

Quicksort Running Time

<https://cs.pomona.edu/classes/cs140/>

Outline

Topics and Learning Objectives

- Learn how quicksort works
- Learn how to partition an array

Exercise

- Running time

Extra Resources

- <https://me.dt.in.th/page/Quicksort/>
- <https://www.youtube.com/watch?v=ywWBy6J5gz8>
- CLRS Chapter 7
- Algorithms Illuminated Chapter 5

Choosing a Pivot

What is Quicksort's running time? (can we use master theorem?)

- It depends on the pivot

What is a recurrence for Quicksort?

Choosing a Pivot

What is Quicksort's running time? (can we use master theorem?)

- It depends on the pivot

What is the worst case for Quicksort, and what is its running time?

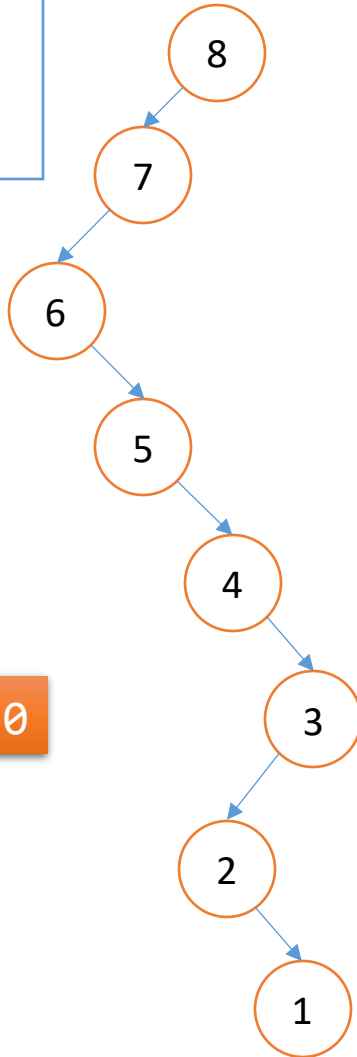
- Always select the smallest (or largest) possible pivot and it takes $O(n^2)$
- Think of a one-sided tree

What is the best case for Quicksort, and what is its running time?

- Always select the median element as a pivot leading to $O(n \lg n)$
- Think of a balanced tree

Recursion tree for the **worst** and **best** cases of Quicksort

Let's assume
the cost of
Partition is 5m



$$T(n) = 180$$

$$5n = 40$$

$$+ 5(n-1) = 35$$

$$+ 5(n-2) = 30$$

$$+ 5(n-3) = 25$$

$$+ 5(n-4) = 20$$

$$+ 5(n-5) = 15$$

$$+ 5(n-6) = 10$$

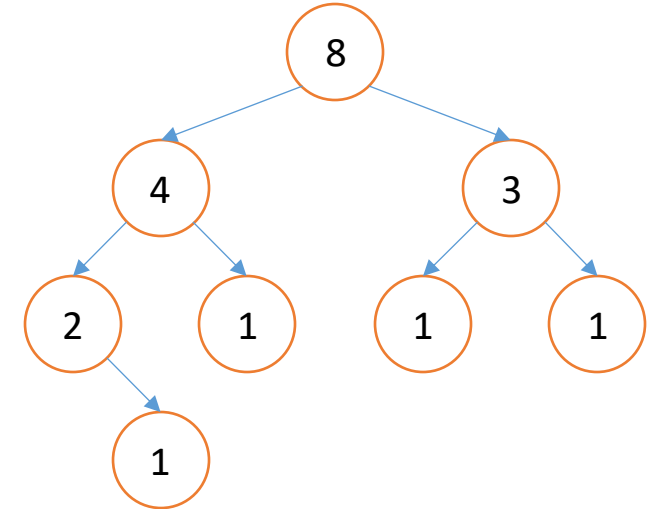
$$+ 5(n-7) = 5$$

$$5n = 40$$

$$+ 5(n-1) = 35$$

$$+ 5(n-3) = 25$$

$$+ 5(n-7) = 5$$



$$T(n) = 105$$

How would you select a pivot?

- If pivot selection is so important, how should we do it?
- Shouldn't we take great care in selecting the pivot?
- **Key idea for Quicksort: select the pivot uniformly at random!**
 - Easy
 - Fast
 - Gets good results as long as the pivots are “decent” fairly “often”

Random Pivots

- Some foreshadowing:

If the randomly chosen pivot is close to the median (in the middle 25-75 percentile range) we will get an average running time of $O(n \lg n)$

- We cannot use the master theorem
- We are going to show the runtime of quicksort another way

What is our Quicksort Theorem?

Theorem: For every input of the array of length n , the **average** running time of quicksort with random pivots is $O(n \lg n)$.

This is a big deal; it means that the average running time is closer to the best-case than it is to the worst-case.

Note: here, average refers to the algorithm itself—it does not depend on the input.

- If we re-run quicksort on the same input we will get different pivots each time, and we are talking about the average running time of quicksort for these different sequences of pivots on the same input array.

Quicksort

FUNCTION QuickSort(array, left_index, right_index)

IF left_index \geq right_index

RETURN

MovePivotToLeft(left_index, right_index)

pivot_index = Partition(array, left_index, right_index)

QuickSort(array, left_index, pivot_index)

QuickSort(array, pivot_index + 1, right_index)

Most of the work is done inside Partition

We are going to count the number of comparisons performed inside the for-loop.

FUNCTION Partition(array, left_index, right_index)

pivot_value = array[left_index]

i = left_index + 1

FOR j **IN** [left_index + 1 ..< right_index]

IF array[j] < pivot_value

swap(array, i, j)

i = i + 1

swap(array, left_index, i - 1)

RETURN i - 1

Some notation

Let $Z_i = i^{\text{th}}$ smallest element of A (not the i^{th} element)

Z_i

What is Z_1

Z								
Index	0	1	2	3	4	5	6	7
Value	51	43	17	83	79	23	61	37

Z_i

What is Z_2

z			Z_1					
Index	0	1	2	3	4	5	6	7
Value	51	43	17	83	79	23	61	37

z_i

z			z_1			z_2		
Index	0	1	2	3	4	5	6	7
Value	51	43	17	83	79	23	61	37

z_i

z	z_5	z_4	z_1	z_8	z_7	z_2	z_6	z_3
Index	0	1	2	3	4	5	6	7
Value	51	43	17	83	79	23	61	37

Some notation

Let Z_i = i^{th} smallest element of A (not the i^{th} element)

Let $X_{i,j}$ be a random variable for the number of times Z_i and Z_j get compared during a call to Quicksort

i and j can be any indices, but I'll normally use i for the lower index

How many times can Z_i and Z_j possibly be compared?

$X_{2,4}$

z	z_5	z_4	z_1	z_8	z_7	z_2	z_6	z_3
Index	0	1	2	3	4	5	6	7
Value	51	43	17	83	79	23	61	37

Exercise Question 1

How many times can two elements be compared by a single run of the Quicksort algorithm?

Some notation

Let Z_i = i^{th} smallest element of A (not the i^{th} element)

Let $X_{i,j}$ be a random variable for the number of times Z_i and Z_j get compared during a call to Quicksort

How many times can Z_i and Z_j possibly be compared?

- Can only be compared 0 or 1 times!
- Every comparison involves the pivot, but the pivot is excluded from recursive calls.

$$X_{2,4} =$$

$$X_{1,7} =$$

left_index

right_index

z	z_5	z_4	z_1	z_8	z_7	z_2	z_6	z_3
Index	0	1	2	3	4	5	6	7
Value	51	43	17	83	79	23	61	37

Every comparison involves the pivot, but the pivot is excluded from recursive calls.

FUNCTION QuickSort(array, left_index, right_index)

IF left_index \geq right_index

RETURN

MovePivotToLeft(left_index, right_index)

pivot_index = Partition(array, left_index, right_index)

QuickSort(array, left_index, pivot_index)

QuickSort(array, pivot_index + 1, right_index)

FUNCTION Partition(array, left_index, right_index)

pivot_value = array[left_index]

i = left_index + 1

FOR j **IN** [left_index + 1 ..< right_index]

IF array[j] < pivot_value

swap(array, i, j)

i = i + 1

swap(array, left_index, i - 1)

RETURN i - 1

The upper index is exclusive

right_index is not included in comparisons

Exercise Question 2

How many comparisons will be performed by Quicksort if we always pick the median element as the pivot? You only need to consider the case when $n = 8$. You should draw a recursion tree and note how many comparisons are performed at each subproblem.

Considering $X_{i,j}$

- Space of all possible outcomes is Ω
 - A comparison happens (1)
 - Or it doesn't (0)
 - This is an **indicator variable**
- What is the expected value of $X_{i,j}$ ($E[X_{i,j}]$)?
 - We need to know the probability of a comparison

$$p(X_{i,j} = 1)$$

Z_i, Z_j

Z_5	Z_4	Z_1	Z_8	Z_7	Z_2	Z_6	Z_3
51	43	17	83	79	23	61	37

What is the probability that Z_3 (37) and Z_7 (79) are compared?

Probability that Z_i, Z_j get compared

$$p(X_{i,j} = 1)$$

Consider any $Z_i, Z_{i+1}, \dots, Z_{j-1}, Z_j$ from the array

- Remember that these are not contiguous in the array, they are numbers in increasing order

What can you tell me about this group of numbers? (Hint: consider different values for the pivot element)

If none of these are chosen as a pivot, all are passed to the same recursive call.

Probability that Z_i , Z_j get compared

$$p(X_{i,j} = 1)$$

Consider any $Z_i, Z_{i+1}, \dots, Z_{j-1}, Z_j$ from the array

Among these values, consider the first one that gets chosen

1. If Z_i or Z_j are chosen first, then Z_i and Z_j are compared.
2. If one of Z_{i+1}, \dots, Z_{j-1} is chosen, then Z_i and Z_j are **NEVER** compared.

Why?

1. If Z_i or Z_j is chosen, then they become a pivot and the two values get compared
2. If a value in the middle gets chosen, then they go to separate calls

Z_i, Z_j

What is the probability that Z_3 (37) and Z_7 (79) are compared?

Z_5	Z_4	Z_1	Z_8	Z_7	Z_2	Z_6	Z_3
51	43	17	83	79	23	61	37

Each recursive call we have three options for Z_3 and Z_7 :

1. One is selected as pivot, and they are compared. ($X_{3,7} = 1$)
2. An item between them is selected and they are split apart. ($X_{3,7} = 0$)
3. An item outside their range is selected and they are partitioned together.

$$7 - 3 + 1$$

$$\frac{2}{5} = 40\%$$

Probability that z_i, z_j get compared

$$p(X_{i,j} = 1) = \frac{2}{\text{total \# of relevant choices}} = \frac{2}{j - i + 1}$$

- What does this mean for two values that are close to each other?

High Prob

- What does this mean for two values that are far from each other?

Low Prob

Counting the total number of comparisons

What is the equation for the **total** number of comparisons?

$$T = \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{i,j}$$

Every possible comparison

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{i,j}]$$

Linearity of expectations

Counting the total number of comparisons

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{i,j}]$$

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n p(X_{i,j} = 1) \cdot 1 + p(X_{i,j} = 0) \cdot 0$$

Counting the total number of comparisons

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{i,j}]$$

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n p(X_{i,j} = 1) \cdot 1 + p(X_{i,j} = 0) \cdot 0$$

Counting the total number of comparisons

$$p(X_{i,j} = 1) = \frac{2}{\text{total \# of relevant choices}} = \frac{2}{j - i + 1}$$

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n p(X_{i,j} = 1) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j - i + 1}$$

Simplifying the Inner Summation

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

Consider a fixed value for i (i=1)

$$\sum_{j=i+1}^n \frac{2}{j-i+1} = \sum_{j=2}^n \frac{2}{j} = 2 \cdot \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1+1} \right)$$

Consider another fixed value for i (i=5)

$$\sum_{j=i+1}^n \frac{2}{j-i+1} = \sum_{j=6}^n \frac{2}{j-4} = 2 \cdot \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-5+1} \right)$$

The inner summation
is maximized with i=1

Counting the total number of comparisons

$$p(X_{i,j} = 1) = \frac{2}{\text{total \# of relevant choices}} = \frac{2}{j - i + 1}$$

$$E[T] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n p(X_{i,j} = 1) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j - i + 1}$$

$$E[T] \leq 2 \sum_{i=1}^{n-1} \sum_{j=2}^n \frac{1}{j - 1 + 1}$$

Summations no longer depends on i

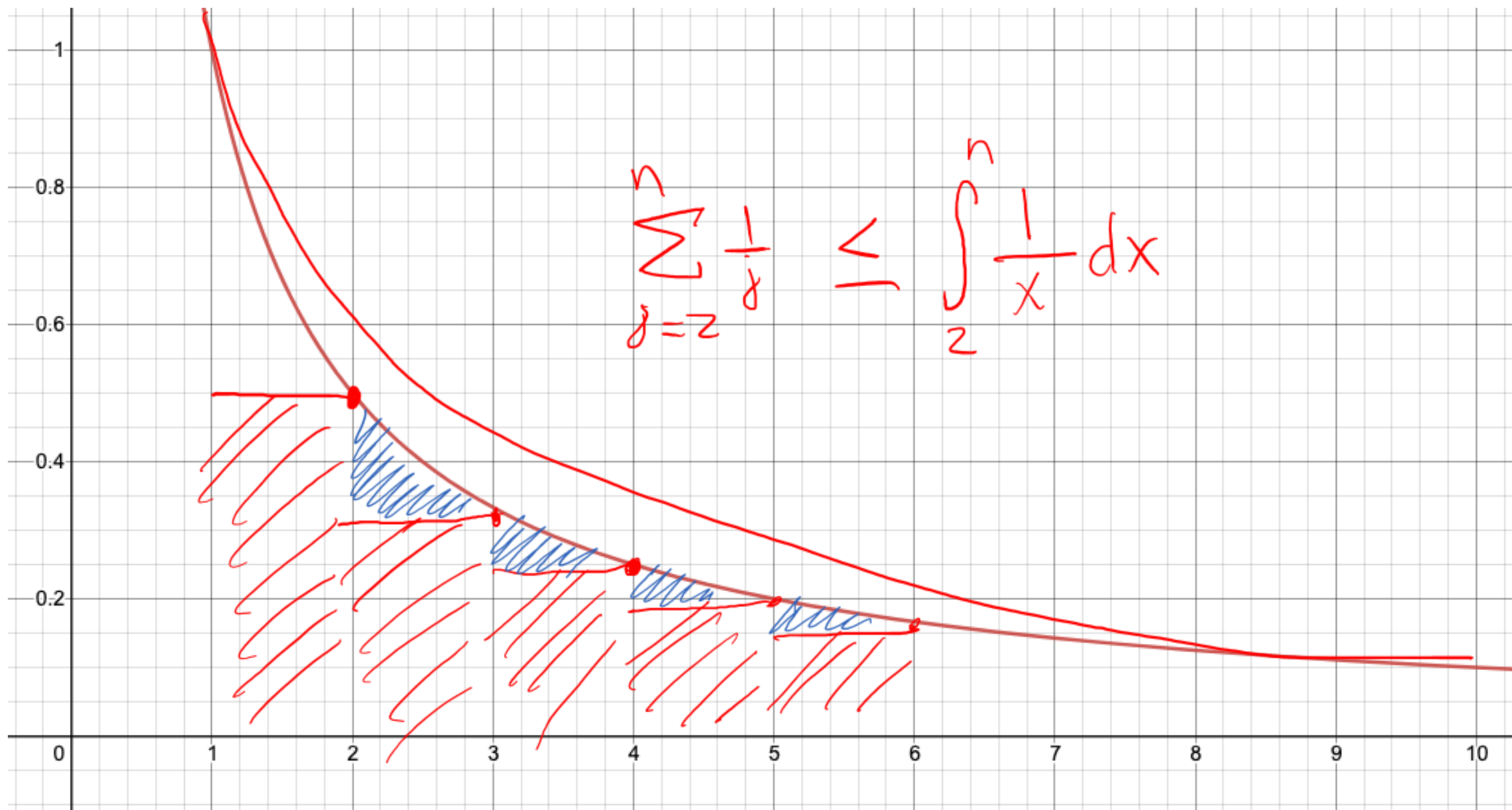
Simplify by turning this into an inequality and taking the value for i that results in the biggest number

Counting the total number of comparisons

$$E[T] \leq 2 \sum_{i=1}^{n-1} \sum_{j=2}^n \frac{1}{j - 1 + 1}$$

Summations no
longer depends on i

$$E[T] \leq 2n \sum_{j=2}^n \frac{1}{j}$$



Counting the total number of comparisons

$$E[T] \leq 2 \sum_{i=1}^{n-1} \sum_{j=2}^n \frac{1}{j - \textcolor{green}{1} + 1}$$

Summations no longer depends on i

$$\Theta(n \lg(n))$$


$$E[T] \leq 2n \sum_{j=2}^n \frac{1}{j}$$

$$E[T] = O(n^2)$$

$$E[T] \leq 2n \int_2^n \frac{1}{x} dx = 2n \ln(x) \Big|_2^n = 2n(\ln(n) - \ln(2)) \leq 2n \ln(n) = \textcolor{green}{O(n \lg(n))}$$

Change of base for logarithms

Summary

$$E[T] \leq O(n \lg(n))$$


- The expected number of comparisons is $O(n \lg n)$
- The expected number of comparisons is directly proportional to the total running time of Quicksort
- The **average** asymptotic running time of Quicksort is $O(n \lg n)$
- **Theorem:** For every input of the array of length n , the **average** running time of Quicksort with random pivots is $O(n \lg n)$.