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.

Exit Tickets

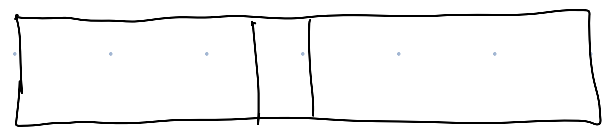
Exit Tickets

QuickSort

Input: Array A of unique integers
Output: Sorted A

- If $|A| = 1$: Return A
- pivInd ← randomly chosen index, with value pivVal
-
-
-

After Partition:

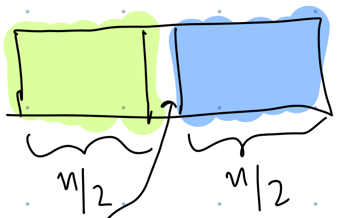


Key Pts

-
-

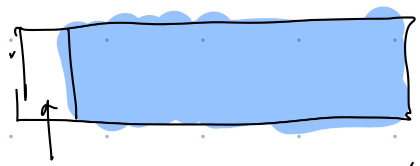
Lucky vs. Unlucky Pivot Choices

Lucky:



pivVal is median array value

Unlucky:



pivVal is smallest/largest array value

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 <u>lucky</u> at every recursive call or QuickSort.

2. Suppose you get <u>unlucky</u> at every recursive call of QuickSort.

Partition (A, pivInd, pivVal)

- Swap pivot with A[1]
- Current ← 2
- While current ≤ |A| :

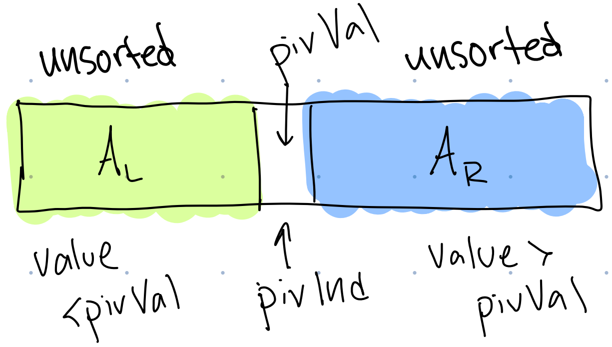
    If A[current] < pivVal:

        Swap A[current], pivVal

        Swap A[pivInd+1], pivVal

    current ++

After Partition:



Analyzing Average Runtime

1.

What is the sample space if QuickSort is run on

$\boxed{8 \mid 5 \mid 7}$

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

8	5	7

Analyzing Average Runtime

2.

3.

2. (Alternate)

4

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

- 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

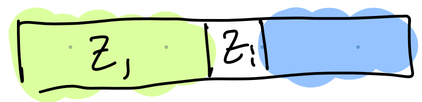
  ☆ z_k chosen as pivot

  ☆ $k > i, j$ ☆ $k < i, j$ ☆ $i < k < j$

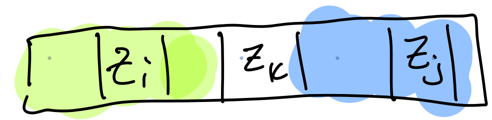
- What values can X_{ij} take (only 2 possible), and under which conditions does it take those values?
- What is probability of z_i, z_j being compared?

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

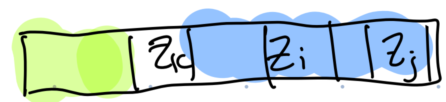
* z_i or z_j chosen as pivot (z_i)



* z_k chosen as pivot, $i<k<j$



* z_k chosen as pivot, $k<i,j$



Back to Average Runtime:

$$\mathbb{E}[R(\sigma)]$$

Probability that $X_{ij} = 1$

$$\underbrace{z_1 \quad z_2 \quad z_3 \quad \cdots \quad z_{i-1}} \quad \underbrace{z_i \quad z_{i+1} \quad \cdots \quad z_{j-1} \quad z_j} \quad \underbrace{z_{j+1} \quad \cdots \quad z_n}$$

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

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

$\mathbb{E}[R] =$

MergeSort? or QuickSort?

- Limited Space?

- Sorting Multiple Lists in Parallel?

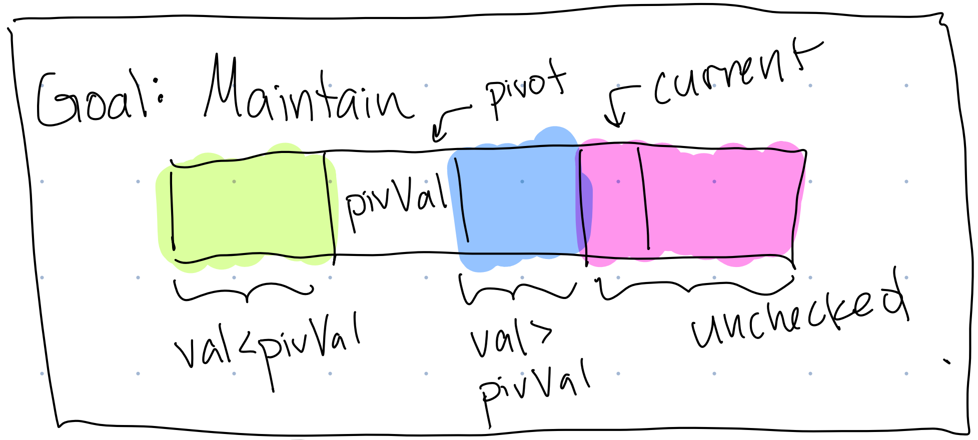
- Array as linked list?

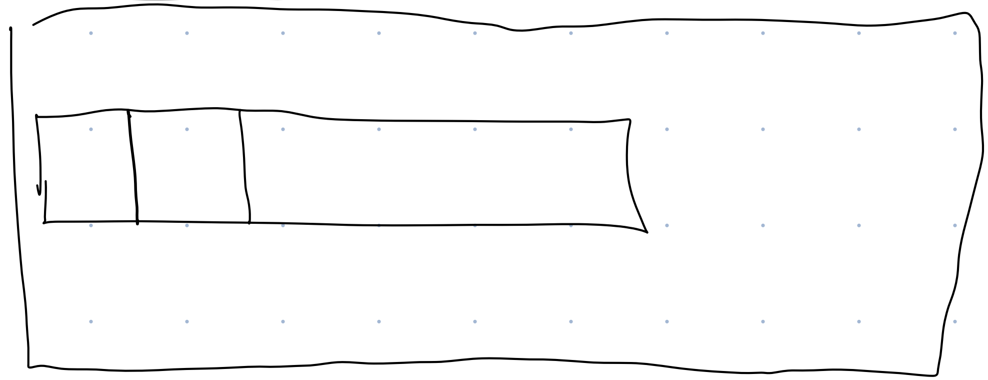
- Small Array

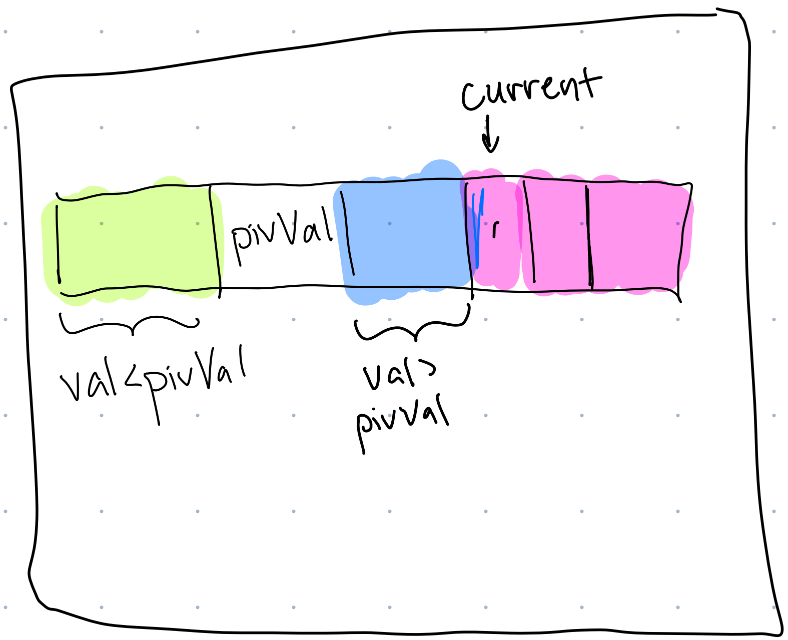
- Want speed, and array calls are quick?

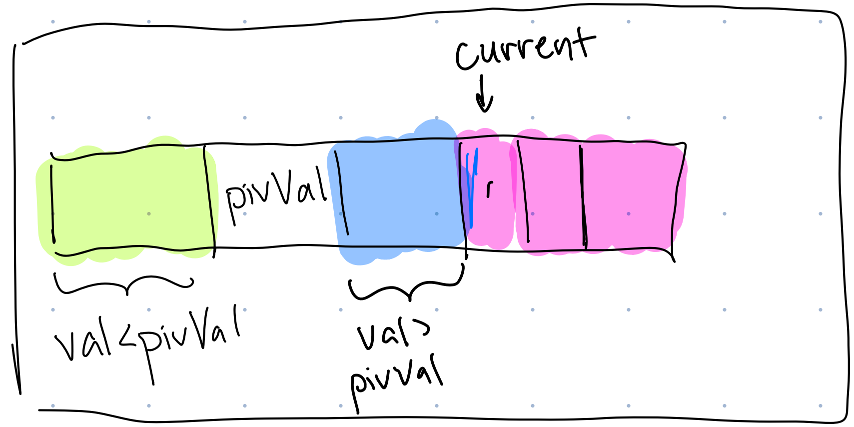
Partition (A, pivInd, pivVal)

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

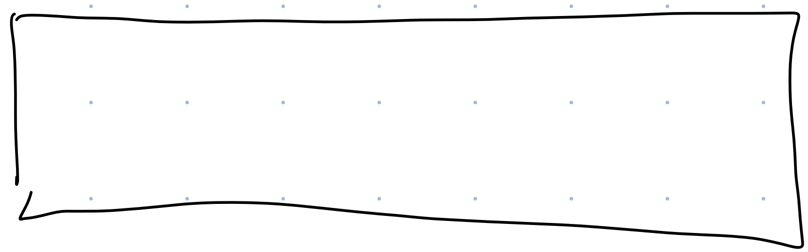








End:



QuickSort

Input: Array A of unique integers
Output: Sorted A

- If $|A| = 1$: Return A (Base case)
- pivot $\leftarrow$ randomly chosen index, with value pivVal
- Partition(A, pivot)
- QuickSort(A_L)
- QuickSort(A_R)

Divide + Conquer

How to analyze (average) runtime?

-
-

Partition(A, pivInd, pivVal

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

How many comparisons are done by Partition if A has size n?

A) Depends on pivot choice

B) $n-1$

C) $O(n)$

D) $O(n\log n)$