QuickSort

Input: Array A of unique integers

Output: Sorted A

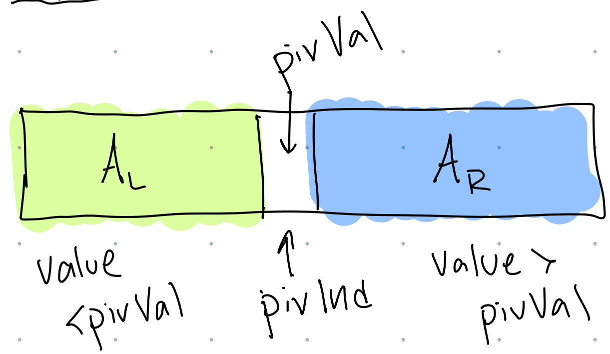
pivInd ⇒ pivot index

pivVal ⇒ pivot value

- If $|A| = 1$: Return A (Base case)
- pivInd ← randomly chosen index, with value pivVal
- Partition(A, pivInd, A[pivInd])
- QuickSort(A_L)
- QuickSort(A_R)

} Divide + Conquer

After Partition:

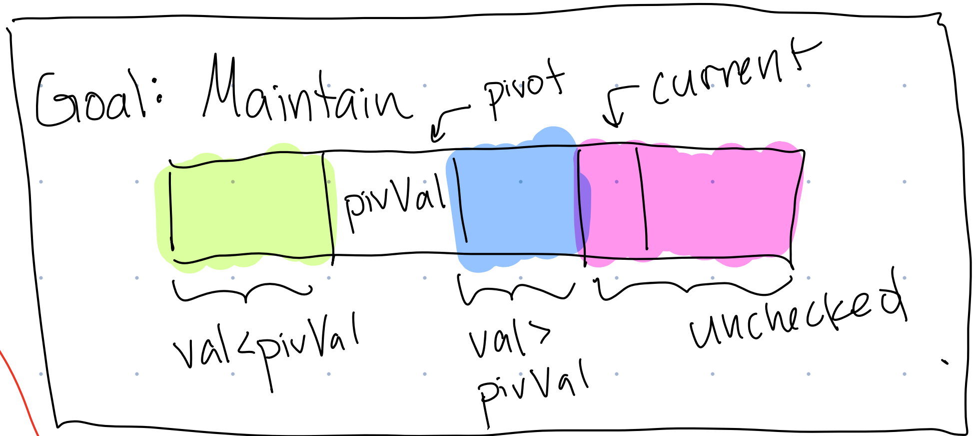


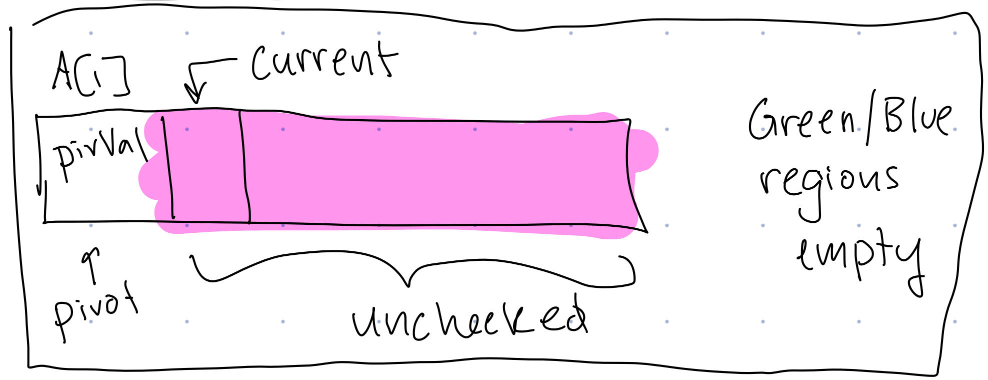
Key Pts

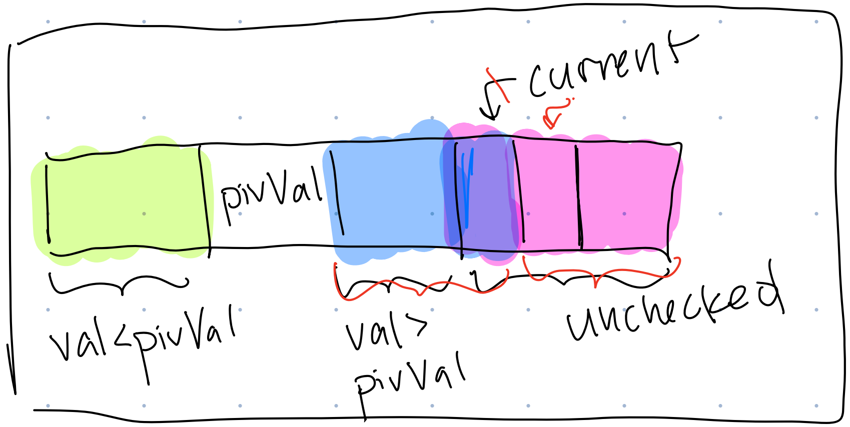
- Partition takes $O(|A|)$ time
- If pivVal is z_i (i^{th} smallest element of A), after partition, pivVal is at position i

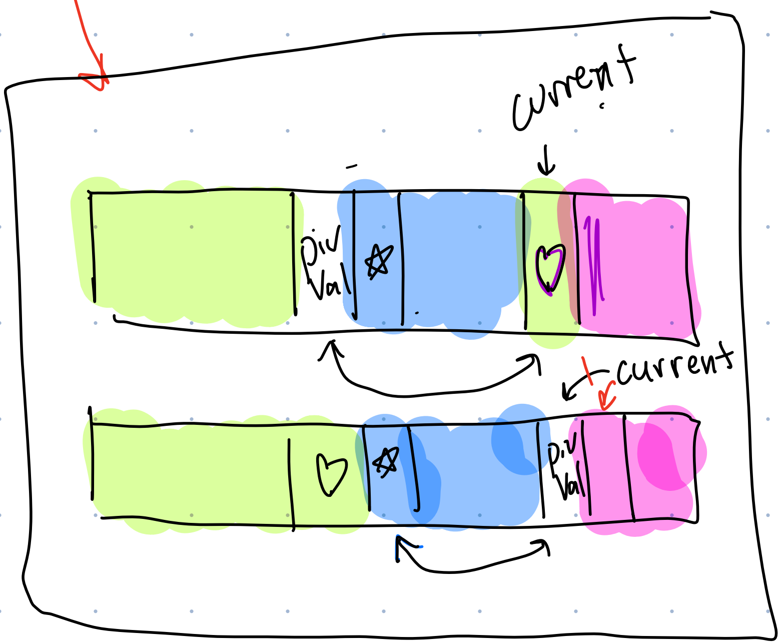
Partition (A, pivInd, pivVal)

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

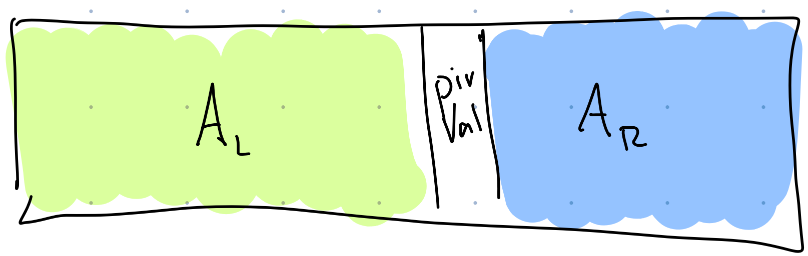








End:

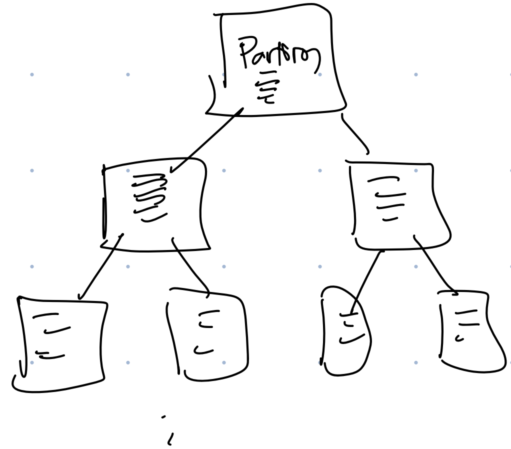


QuickSort

Input: Array A of unique integers
Output: Sorted A

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

Divide + Conquer



How to analyze (average) runtime?

- Notice: the Partition subroutine takes up the most time in each recursive call
- The runtime of partition scales with the # of comparisons

$\boxed{\text{- If } A[\text{current}] < \text{pivVal}}$

If count the number of comparisons over the whole algorithm, will give us the runtime scaling

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

$\boxed{\text{B) } n-1}$

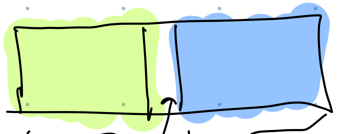
C) O(n)

D) O(nlogn)

pivVal gets compared to every other element, n-1 other elts, so n-1 comparisons

Lucky vs. Unlucky Pivot Choices

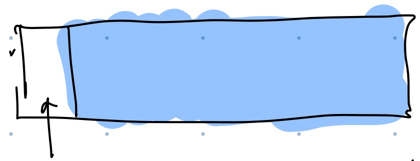
Lucky:



$n/2$ $n/2$

pivVal is median array value, $z_{n/2}$

Unlucky:



pivVal is smallest/largest array value → z_1 or z_n

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.

partition — Base case

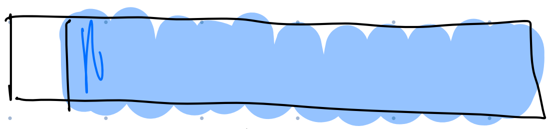
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \quad (\text{if } n \geq 2) \; ; \; O(1) \quad (\text{if } n = 1)$$

2 recursive calls — A_L, A_R both size $n/2$

Tree Formula $\Downarrow$

$$T(n) = O(n \log n)$$ $\leftarrow$ GOOD!

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



A_R

No A_L, pivot is in correct location, only 1 recursive call needed!

1 recursive call on A_R — Partition

$$\boxed{T(n) = \begin{cases} T(n-1) + O(n) & n \geq 2 \\ O(1) & n = 1 \end{cases}}$$

$\xRightarrow[\text{+ Hope}]{\text{Expand}}$ $T(n) = O(n^2)$ $\leftarrow$ AWFUL!

QuickSort not so quick??
Divide + Conquer??

Analyzing Average Runtime

1. Determine "Sample Space": S = Set of all possible sequences of random events that might occur over the course of the algorithm.

   ex: QuickSort → S = set of possible pivot choices of the algorithm

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)}

[8 | 5 | 7]

1/3 1/3 1/3

- 7 chosen as 1st pivot:

[5 | 7 | 8]

Base Case

[5] [8]

No further pivot choices!

(7)

$P(7) = 1/3$

- 8 chosen as 1st pivot

[7 | 5 | 8]

[7 | 5]

1/2 1/2

2 options: 5 or 7 could be pivot

(8, 5), (8, 7)

5 → pivot

[5 | 8 | 7]

1/2 1/2

8 7

(5, 8), (5, 7)

$P(8,5) = P(8,7) = P(5,8) = P(5,7) = \frac{1}{6}$

Analyzing Average Runtime

Go back to cnxn of Partition/comparison

2. Create "Random Variable" that scales with runtime

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

random variable is a function that maps each element of sample space to a $\boxed{\text{number}}$

QuickSort: $R(\sigma) = \boxed{\text{\# of comparisons}}$ of QuickSort if pivot sequence σ is chosen

8	5	7

$\Rightarrow R(7) < R((8,7))$

3. Take Expectation Value of R to get average runtime:

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

probability of σ occuring

Problem: S, R, p all get hard to determine for large n

2. (Alternate) Create $R: S \to \mathbb{R}$, but break into sum of other simpler random variables

e.g.

$$\boxed{\begin{array}{ll} f(x) = x + x^2 & f_1(x) = x \\ \Downarrow & f_2(x) = x^2 \\ f = f_1 + f_2 & \end{array}}$$

QuickSort: Simpler random variable:

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

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

(Z_i = i^{th} smallest)

$Z_1 = 5$, $Z_2 = 7$ $Z_3 = 8$

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

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

$$R(\sigma) = \sum_{i<j} X_{ij}(\sigma)$$

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

4 (alternate) Use linearity of expectation

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

take $\mathbb{E}$ of both sides

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

move $\mathbb{E}$ inside sum

Hopefully easier than $\mathbb{E}[R]$

To Analyze $E[X_{ij}]$, Consider: • pivot never in recursive call

• only pivot compared to each other elt

- 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

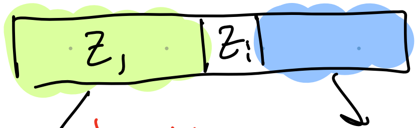
  ☆ $k > i, j$ ☆ $k < i, j$ ☆ $i < k < j$

- 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:

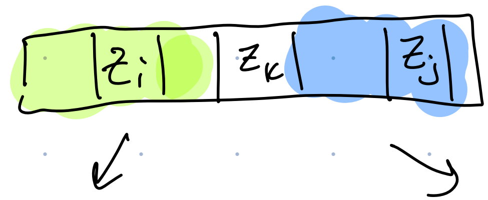
☆ z_i or z_j chosen as pivot (z_i)

- z_i, z_j compared (pivot is compared to every other elt)



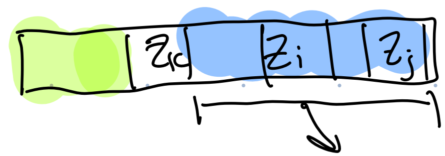
- Separated
  ↳ can't be compared again! $X_{ij} = 0$

☆ z_k chosen as pivot, $i<k<j$



- z_i, z_j not compared
- separated → $X_{ij} = 0$

☆ z_k chosen as pivot, $k < i, j$



- z_i, z_j not compared
  (comparison always involves pivot)
- kept together
  ↳ X_{ij} not decided, might be compared or not in future

only takes value 0, 1

$X_{ij} = 0$ or 1, X_{ij} is an indicator random variable

Back to Average Runtime:

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

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

0 or 1

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

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

= Probability that $X_{ij} = 1$

$z_1 \quad z_2 \quad z_3 \quad \cdots \quad z_{i-1} \quad z_i$

Probability that $X_{ij} = 1$

If pivot, $X_{ij} = 0$

$$\underbrace{z_1 \ z_2 \ z_3 \ \cdots \ z_{i-1}}_{\text{Pivot here, delayed decision}} \ \underbrace{z_i \ \overbrace{z_{i+1} \ \cdots \ z_{j-1}}^{\text{If pivot, } X_{ij}=0} \ z_j}_{\text{Pivot here, decision}} \ \underbrace{z_{j+1} \ \cdots \ z_n}_{\text{Pivot here, delayed decision}}$$

Either pivot, then $X_{ij} = 1$

What is the probability that z_i, z_j are compared?

A) $\frac{1}{j-i}$ B) $\boxed{\frac{2}{j-i+1}}$ C) $\frac{2}{n}$ D) $\frac{1}{n^2}$

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

$$\mathbb{E}[R] = \sum_{i<j} \Pr(X_{ij} = 1)$$

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

expand

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

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

useful math fact

$$\boxed{\sum_{i=1}^{n} \frac{1}{n} \leq \underbrace{\ln}_{\log_e(n)}(n) + 1}$$

$$\leq \sum_{i=1}^{n-1} 2\left[\ln(n) + 1\right]$$

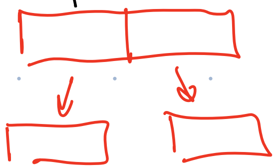
$$= 2(n-1)\left[\ln(n) + 1\right]$$

$$= O(n \log n)$$

MergeSort? or QuickSort?

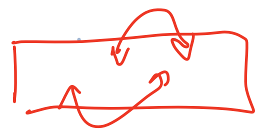
- Limited Space? QuickSort

MS:



Makes copy at each call

QS:



Partition swaps in place

- Sorting Multiple Lists in Parallel? MergeSort

MS: Each parallel call will terminate at same time

QS: " " random time

- Array as linked list? MergeSort

Hard to do swaps on linked list

- Small Array Insertion Sort
- Want speed, and array calls are quick? QuickSort