Missouri University of Science & Technology — Department of Computer Science
Spring 2022 — **CS 2500: Algorithms**

Sample Questions for Exam 1

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

Midterm 1 will be designed as a 1-hour 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 Midterm 1 examination on Mar 8, 2022.

Section 1 Asymptotic Notation

1. Definitions for Θ, O, Ω, o and ω notations.
2. Prove that $\displaystyle\sum_{i=1}^{n} i = \Theta(n^2)$.
3. If $f_1(n) = \Theta(g_1(n))$ and $f_2(n) = \Theta(g_2(n))$, prove that
$$f(n) = f_1(n) + f_2(n) = \Theta(g_1(n) + g_2(n)).$$

Section 2 Recursion

1. Using substitution method, prove that the recurrence relation $f(n) = 2f(n/2) + n$ results in $f(n) = O(n \log n)$.
2. Solve $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$ using recursion trees.

Section 3 Divide & Conquer

1. 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. If you were to adopt a divide-and-conquer approach, what is the order of improvement in terms of the number of comparisons?
2. 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?

Section 4   Comparison Sorting

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

4. If we were to employ divide and conquer approach to design a comparison sort, demonstrate various sorting algorithms when the pivot element q is always chosen to be

   (a) mid-point of the array in each , i.e. MERGE-SORT,

   (b) such that $A[1 : q-1] \leq A[q] \leq A[q+1 : n]$, i.e. QUICK-SORT,

   for a given input array $A = [1,\ 4,\ 7,\ 6,\ 3,\ 9,\ 5,\ 2,\ 4,\ 6]$.

Section 5   Sorting in Linear Time

1. Prove that the running time for comparison sorts is $\Omega(n \log n)$.

2. Demonstrate COUNTING-SORT for the input array $A = [1,\ 4,\ 2,\ 5,\ 3,\ 1,\ 2,\ 4,\ 3,\ 1,\ 4]$.