CS101-Quiz4-Review

Suting Chen

CS101-Quiz4-Review

Key Points

1. Quick Sort

2. Master Theorem

Quick Sort

Key Points

1. Divide-and-Conquer
2. Good average case
3. In-place
4. NOT stable

```
QuickSort(A, p, r)
  if p < r
    q = Partition(A, p, r)
    QuickSort(A, p, q - 1)
    QuickSort(A, q + 1, r)

Partition(A, p, r)
  x = A[r]
  i = p - 1
  for j = p to r - 1
    if A[j] ⩽ x
      i = i + 1
      swap(A[i], A[j])
  swap(A[i + 1], A[j])
  return i + 1
```

Quick Sort

Time complexity Analysis

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

Quick Sort

Time complexity Analysis — Best case

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

In the best case, we (~~magically~~) choose the median as the pivot in $\Theta(1)$ time.

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

We can easily see that:

$$T(n) = \Theta(n \log n)$$

Quick Sort

Time complexity Analysis — Worst case

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

In the worst case, we keep partitioning n elements into $n - 1$ and 0.

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

We can easily see that:

$$T(n) = \Theta(n^2)$$

Quick Sort

Time complexity Analysis — Average case

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + cn$$

Quick Sort

Time complexity Analysis — Average case

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

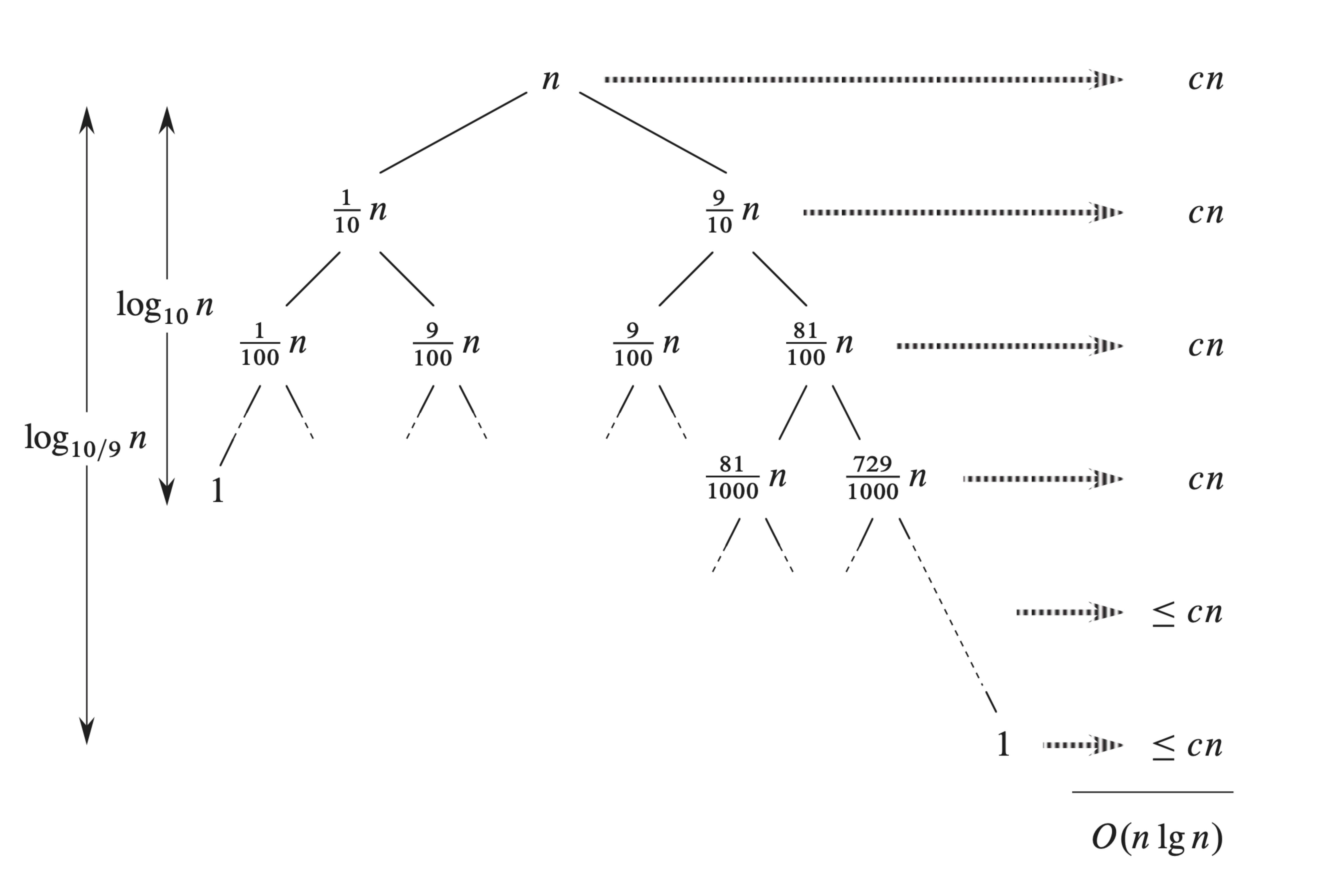
Assume we have a “bad” partition strategy, generating 9-to-1 split.

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + cn$$

Quick Sort

Time complexity Analysis — Average case

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + cn$$



Quick Sort

Time complexity Analysis — Average case

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

Assume we have a “bad” partition strategy, generating 9-to-1 split.

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + cn$$

We still have a rather good overall time complexity:

$$\Theta(n \log n)$$

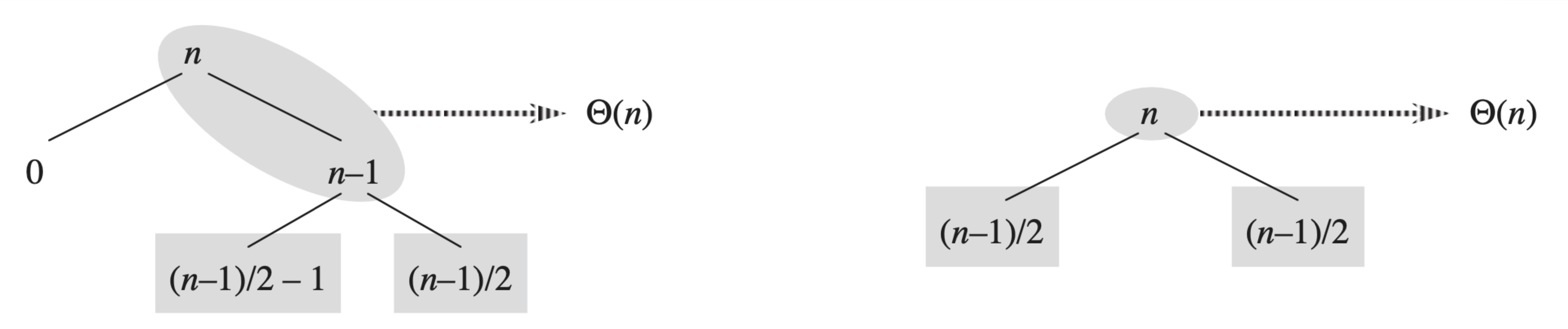
Quick Sort

Time complexity Analysis — Average case

Best Case	Average Case	Worst Case
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

1. Partitioning produces a mix of “good” and “bad” splits. (randomly distributed)

2. We assume “good” splits are optimal, and “bad” are worst-case scenarios.



3. Most of cases (actually about 80 %), a partition is more balanced than 9-to-1.

Quick Sort

Space complexity Analysis

Best Case	Average Case	Worst Case
$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(n)$

Remember function call stack!

Quick Sort

Possible optimization

Based on choosing a better pivot

1. Random sampling for pivot
2. Median-of-three

Quick Sort

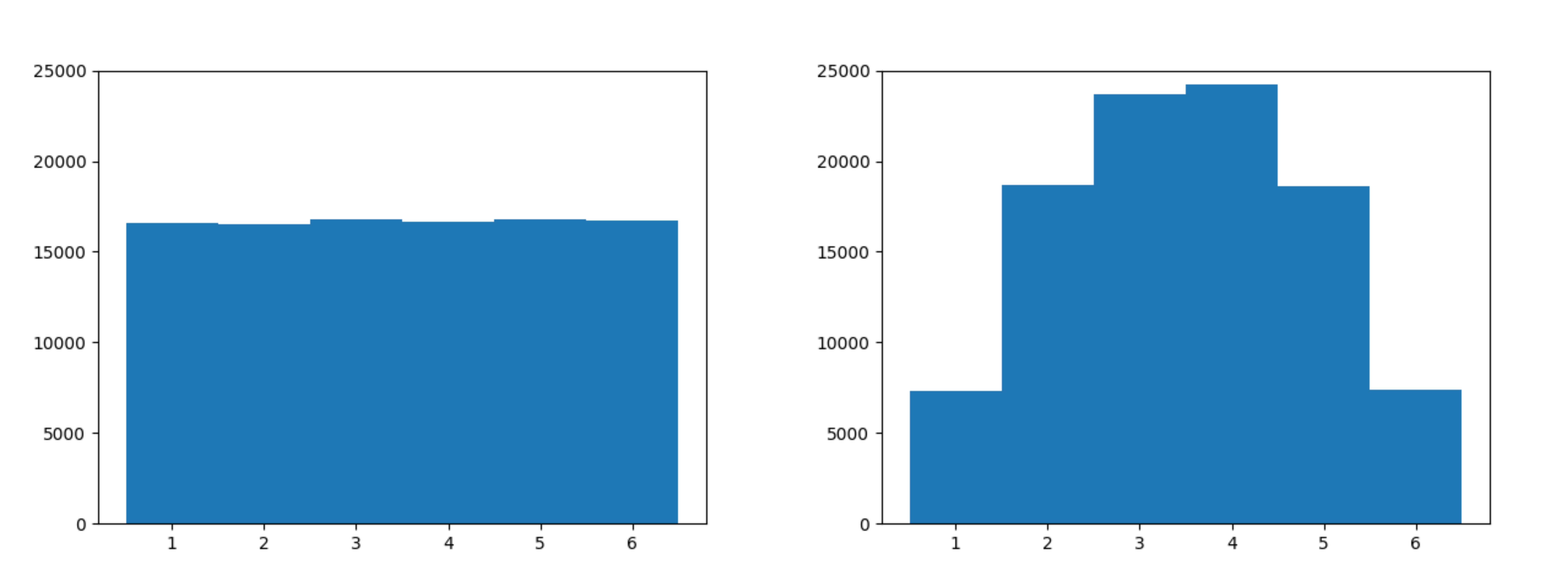
Random sampling for pivot

1. For arrays generated with iid random variables, it makes NO difference.
2. However, it is unacceptable to sort a nearly-sorted array in $\Theta(n^2)$ time.
3. Make the algorithm less vulnerable to attack. ([Link])

Quick Sort

Median-of-three

Gives a better possibility of choosing a pivot “closer” to the median.



Quick Sort

Median-of-three

- We name a pivot “good” if the pivot is located in the 25th ~ 75th percentile of the array.
- What is the possibility for us to get a “good” pivot with median-of-three?

CS101-Quiz4-Review

Key Points

1. Quick Sort
2. **Master Theorem**

Master Theorem

Definition

Given constants $a \geq 1$, $b > 1$, function $f(n)$, asymptotically positive function $T(n)$:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

1. If for a constant $\epsilon > 0$, $f(n) = O\left(n^{\log_b a - \epsilon}\right)$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.

2. If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.

3. If for a constant $\epsilon > 0$, $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$; for a constant $c < 1$ and sufficiently large n, $af\left(\frac{n}{b}\right) \leq cf(n)$, then $T(n) = \Theta\left(f(n)\right)$.

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$		

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	**Binary Search**

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	**Binary Search**
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$		

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	Binary Search
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$	Merge Sort

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	**Binary Search**
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$	**Merge Sort**
$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$		

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	**Binary Search**
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$	**Merge Sort**
$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$	$\Theta\left(n^{\log_2 7}\right)$	**Strassen**

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	**Binary Search**
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$	**Merge Sort**
$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$	$\Theta\left(n^{\log_2 7}\right)$	**Strassen**
$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$		

Master Theorem

Exercise

$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	**Binary Search**
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$	**Merge Sort**
$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$	$\Theta\left(n^{\log_2 7}\right)$	**Strassen**
$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	-	**Not applicable**

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1
$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$	

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1
$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$	$\boxed{\forall \epsilon > 0, \frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{\log_2 2}} = \frac{1}{\log n} < n^{\epsilon}}$ ✗

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1
$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$	Non-polynomial difference
$T(n) = 2T\left(\frac{n}{2}\right) + n\cos n$	

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1
$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$	Non-polynomial difference
$T(n) = 2T\left(\frac{n}{2}\right) + n \cos n$	~~No regularity~~, not positive

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1
$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$	Non-polynomial difference
$T(n) = 2T\left(\frac{n}{2}\right) + n\cos n$	~~No regularity~~, not positive
$T(n) = 2^nT\left(\frac{n}{2}\right) + n^n$	

Master Theorem

Exercise for Not Applicable

$T(n) = 0.5T\left(\frac{n}{2}\right) + O(1)$	0.5 < 1
$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$	Non-polynomial difference
$T(n) = 2T\left(\frac{n}{2}\right) + n\cos n$	~~No regularity~~, not positive
$T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$	Not constant

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$$

$$T(n) = \begin{cases} \Theta\left(n^d\right) & d > \log_b a \\ \Theta\left(n^d \log n\right) & d = \log_b a \\ \Theta\left(n^{\log_b a}\right) & d < \log_b a \end{cases}$$