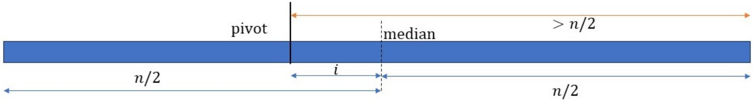


- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots. Median lies between the pivots

- Partition will give subarray of elements between pivots
- Comparisons to be made with two pivots in partition

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

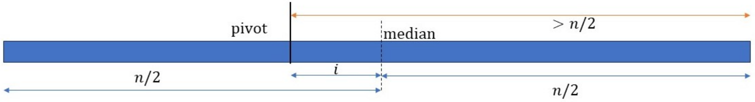


- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots. Median lies between the pivots

- Partition will give subarray of elements between pivots
- Comparisons to be made with two pivots in partition
  - Number of comparisons: $2n - x_1$

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

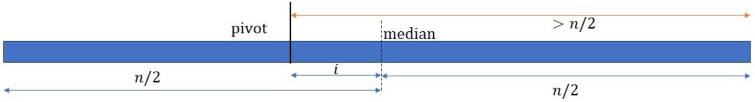


- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots. Median lies between the pivots

- Partition will give subarray of elements between pivots
- Comparisons to be made with two pivots in partition
  - Number of comparisons: $2n - x_1 \implies x_1 = n/2 - o(n)$

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

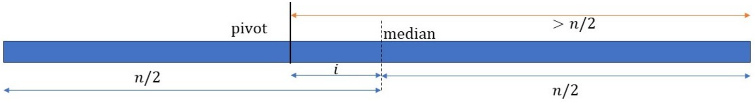


- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots. Median lies between the pivots

- Partition will give subarray of elements between 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

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$

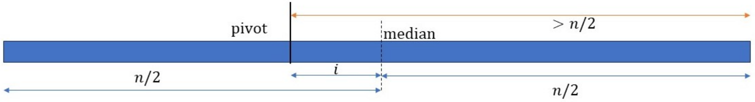


- Have to find ith element in $0.5n$ comparisons for size $> n/2$
- Reason of failure: array size left after first recursive call
- Solution: two pivots. Median lies between the pivots

- Partition will give subarray of elements between 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 $\implies x_2 - x_1 = o(n)$

Improving Partition function

- Takes n comparisons in first recursive call
- Left with $0.5n$ comparisons, but array of size $> n/2$



- Have to find ith element 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

- Partition will give subarray of elements between 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 ?

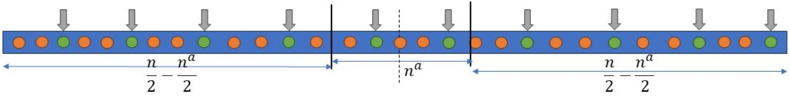
- Cannot select pivots deterministically in $o(n)$ comparisons
  - Need to look (and compare) $\Theta(n)$ elements
- Use the idea from randomized approx median algorithm

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 approx median algorithm
- Suppose we want the pivots at $\frac{n}{2} \pm \frac{n^a}{2}$ ranks, $a < 1$

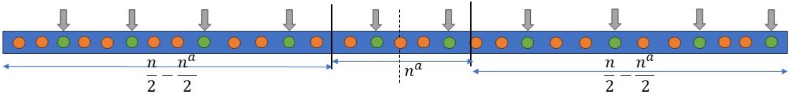
How to select the pivots ?

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



How to select the pivots ?

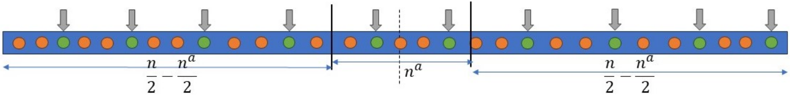
- Cannot select pivots deterministically in $o(n)$ comparisons
  - Need to look (and compare) $\Theta(n)$ elements
- Use the idea from randomized approx median algorithm
- Suppose we want the pivots at $\frac{n}{2} \pm \frac{n^a}{2}$ ranks, $a < 1$



- Choose set B of k elements **r.u.i** from A, sort them and select elements at ranks in B :

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 approx median algorithm
- Suppose we want the pivots at $\frac{n}{2} \pm \frac{n^a}{2}$ ranks, $a < 1$



- Choose set B of k elements **r.u.i** from A, sort them and select elements at ranks in B :

$$E\left[\text{elements from } B \text{ with rank} < \left(\frac{n}{2} \pm \frac{n^a}{2}\right) \text{ in } A\right]$$

Designing the algorithm

Designing the algorithm

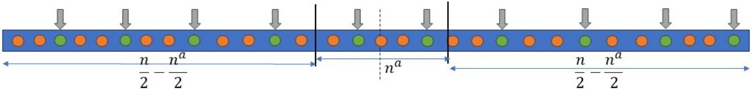
- Since we need to sort B, let $k = n^b, b < 1$

Designing the algorithm

- Since we need to sort B, let $k = n^b, b < 1$
- Need to find rank of pivots in B

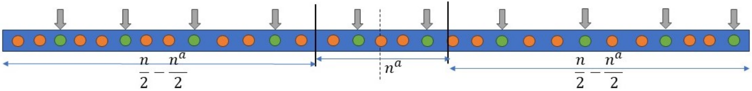
Designing the algorithm

- Since we need to sort B, let $k = n^b, b < 1$
- Need to find rank of pivots in B



Designing the algorithm

- Since we need to sort B, let $k = n^b, b < 1$
- Need to find rank of pivots in B

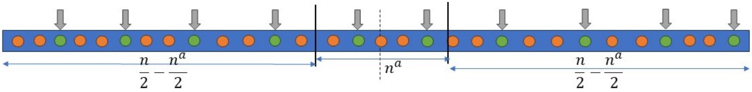


- Expected number of elements in B with $< \frac{n}{2} \pm \frac{n^a}{2}$ in $A =$

$$n^b \cdot P\left(\text{rank} < \frac{n}{2} \pm \frac{n^a}{2}\right) = n^b \cdot \frac{1}{n} \cdot \left(\frac{n}{2} \pm \frac{n^a}{2}\right) = \frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}}$$

Designing the algorithm

- Since we need to sort B, let $k = n^b, b < 1$
- Need to find rank of pivots in B



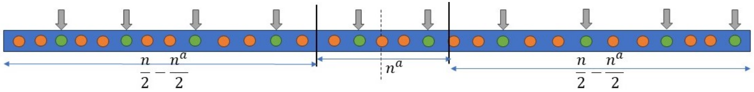
- Expected number of elements in B with $< \frac{n}{2} \pm \frac{n^a}{2}$ in $A =$

$$n^b \cdot P\left(\text{rank} < \frac{n}{2} \pm \frac{n^a}{2}\right) = n^b \cdot \frac{1}{n} \cdot \left(\frac{n}{2} \pm \frac{n^a}{2}\right) = \frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}}$$

- Hence, pivots are at ranks $(\frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}})^{th}$ in B

Designing the algorithm

- Since we need to sort B, let $k = n^b, b < 1$
- Need to find rank of pivots in B



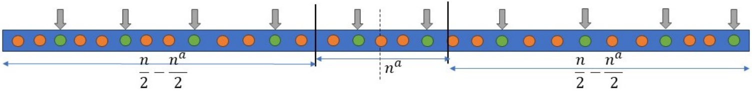
- Expected number of elements in B with $< \frac{n}{2} \pm \frac{n^a}{2}$ in $A =$

$$n^b \cdot P\left(\text{rank} < \frac{n}{2} \pm \frac{n^a}{2}\right) = n^b \cdot \frac{1}{n} \cdot \left(\frac{n}{2} \pm \frac{n^a}{2}\right) = \frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}}$$

- Hence, pivots are at ranks $(\frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}})^{th}$ 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 sort B, let $k = n^b, b < 1$
- Need to find rank of pivots in B



- Expected number of elements in B with $< \frac{n}{2} \pm \frac{n^a}{2}$ in $A =$

$$n^b \cdot P\left(\text{rank} < \frac{n}{2} \pm \frac{n^a}{2}\right) = n^b \cdot \frac{1}{n} \cdot \left(\frac{n}{2} \pm \frac{n^a}{2}\right) = \frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}}$$

- Hence, pivots are at ranks $(\frac{n^b}{2} \pm \frac{n^b}{2n^{1-a}})^{th}$ 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: $C[\frac{n}{2} - x_1 + 1]$

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;
$p_1 \leftarrow B[\frac{k}{2} - t]$;
$p_2 \leftarrow B[\frac{k}{2} + t]$;
$(A_{\text{new}}, x_1, x_2) \leftarrow$ 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

Upper bound on failure probability

Upper bound on failure probability

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

Upper bound on failure probability

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

Upper bound on failure probability

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

Upper bound on failure probability

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

Upper bound on failure probability

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

Upper bound on failure probability

- To show: algorithm performs $1.5n$ expected number of comparisons with high probability
- Define Failure Events
  - E_1 : Event that the median doesn't lie in subarray C
  - E_2 : Event that the size of C is too large, say, size$(C) \geq 2n^a$
- Will show, neither E_1 nor $E_2 \implies 1.5n + o(n)$ comparisons
- First will show bound on $E = E_1 \cup E_2$

Upper bound on failure probability

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

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

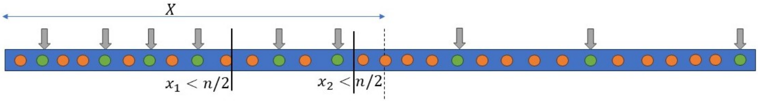
Estimating an upper bound on $P(E_1)$

Estimating an upper bound on $P(E_1)$

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

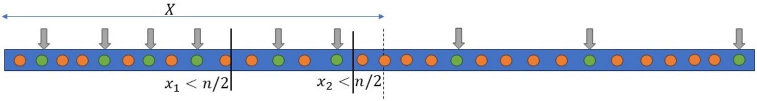
Estimating an upper bound on $P(E_1)$

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



Estimating an upper bound on $P(E_1)$

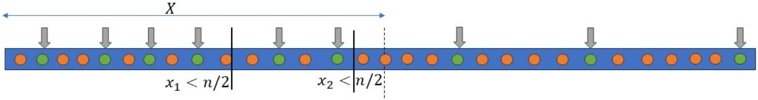
- Define RV X: Number of elements in B with rank $< \frac{n}{2}$ in A



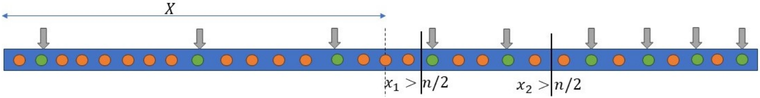
- Occurs when: $X > \text{rank}(p_2)$ in B

Estimating an upper bound on $P(E_1)$

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

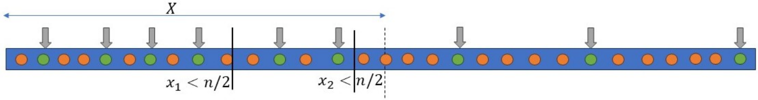


- Occurs when: $X >$ rank(p_2) in B

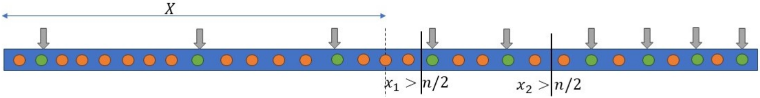


Estimating an upper bound on $P(E_1)$

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



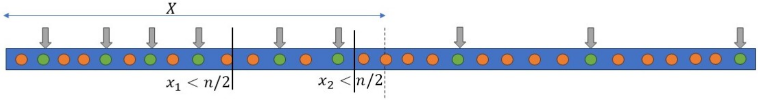
- Occurs when: $X > \text{rank}(p_2)$ in B



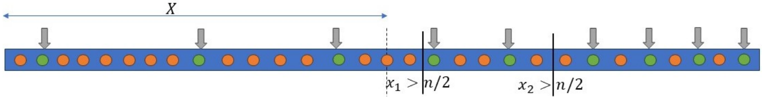
- Occurs when: $X < \text{rank}(p_1)$ in B

Estimating an upper bound on $P(E_1)$

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



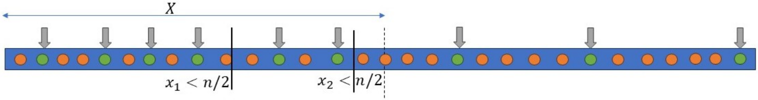
- Occurs when: $X >$ rank(p_2) in B



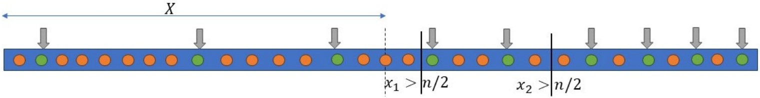
- Occurs when: $X <$ rank(p_1) in B
- $E_1 = \left(X \geq \frac{n^b}{2} + \frac{n^b}{2n^{1-a}}\right) \cup \left(X \leq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}\right)$

Estimating an upper bound on $P(E_1)$

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



- Occurs when: $X >$ rank(p_2) in B



- Occurs when: $X <$ rank(p_1) in B
- $E_1 = \left(X \geq \frac{n^b}{2} + \frac{n^b}{2n^{1-a}}\right) \cup \left(X \leq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}\right)$
- $E_1 = \left|X - \frac{n^b}{2}\right| \geq d, d = \frac{n^b}{2n^{1-a}}$

Estimating an upper bound on $P(E_1)$

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise.

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$
- Can use Chernoff's bound to find $P\left(|X - E[X]| \geq d\right)$

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$
- Can use Chernoff's bound to find $P\left(|X - E[X]| \geq d\right)$

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

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$
- Can use Chernoff's bound to find $P\left(|X - E[X]| \geq d\right)$

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

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

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$
- Can use Chernoff's bound to find $P\left(|X - E[X]| \geq d\right)$

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

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

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

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$
- Can use Chernoff's bound to find $P\left(|X - E[X]| \geq d\right)$

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

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

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

Estimating an upper bound on $P(E_1)$

- $X = \sum_{i=1}^{k} X_i$ where $X_i = 1$ if i^{th} element in B has rank $< \frac{n}{2}$ in A, 0 otherwise. $P\left(\text{rank} < \frac{n}{2}\right) = \frac{1}{2}$
- X is a Bernoulli RV with $p = \frac{1}{2}$ and $E[X] = \frac{n^b}{2}$
- Can use Chernoff's bound to find $P\left(|X - E[X]| \geq d\right)$

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

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

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

- 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 $P(E_2)$

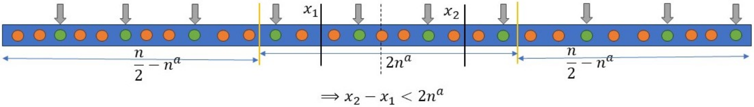
- Define events ($X_1(X_2)$ = rank of first (second) pivot in A)
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen.

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:

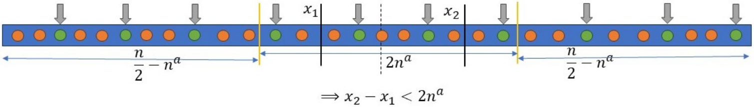
Estimating an upper bound on $P(E_2)$

- 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 $P(E_2)$

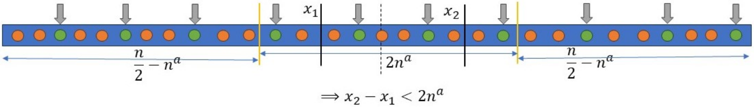
- Define events ($X_1(X_2)$ = rank of first (second) pivot in A)
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen. Reason:



- $E_2 \subseteq E_{2,1} \cup E_{2,2} \implies P(E_2) = P(E_{2,1} \cup E_{2,2})$

Estimating an upper bound on $P(E_2)$

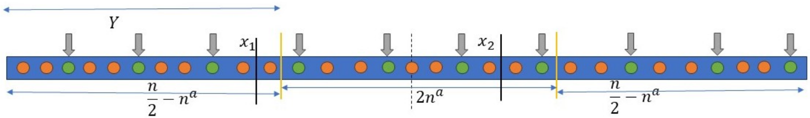
- Define events ($X_1(X_2)$ = rank of first (second) pivot in A)
  - $E_{2,1} : X_1 \leq \frac{n}{2} - n^a$
  - $E_{2,2} : X_2 \geq \frac{n}{2} + n^a$
- Observation : If event E_2 happens, then at least one of $E_{2,1}$ or $E_{2,2}$ must also happen. Reason:



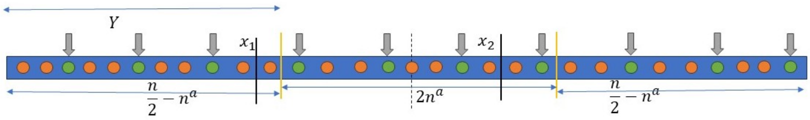
- $E_2 \subseteq E_{2,1} \cup E_{2,2} \implies P(E_2) = P(E_{2,1} \cup E_{2,2})$
- $P(E_2) \leq P(E_{2,1}) + P(E_{2,2})$ by Union theorem

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

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

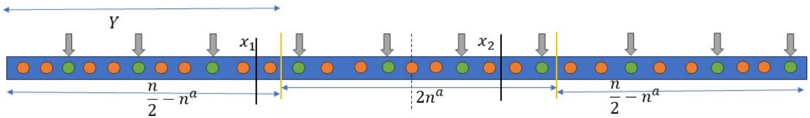


Estimating an upper bound on $P(E_{2,1})$



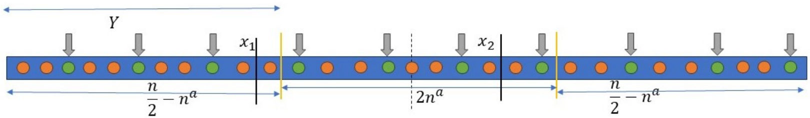
- Y : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$

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



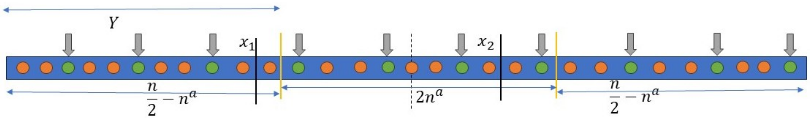
- Y : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in B

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



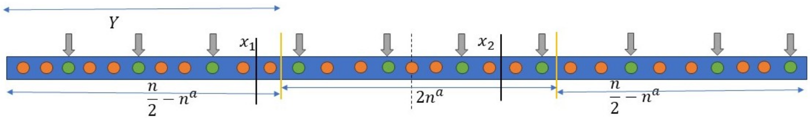
- Y : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in $B = Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$

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



- Y : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- 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}}$
- Therefore, using Chernoff's bound :

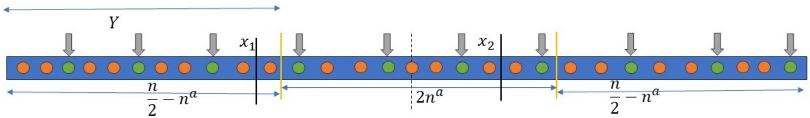
Estimating an upper bound on $P(E_{2,1})$



- Y : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in $B= Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Therefore, using Chernoff's bound :

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

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

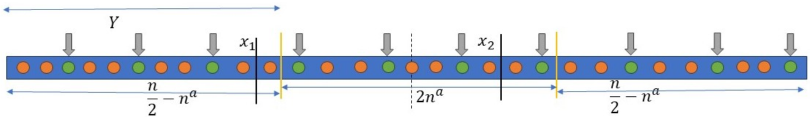


- Y : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in $B = Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Therefore, using Chernoff's 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 : Bernoulli RV equal to number of elements in B with rank $\leq \frac{n}{2} - n^a$ in A with $p = \frac{1}{2} - \frac{1}{n^{1-a}}, E[Y] = \frac{n^b}{2} - \frac{n^b}{n^{1-a}}$
- As before, $E_{2,1} = Y \geq \text{rank}(p_1)$ in B= $Y \geq \frac{n^b}{2} - \frac{n^b}{2n^{1-a}}$
- Therefore, using Chernoff's 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

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

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

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

$$P\left[E_{2,1}\right] \leq e^{-g(n)}$$

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

$$P\left[E_{2,1}\right] \leq e^{-g(n)}$$

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

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

$$P\left[E_{2,1}\right] \leq e^{-g(n)}$$

- 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

$$P\left[E_{2,1}\right] \leq e^{-g(n)}$$

- 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 for some c

Results

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

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

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

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