CS101-Quiz4-Review

Suting Chen

CS101-Quiz4-Review

Key Points

1. Quick Sort
2. Master Theorem

省流定义：partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot

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
$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$

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

算法导论原话：The average-case running time of quicksort is much closer to the best case than to the worst case

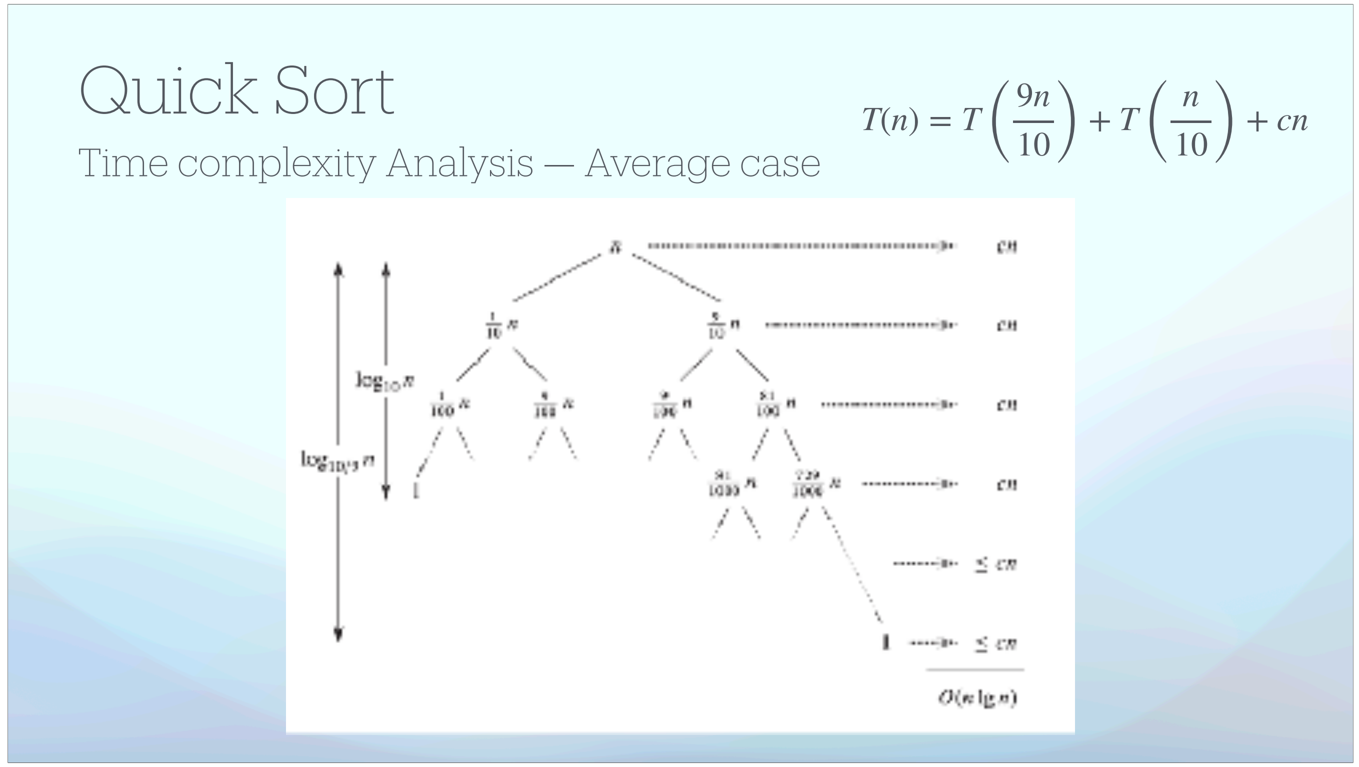
Quick Sort

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.

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



针对右上角的递归树证明
算法导论在4.4同样提供了代入法证明，因为上次讲过了，这次换换。

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:

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

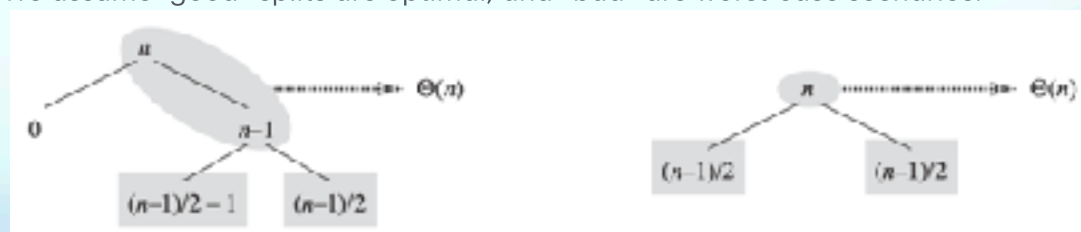
Quick Sort

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.

这里给一个符合直觉的说明，在7.2。严密的证明在算法导论7.4.2。时间关系不讲。
这张图的意思是：一个bad后面会跟一个good，这样一来这两个步骤就形成了三个子问题。这保证了算法整体是nlogn的。

Quick Sort

Space complexity Analysis

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

Remember function call stack!

空间复杂度来自于函数调用，对于正常的实现（比如标准库）。这种简单的功能O(N)的调用层数无法接受。这也是这种实现下的快排的致命缺点。

Quick Sort

Possible optimization

Based on choosing a better pivot

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

这两种课上提到的都是希望选到一个更接近中位数的pivot

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)

算法导论默认写法是选择第一个作为pivot

那个link是https://www.cs.dartmouth.edu/~doug/mdmspe.pdf。事实上，只要攻击够强，所有quicksort都会被打成O(N^2)。

但是，课件里挖了个坑，课件说的是最左面，中间，右面三选一。没有随机化的pivot-choosing一定是失败的。
下面的图是扔1～6的骰子，左面是一次，右面是Median-of-three。显然更靠近中间。

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?

思考题，有点太数学了，考试应该不会出。

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**

Strassen algorithm是矩阵相乘的东西，课件有

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

因为0.5<1

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

这里划掉单调性，因为上课没讲

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
$T(n) = 2^nT\left(\frac{n}{2}\right) + n^n$	Not constant

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

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