Learning Goals

- (Review/Learn) QuickSort ✓
- Benchmark Worst/Best QuickSort runtimes ✓
- Define + Describe: Sample Space, Random Variable, Expectation value, linearity of expectation
- Describe processes (basic/clever) for calculating average runtime.
- Analyze X_{ij} and calculate average runtime of QuickSort ✓
- Describe pros/cons of different sorting algs.

Announcements

- What I Did This Summer go/WIDTS 11/4, 11/11 12:30-1:30
- OH today 3:30-4:30, No OH tomorrow, Sign-up @ door
- go/SK_F24_checkin

Exit Tickets

Harmonic Series

$O(n)$

What do we know about Runtime of QuickSort

✗ A) The runtime is less than $O(n \log n)$ on average

B) The runtime is $O(n \log n)$ half the time

✓ C) The average runtime is at most $O(n \log n)$

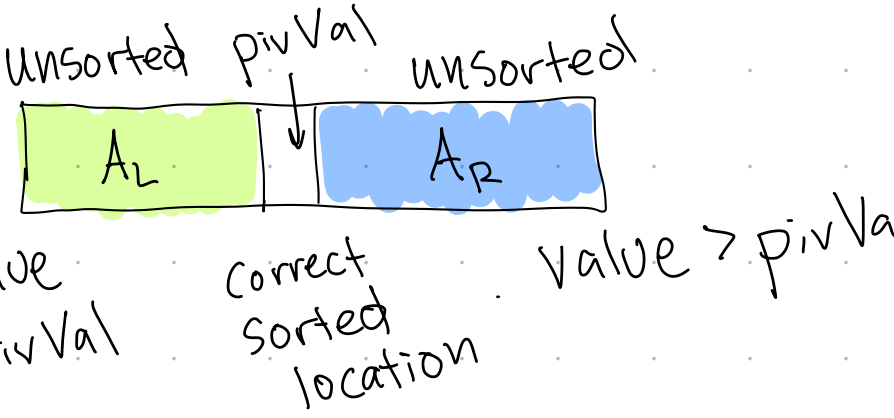
✓ D) The average runtime is $\Theta(n \log n)$

QuickSort

Input: Array A of unique integers
Output: Sorted A

- If $|A| = 1$: Return A
- pivInd ← randomly chosen index, with value pivVal } Preprocessing
- Partition (A, pivInd) } Dividing
- QuickSort (A_L) } Conquering
- QuickSort (A_R)

After Partition:

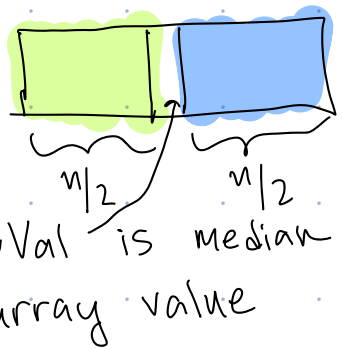


Key Pts

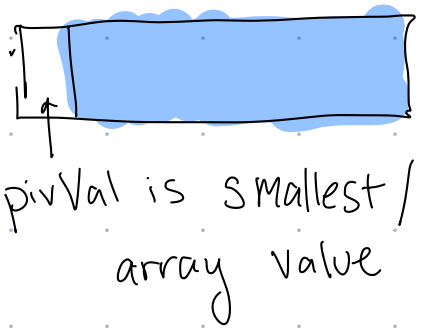
- Partition takes $O(|A|)$ time
- If pivVal is z_i (i^{th} smallest element of A) after Partition, pivVal is at position i.

Lucky vs. Unlucky Pivot Choices

Lucky:



Unlucky:



1. Suppose you get lucky at every recursive call of QuickSort.
2. Suppose you get unlucky at every recursive call of QuickSort.

- Create recurrence relation for runtime of QuickSort in each case
- Solve recurrence to determine runtime in each case

3. What is Sample Space? Random Variable? Expectation value? Linearity of Expectation?

Lucky vs. Unlucky Pivot Choices

1. Suppose you get lucky at every recursive call of QuickSort.

$$T(n) = \begin{cases} O(1) & \text{if } n \le 1 \\ T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(n) = 2T\left(\frac{n}{2}\right) + O(n) \end{cases}$$

A_L A_R

Tree formula $\Rightarrow$ $O(n \log n)$

Great!

2. Suppose you get unlucky at every recursive call of QuickSort.

$$T(n) = \begin{cases} O(1) & n \le 1 \\ T(n-1) + O(n) \end{cases} \Rightarrow \text{Expand + Hope}$$

$T(n) = O(n^2)$

Awful

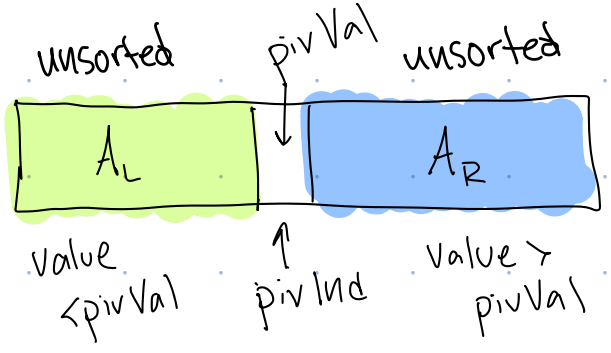
Partition (A, pivInd, pivVal)

- Swap pivot with A[1]
- Current ← 2
- While current ≤ |A|:
  - ★ If A[current] < pivVal:
    - Swap A[current], pivVal
    - Swap A[pivInd+1], pivVal
  - current ++

← every element of array is compared to pivot

Strategy: count the # of times ★ is run over the whole algorithm

After Partition:



Analyzing Average Runtime

1. Determine Sample Space S = set of all possible sequences of random events that might occur over the course of alg.

   ex: QuickSort → S = set of possible sequences of pivot choices that the alg might make

What is the sample space if QuickSort is run on

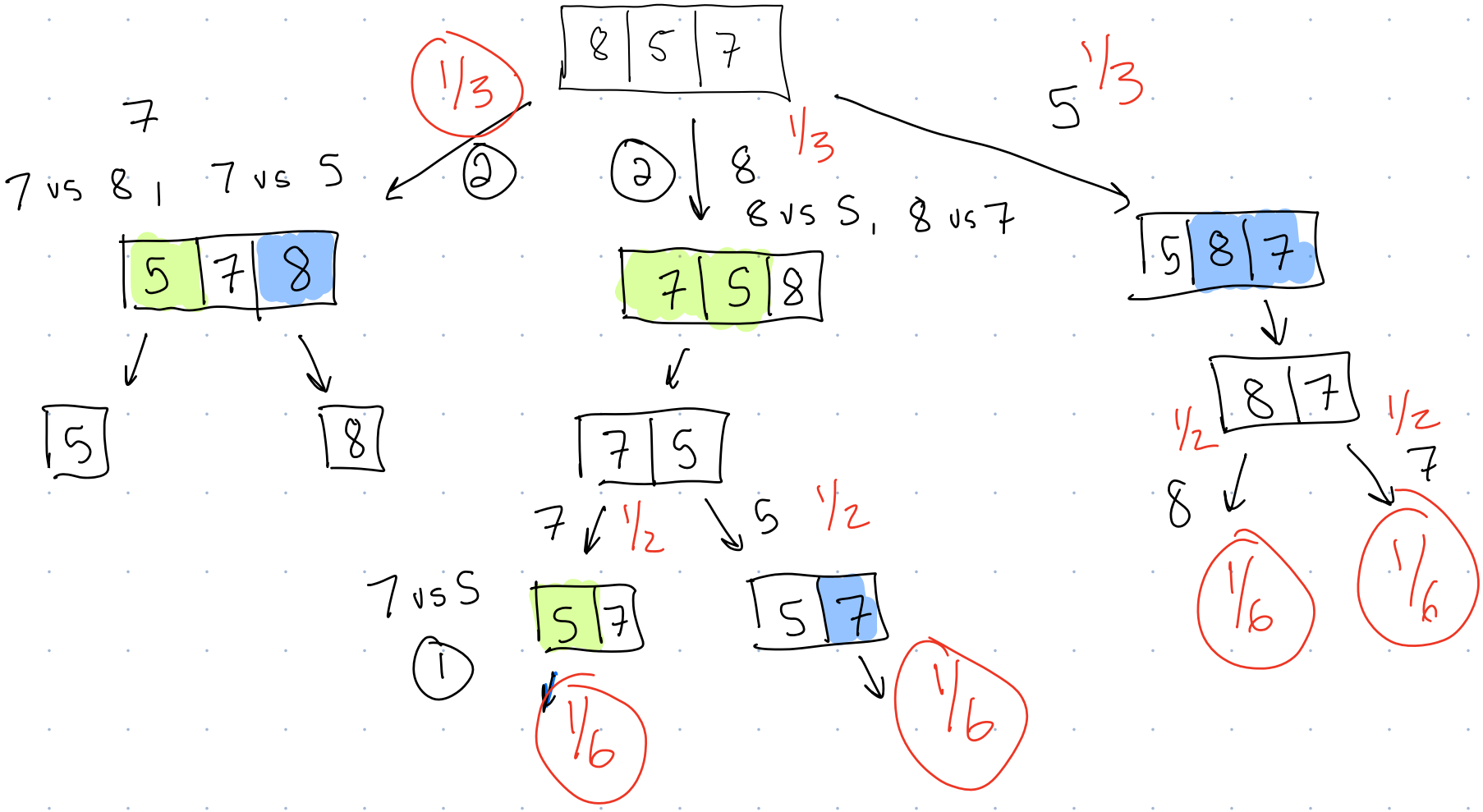
8	5	7

A) $S = \{8, 5, 7\}$

B) S = All possible permutations of $\{8, 5, 7\}$

C) S = Power set of $\{8, 5, 7\}$ (set of all subsets of $\{8, 5, 7\}$)

D) $S = \{(7), (8, 5), (8, 7), (5, 8), (5, 7)\}$



$$S = \{ 7, (8,7), (8,5), (5,8), (5,7) \}$$

$R(7) = 2$ $\quad$ $R(8,7) = 3$

Analyzing Average Runtime

2. Create a Random Variable that maps each element of the sample space to a number.

$$R : S \to \mathbb{R}$$

ex: QuickSort: $R(\sigma)$ = # of comparisons of our array if pivot sequence σ is chosen.

(# of times ✱ is run on 2 elements)

3. Take Expectation value of R to get average run time

$$\mathbb{E}[R] = \sum_{\sigma \in S} p(\sigma) \cdot R(\sigma)$$

(prob. of σ occuring)

ex: [8|5|7]

(7) (5 7) (5 8) (8 5) (8 7)

$= \frac{1}{3} \cdot 2 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 3 = 2\frac{2}{3}$

2. (Alternate) $R: S \to \mathbb{R}$ $\Rightarrow$ break up into a sum of simple random variables

$f(x) = x + x^2$	$f_1(x) = x$
$f = f_1 + f_2$	$f_2(x) = x^2$

$X_{ij}(\sigma)$ = # of comparisons between i^{th} smallest element of A (z_i) and the j^{th} smallest element of A (z_j)

8	5	7

$z_1 = 5$ $z_2 = 7$ $z_3 = 8$

$X_{23}((8,5)) = 1$ (# of times 7 and 8 are compared over the course of alg if pivot choices are 8, 5)

$$R(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma) \iff \text{total \# of comparisons done over course of alg.}$$

$$= \sum_{i<j} X_{ij}$$

4 $$\mathbb{E}[R] = \mathbb{E}\left[\sum_{i<j} X_{ij}\right]$$

$$= \sum_{i<j} \mathbb{E}[X_{ij}]$$

linearity of Expectation

To Analyze $E[X_{ij}]$, Consider:

- Pivot in recursive call
- In Partition, pivot is compared to every other elt, and that is it.

- Suppose z_i, z_j $(i<j)$ are both in a subarray that is input to some recursive call of QuickSort. For each of the following cases (*)
  - are z_i, z_j compared in this call?
  - are they kept together or separated in future recursive calls
  - * z_i or z_j chosen as pivot
  - * z_k chosen as pivot
    - * $k > i, j$
    - * $k < i, j$
    - * $i < k < j$

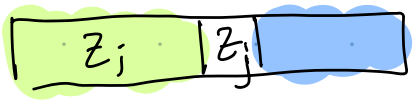
- What values can X_{ij} take (only 2 possible), and under which conditions does it take those values?
- What is probability of z_i, z_j being compared?

To Analyze $E[X_{ij}]$, consider:

1 ✱ z_i or z_j chosen as pivot (z_i)

- z_i, z_j are compared
- Separated and will never be compared again

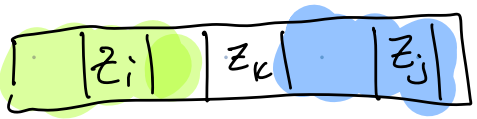
$X_{ij} = 1$



2 ✱ z_k chosen as pivot, $i < k < j$

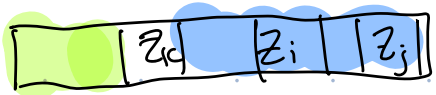
- z_i, z_j not compared
- Separated

$X_{ij} = 0$



✱ 3 z_k chosen as pivot, $k < i, j$

- z_i, z_j not compared
- Kept together



X_{ij} not decided, might be compared or not in future

$X_{ij} \in \{0, 1\}$ so X_{ij} indicator random variable

Back to Average Runtime:

$$\mathbb{E}[R(\sigma)] = \mathbb{E}\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} \mathbb{E}[X_{ij}]$$

$$\mathbb{E}[X_{ij}] = \sum_{\sigma \in S} \Pr(\sigma) \cdot X_{ij}(\sigma)$$

$$= \sum_{\substack{\sigma \in S: \\ X_{ij}(\sigma)=0}} \Pr(\sigma)\, X_{ij}(\sigma) + \sum_{\substack{\sigma \in S: \\ X_{ij}(\sigma)=1}} \Pr(\sigma) \cdot X_{ij}(\sigma)$$

$X_{ij}(\sigma)$ is 0 in the first sum and 1 in the second.

$$= \sum_{\substack{\sigma \in S \\ X_{ij}(\sigma)=1}} \Pr(\sigma)$$

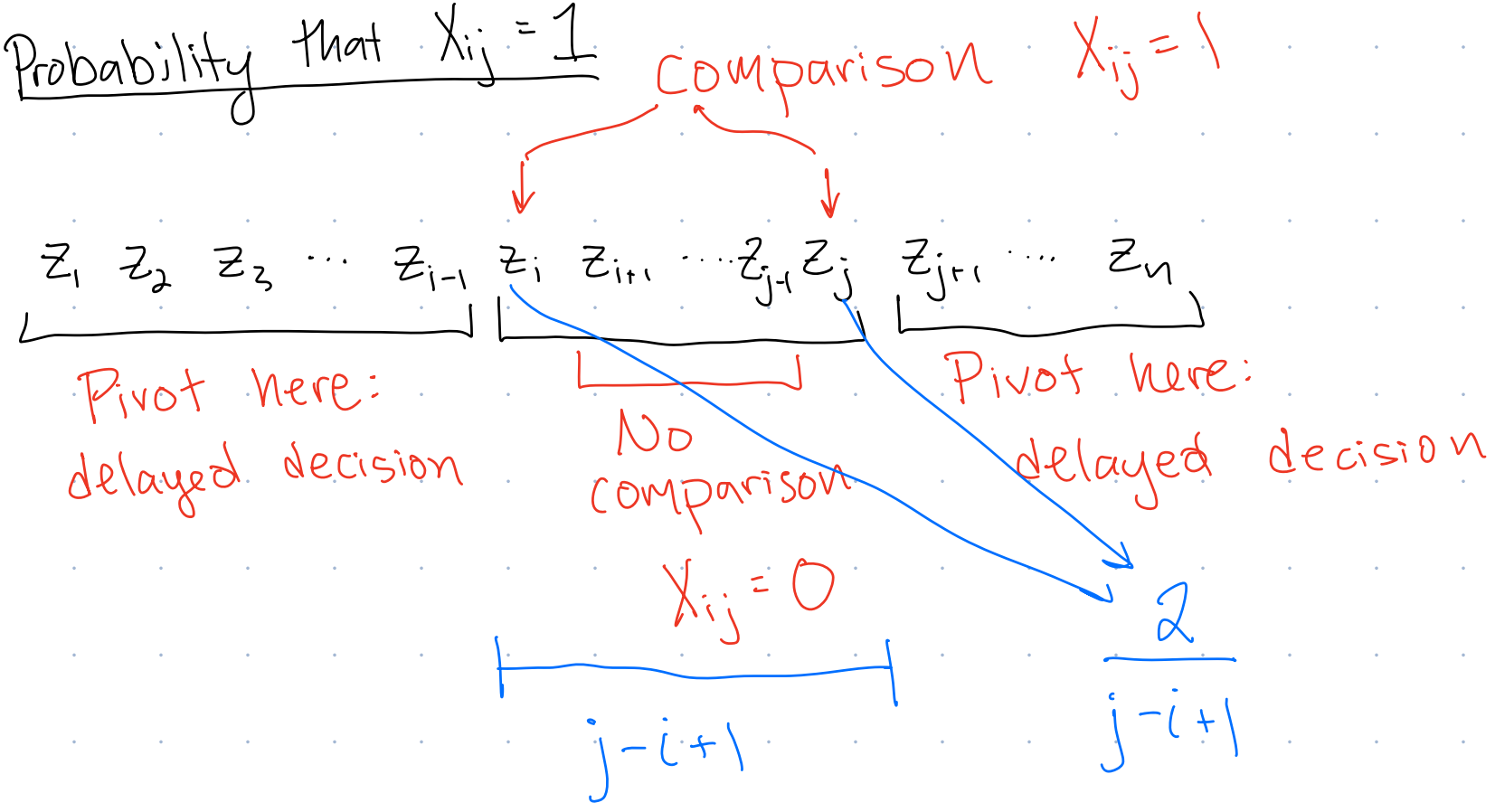
$$+ \sum_{\substack{\sigma \in S \\ X_{ij}(\sigma)=2}} \Pr(\sigma) \cdot 2$$

by def

$$= \text{Probability that } X_{ij} = 1$$

$$= \text{Probability that } z_i, z_j \text{ are compared}$$

Probability that $X_{ij} = 1$



What is the probability that z_i, z_j are compared?

A) $\frac{1}{j-i}$

B) $\frac{2}{j-i+1}$

~~C)~~ $\frac{2}{n}$

~~D)~~ $\frac{1}{n^2}$

$n = j \quad i = 1 \rightarrow \frac{2}{n}$ $\qquad$ $j = i+1 \quad \frac{2}{2}$

Continuing $\mathbb{E}[R]$ analysis:

$$\mathbb{E}[R] = \sum_{i<j} \mathbb{E}[X_{ij}] = \sum_{i<j} \Pr(z_i, z_j \text{ are compared})$$

$$= \sum_{i<j} \frac{2}{j-i+1}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$\frac{2}{(i+1)-i+1} = \frac{2}{2}$ $\quad$ $\frac{2}{(n)-i+1}$

$$= \sum_{i=1}^{n-1} \left[\frac{2}{2} + \frac{2}{3} + \frac{2}{4} + \cdots \frac{2}{n-i+1} \right]$$

$$< \sum_{i=1}^{n} \left[\frac{2}{1} + \frac{2}{2} + \frac{2}{3} + \frac{2}{4} + \cdots \frac{2}{n-i+1} + \frac{2}{n-i+2} + \frac{2}{n-i+3} + \cdots \frac{2}{n} \right]$$

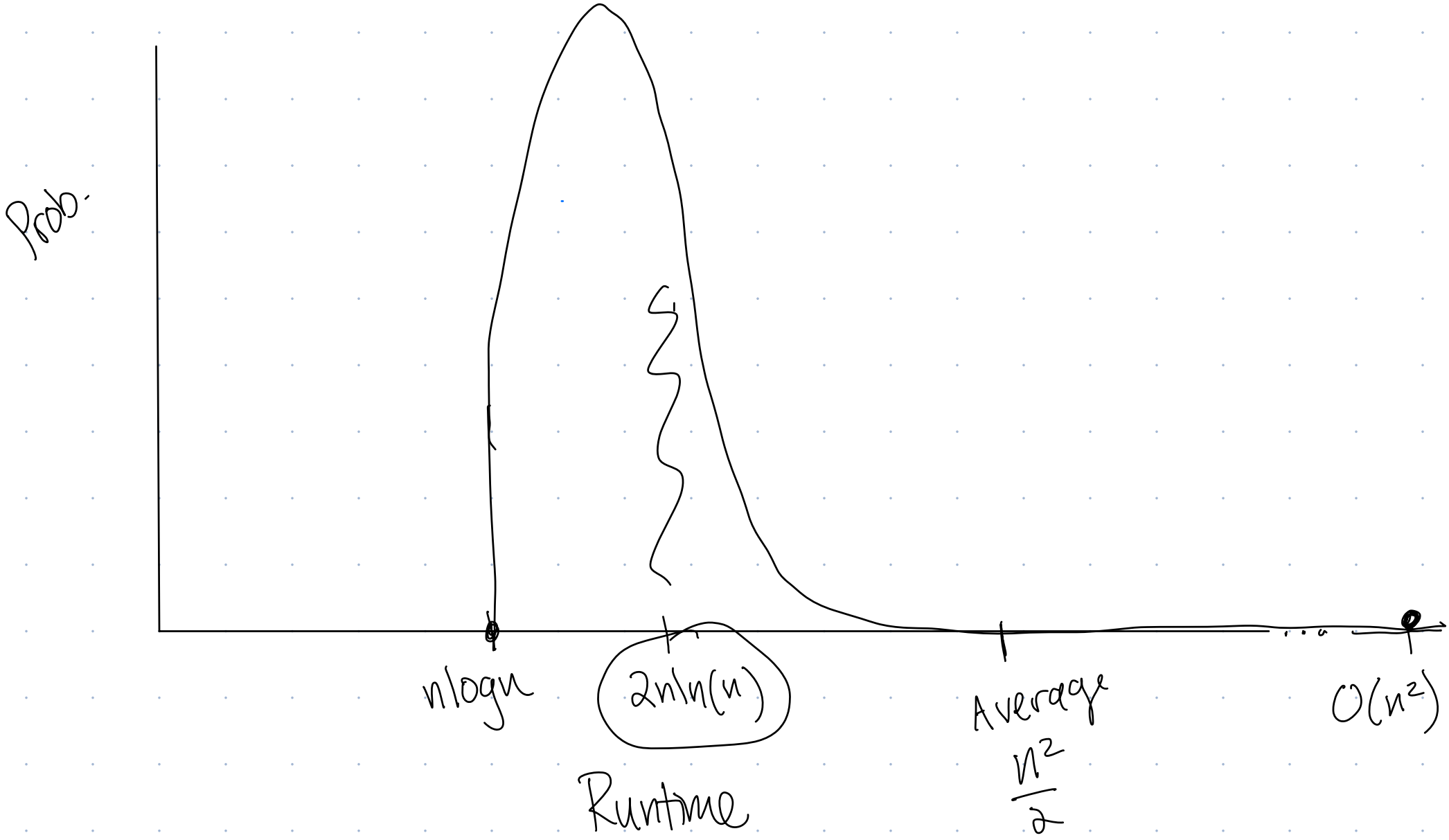
$$= \sum_{i=1}^{n} \left(2 \sum_{j=1}^{n} \frac{1}{j} \right)$$

math fact → Harmonic Series

$$\sum_{i=1}^{n} \frac{1}{i} < 2\ln(n)+1$$

$$< \sum_{i=1}^{n} 2(\ln(n)+1)$$

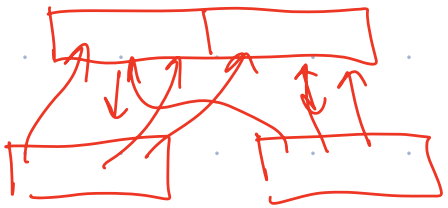
$$= 2n(\ln(n+1)) = O(n \log n)$$


Prob.
nlogn
2nln(n)
Runtime
Average
$\frac{n^2}{2}$
$O(n^2)$

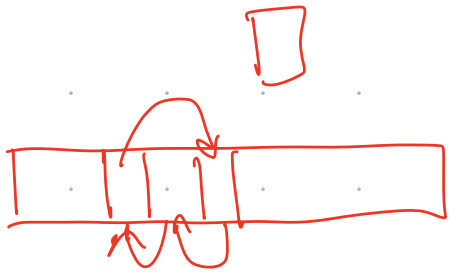
MergeSort? or QuickSort?

- Limited Space? QuickSort

MS



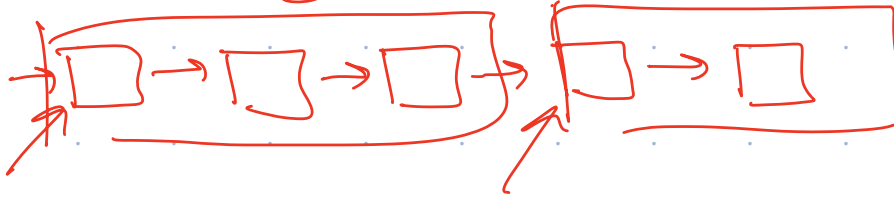
QS



- Sorting Multiple Lists in Parallel?

MS: All sorts end at same time

QS: Might have to wait for some to end



- Array as linked list?

Use mergesort

Hard to do swaps on a linked list.

- Small Array Insertion Sort

- Want speed, and array calls are quick? QuickSort