Design and Analysis of Algorithms

CSE 5311
Midterm Exam Practice

Junzhou Huang, Ph.D.
Department of Computer Science and Engineering
Midterm Exam

• Covered Contents
  – Chapter 1-3, Introduction & Growth Functions
  – Chapter 4 Divide-and-Conquer & Master Theorem
  – Chapter 30, FFT
  – Chapter 6-7, Heapsort & Quicksort
  – Chapter 8, Sorting in Linear Time
  – Chapter 9, Median & Order Statistics
  – Chapter 12-13, Binary Search Tree & Red-black Tree

• Questions
  – PART I. False and True (10 x 1pts=10pts)
  – PART II. Multiple Choices (10 x 3pts=30 pts)
  – PART III. Algorithm Questions (6 x 10pts=60pts)
How to review

• Questions
  – 60% of questions come from the questions in Homework Assignments

• How
  – Homework Assignments
  – Lectures
  – Textbooks
Practices

• True and False

– Binary insertion sorting (insertion sort that uses binary search to find each insertion point) requires $O(n \log n)$ total operations.

Solution: False. While binary insertion sorting improves the time it takes to find the right position for the next element being inserted, it may still take $O(n)$ time to perform the swaps necessary to shift it into place. This results in an $O(n^2)$ running time, the same as that of insertion sort.

– In a BST, we can find the next smallest element to a given element in $O(1)$ time.

Solution: False. Finding the next smallest element, the predecessor, may require traveling down the height of the tree, making the running time $O(h)$. 
Multiple Choices

• You are running a library catalog. You know that the books in your collection are almost in sorted ascending order by title, with the exception of one book which is in the wrong place. You want the catalog to be completely sorted in ascending order.
  – (a) Insertion Sort
  – (b) Merge Sort
  – (c) Radix Sort
  – (d) Heap Sort
  – (e) Counting Sort
Multiple Choices

• Which of the following complexity analysis is/are correct.
  – (a) The worst case running time for building a binary search tree is $O(n \lg n)$.
  – (b) The worst case running time for building a red-back tree is $O(n \lg n)$.
  – (c) The worst case running time for building a binary search tree is $O(n^2)$.
  – (d) The worst case running time for building a red-back tree is $O(n^2)$.

[b,c]
Algorithm Questions

- Give asymptotic upper and lower bound for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 2T(n/2) + n^4$
2. $T(n) = T(n-2) + n^2$
Algorithm Questions

- Give asymptotic upper and lower bound for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 2T(n/2) + n^4$
2. $T(n) = T(n-2) + n^2$
Master Theorem

The master method applies to recurrences of the form

\[ T(n) = a \cdot T(n/b) + f(n), \]

where constants \( a \geq 1 \), \( b > 1 \), and \( f \) is asymptotically positive function

1. \( f(n) = O(n^{\log_b a - \varepsilon}) \) for some constant \( \varepsilon > 0 \), then \( T(n) = \Theta(n^{\log_b a}) \)
2. \( f(n) = O(n^{\log_b a}) \) for some constant \( \varepsilon > 0 \), then \( T(n) = \Theta(n^{\log_b a \log n}) \)
3. \( f(n) = O(n^{\log_b a + \varepsilon}) \) for some constant \( \varepsilon > 0 \), and if \( a f(n/b) \leq c f(n) \) for some constant \( c < 1 \), then \( T(n) = \Theta(f(n)) \).
Algorithm Questions

1. \( T(n) = T(n-2) + n^2 \)

- Solution:

\[
T(n) = n^2 + T(n-2) = n^2 + (n-2)^2 + T(n-4)
\]

\[
= \sum_{i=0}^{n/2} (n - 2i)^2 = \Theta(n^3)
\]
Algorithm Questions

- \( T(n) = 2T(n/2) + n^4 \)

- Solution:

  \( a=2, \ b=2, \ n^{\log_b a} = n, \)

  \( f(n) = n^4 \)

  \( af(n/b) \leq cf(n) \) ??? \( 2(n/2)^4 = 1/16 \ast (n)^4 \)

  Case 3.

  \( T(n) = \Theta(n^4) \).