

# Notes on Quicksort

Geoffrey Matthews

May 1, 2018

# Quicksort

- ▶  $\Theta(n^2)$  worst case.
- ▶  $\Theta(n \lg n)$  expected running time.
- ▶ Constants are small.
- ▶ Sorts in place.

## Quicksort: three step process

- ▶ To sort  $A[p..r]$ :
  - ▶ **Divide:** Partition  $A[p..r]$  into two (possibly empty) subarrays  $A[p..q-1]$  and  $A[q+1..r]$ , such that each element in the first subarray is  $\leq A[q]$  and  $A[q] \leq$  each element in the second subarray.
  - ▶ **Conquer:** Sort the two subarrays by recursive calls.
  - ▶ **Combine:** Nothing needs to be done.

QUICKSORT( $A, p, r$ )

```
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )
```

Initial call is QUICKSORT( $A, 1, n$ )

# Compare QUICKSORT and MERGESORT

QUICKSORT( $A, p, r$ )

**if**  $p < r$

$q = \text{PARTITION}(A, p, r)$

    QUICKSORT( $A, p, q - 1$ )

    QUICKSORT( $A, q + 1, r$ )

MERGE-SORT( $A, p, r$ )

**if**  $p < r$

$q = \lfloor (p + r) / 2 \rfloor$

    MERGE-SORT( $A, p, q$ )

    MERGE-SORT( $A, q + 1, r$ )

    MERGE( $A, p, q, r$ )

// check for base case

// divide

// conquer

// conquer

// combine

# Compare PARTITION and MERGE

PARTITION( $A, p, r$ )

$x = A[r]$

$i = p - 1$

**for**  $j = p$  **to**  $r - 1$

**if**  $A[j] \leq x$

$i = i + 1$

        exchange  $A[i]$  with  $A[j]$

exchange  $A[i + 1]$  with  $A[r]$

**return**  $i + 1$

MERGE( $A, p, q, r$ )

$n_1 = q - p + 1$

$n_2 = r - q$

let  $L[1 \dots n_1 + 1]$  and  $R[1 \dots n_2 + 1]$  be new arrays

**for**  $i = 1$  **to**  $n_1$

$L[i] = A[p + i - 1]$

**for**  $j = 1$  **to**  $n_2$

$R[j] = A[q + j]$

$L[n_1 + 1] = \infty$

$R[n_2 + 1] = \infty$

$i = 1$

$j = 1$

**for**  $k = p$  **to**  $r$

**if**  $L[i] \leq R[j]$

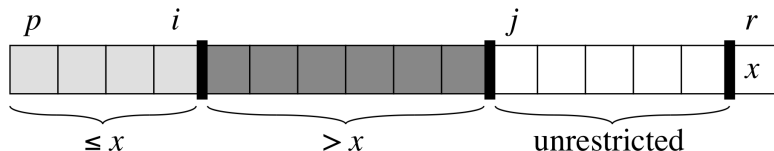
$A[k] = L[i]$

$i = i + 1$

**else**  $A[k] = R[j]$

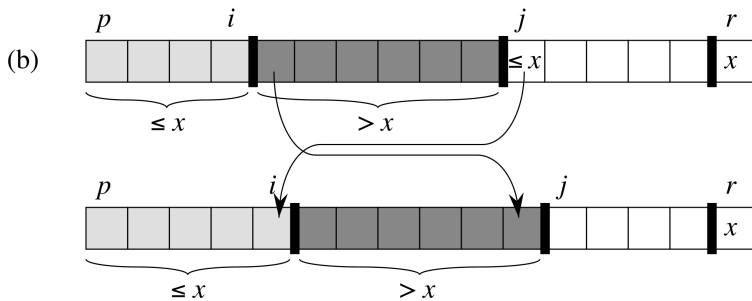
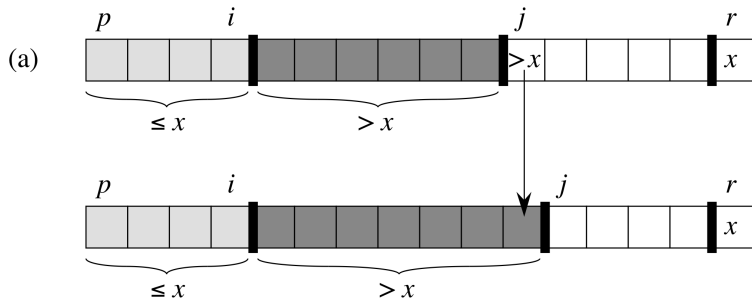
$j = j + 1$

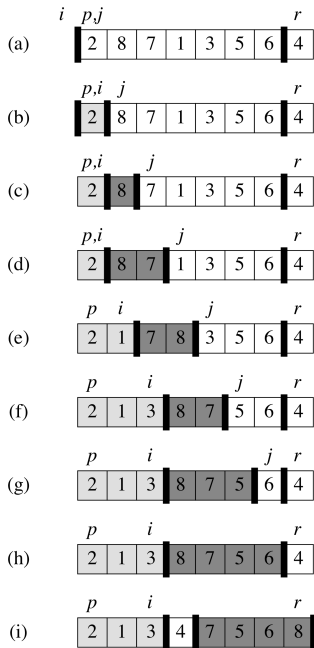
# Partition



Loop invariant:

1. All entries in  $A[p, \dots, i]$  are  $\leq$  pivot
2. All entries in  $A[i + 1, \dots, j - 1]$  are  $>$  pivot
3.  $A[r] =$  pivot





PARTITION( $A, p, r$ )

$x = A[r]$

$i = p - 1$

**for**  $j = p$  **to**  $r - 1$

**if**  $A[j] \leq x$

$i = i + 1$

        exchange  $A[i]$  with  $A[j]$

exchange  $A[i + 1]$  with  $A[r]$

**return**  $i + 1$



# Partition

PARTITION( $A, p, r$ )

```
1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4      if  $A[j] \leq x$ 
5           $i = i + 1$ 
6          exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

► Always selects  $A[r]$  as the **pivot**

► Loop invariant:

1. All entries in  $A[p, \dots, i]$  are  $\leq$  pivot
2. All entries in  $A[i + 1, \dots, j - 1]$  are  $>$  pivot
3.  $A[r] =$  pivot

$O(n)$

Can MERGESORT be done in place?

1	3	5	7	9	0	2	4	6	8
---	---	---	---	---	---	---	---	---	---

Can MERGESORT be done in place?

1	3	5	7	9	0	2	4	6	8
---	---	---	---	---	---	---	---	---	---

- Yes, but *very* tricky!

# Running time of quicksort

- ▶ Depends on partitioning of subarrays.
- ▶ If subarrays are balanced: fast as mergesort.
- ▶ If subarrays are unbalanced: slow as insertion sort.

# Worst case for quicksort

- ▶ Arrays completely unbalanced.
- ▶ 0 elements in one and  $n - 1$  in the other
- ▶ Recurrence:

$$\begin{aligned}T(n) &= T(n - 1) + T(0) + \Theta(n) \\&= T(n - 1) + \Theta(n) \\&= \Theta(n^2)\end{aligned}$$

- ▶ Same as insertion sort.
- ▶ Worst case for quicksort is the array is already sorted.
- ▶ This is the best case for insertion sort, which is  $O(n)$ .

# Best case for quicksort

- ▶ Each subarray has  $n/2$  elements.
- ▶ Recurrence:

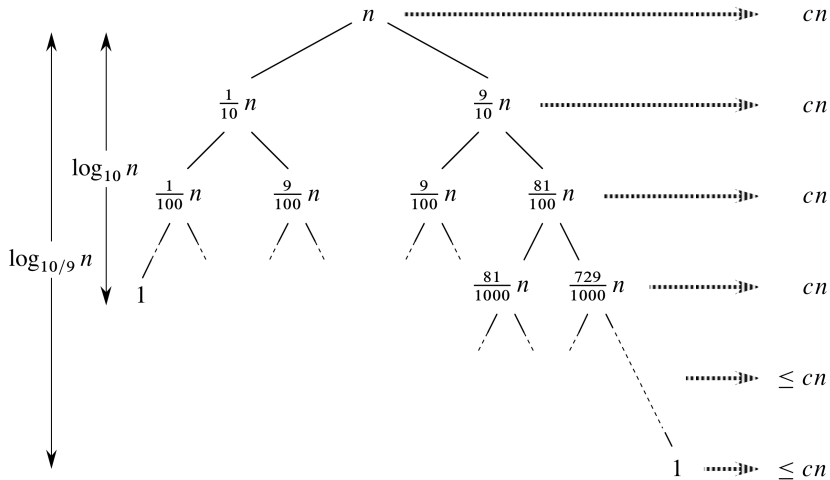
$$\begin{aligned}T(n) &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \lg n)\end{aligned}$$

- ▶ Same as mergesort.

# “Average” running time for quicksort

Assume PARTITION always makes a 9-to-1 split.

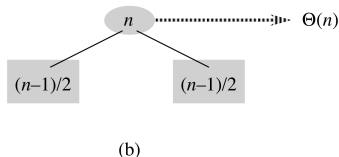
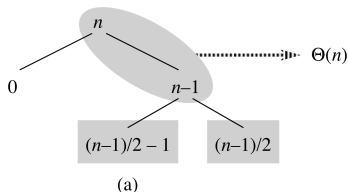
$$T(n) \leq T(9n/10) + T(n/10) + \Theta(n) = O(n \lg n)$$



---

$$O(n \lg n)$$

# “Average” running time for quicksort



- ▶ If levels alternate between good and bad splits, still  $O(n \lg n)$
- ▶ If we randomize choice of pivot, what is the probability that all of the choices will be worst case?
- ▶ If we randomize choice of pivot, what is the probability that more than half of the choices will be worst case?



# Randomized quicksort

- ▶ Instead of randomizing the entire array, which adds a large constant factor, just randomize the choice of pivot.

RANDOMIZED-PARTITION( $A, p, r$ )

```
1   $i = \text{RANDOM}(p, r)$   
2  exchange  $A[r]$  with  $A[i]$   
3  return PARTITION( $a, p, r$ )
```

RANDOMIZED-QUICKSORT( $A, p, r$ )

```
1  if  $p < r$   
2       $q = \text{RANDOMIZED-PARTITION}(A, p, r)$   
3      RANDOMIZED-QUICKSORT( $A, p, q - 1$ )  
4      RANDOMIZED-QUICKSORT( $A, q + 1, r$ )
```

- ▶ On average, the splits will be well balanced.
- ▶  $O(n \lg n)$  virtually guaranteed when  $n$  is large.
- ▶ Stops any bad input from causing worst-case behavior.

## Worst-case analysis of randomized quicksort

$$T(n) = \max_{0 \leq q \leq n-1} (T(q) + T(n - q - 1)) + \Theta(n)$$

- **Guess:**  $T(n) \leq cn^2$  for some  $c$ .

$$\begin{aligned} T(n) &\leq \max_{0 \leq q \leq n-1} (cq^2 + c(n - q - 1)^2) + \Theta(n) \\ &= c \cdot \max_{0 \leq q \leq n-1} (q^2 + (n - q - 1)^2) + \Theta(n) \end{aligned}$$

- This is max when  $q = 0$  or  $q = n - 1$  (parabolas).

$$\max_{0 \leq q \leq n-1} (q^2 + (n - q - 1)^2) \leq (n - 1)^2 = n^2 - 2n + 1$$

$$\begin{aligned} T(n) &\leq cn^2 - c(2n - 1) + \Theta(n) \\ &\leq cn^2 && \text{if } c(2n - 1) \geq \Theta(n) \\ &= O(n^2) \end{aligned}$$

- Can also show  $T(n) = \Omega(n^2)$ , so  $T(n) = \Theta(n^2)$ .

# Average-case analysis of randomized quicksort

- ▶ The dominant cost of the algorithm is in the calls to `PARTITION`.
- ▶ `PARTITION` removes the pivot from future consideration.
- ▶ `PARTITION` is called at most  $n$  times.
- ▶ Each call to `PARTITION` does a constant amount of work plus a constant times the number of comparisons done in the **for** loop.
- ▶ Let  $X$  be the total number of comparisons performed in all calls to `PARTITION`.
- ▶ Total work done is  $O(n + X)$ .
- ▶ We seek a bound on the total number of comparisons.

# Average-case analysis of randomized quicksort

- ▶ Let the elements of  $A$  be  $z_1, z_2, \dots, z_n$  with  $z_i$  being the  $i$ th smallest.
- ▶ Define  $Z_{ij} = \{z_i, z_{i+1}, \dots, z_j\}$
- ▶ Each pair of elements is compared at most once.
  - ▶ Compared only to the pivot, and then the pivot is removed.
- ▶ Let  $X_{ij} = \mathbf{I}\{z_i \text{ is compared to } z_j\}$
- ▶ Since each pair is compared at most once,

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij}$$

$$\begin{aligned} E[X] &= E \left[ \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij} \right] \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \Pr\{z_i \text{ is compared to } z_j\} \end{aligned}$$

## Probability $z_i$ is compared to $z_j$

- ▶ Numbers in separate partitions will not be compared.
- ▶ If a pivot  $x$  is chosen such that  $z_i < x < z_j$ ,  $z_i$  and  $z_j$  will not be compared.
- ▶ If either  $z_i$  or  $z_j$  is chosen before any other element of  $Z_{ij}$ , then it will be compared to every element of  $Z_{ij}$ , except itself.
- ▶ The probability that  $z_i$  is compared to  $z_j$  is the probability that either  $z_i$  or  $z_j$  is chosen first.
- ▶ There are  $j - i + 1$  elements of  $Z_{ij}$ , and pivots are chosen randomly and independently.
- ▶ The probability that any one of them is chosen first is  $1/(j - i + 1)$ .

$$\begin{aligned}\Pr\{z_i \text{ is compared to } z_j\} &= \Pr\{z_i \text{ or } z_j \text{ is chosen first from } Z_{ij}\} \\ &= \Pr\{z_i \text{ is chosen first}\} + \Pr\{z_j \text{ is chosen first}\} \\ &= \frac{2}{j - i + 1}\end{aligned}$$

## Expected number of comparisons

$$\begin{aligned} E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \Pr\{z_i \text{ is compared to } z_j\} \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\ &< \sum_{i=1}^{n-1} \sum_{k=1}^n \frac{2}{k} \\ &= \sum_{i=1}^{n-1} O(\lg n) \\ &= O(n \lg n) \end{aligned}$$