Missouri University of Science & Technology — Department of Computer Science

Spring 2024 — **CS 2500: Algorithms (Sec: 102)**

Exam 1: Sample Questions

Instructor: *Sid Nadendla* — **Email:** *nadendla@mst.edu*

Exam 1 will be designed as a 75-minutes long, in-class exam. You may have 3-4 questions to solve, depending on the size of the questions. This handout presents a few review questions for Exam 1 examination on **Thursday, Mar 7, 2024**.

Problem 1 Integer Multiplication

1. The problem is to compute the product of two n-bit input arrays A and B, where n is significantly larger than the register size in your processor (e.g. Consider the product of 256-bit arrays over a 64-bit processor). In such a case, it is infeasible to run the traditional method for finding the product of two arrays. How would you solve this problem if the size of the register $m = \frac{n}{k}$, for some positive integer k?

2.

Problem 2 Asymptotic Notation

1. Definitions for Θ, O, and Ω notations.

2. Prove that $T(n) = \frac{1}{2}n^2 - 3n = O(n^2)$. [Hint: You need to find two constants c and N_0 in the O-definition.]

3. Suppose $T(n) = a_k n^k + a_{k-1}n^{k-1} + \cdots + a_n + a_0$. Let k be a non-negative integer and $a_i \in \mathbb{R}$ for all $i = 1, \cdots, k$. Then, prove that $T(n) = O(n^k)$. [Ref. Topic 1.3, Proposition 2.1 in class notes]

4. Let f and g denote functions from positive integers to the non-negative real numbers. Let $T(n) = \max\{f(n), g(n)\}$, for all $n \geq 1$. Then, prove that $T(n) = \Theta(f(n) + g(n))$. [Ref. Topic 1.3, Proposition 2.5 in class notes]

Problem 3 Master Method

1. Using both recursion trees and Master method, prove that the recurrence relation $T(n) = 2f(n/2) + O(n)$ results in $T(n) = O(n \log n)$.

Problem 4 Divide & Conquer

1. Say, our goal is to find both maximum and minimum elements of a given input array A of size n efficiently. A naive approach is to find the maximum using $n - 1$ comparisons and then find the minimum element over the remaining entries using $n - 2$ comparisons. Therefore, this algorithm takes about $2n - 3$ comparisons. How can we reduce the number of comparisons by adopting a divide-and-conquer approach?

2.

Problem 5 Sorting Algorithms

1. Write the pseudocode for INSERTION-SORT and prove its correctness.

2. Given two sorted input arrays A and B, how can we merge them into a sorted array? Write the pseudocode for your approach and prove its correctness.

3. Given any input array A, how can we partition it into two subarrays with a pivot element such that every entry to the left of pivot is smaller than the pivot, and every entry to the right of pivot is larger than the pivot? Write the pseudocode for your approach and analyze its worst-case run-time.

4. Prove that the worst-case run time for HEAP-SORT is $\Theta(n \log n)$.

5. Demonstrate the various iterations of MERGE-SORT, QUICK-SORT and HEAP-SORT for a given input array $A = [3,\ 8,\ 2,\ 5,\ 1,\ 4,\ 7,\ 6]$.

Problem 6 Bounds on Comparison-Based Sorting

1. Prove that the largest lower bound on the runtime of comparison-based sorting is $\Omega(n \log n)$.