Missouri University of Science & Technology — Department of Computer Science
Spring 2024 — **CS 2500: Algorithms (Sec: 102)**

Homework 2b: Quick and Heap Sort

Instructor: *Sid Nadendla* — **Due:** *February 28, 2024*

Instructions

1. All solutions have to be written in **detail** and with **proper mathematical rigor**. Solutions can be handwritten, or typesetted using Word or LaTeX.
2. Python is the chosen programming language for this class. Due to the large class size, no other programming languages are accepted in order to guarantee timely feedback.
3. All files have to be submitted in the assigned Gitlab repositories. Any submissions outside Gitlab will be disregarded for grading purposes.
4. Problems are mostly assigned from the prescribed textbook: Tim Roughgarden, "Algorithms Illuminated [Omnibus Edition]", 2022.
5. You are allowed to discuss with fellow students, or use external sources such as blogs, YouTube videos, or AI tools (e.g. ChatGPT), if and only if you cite all your sources. In the case of AI tools, you should also report their query and the tool's response in verbatim. Also, your solution should describe how this information was used to develop their solution.

Problem 1 Partition Subroutine *1 point*

Problem 5.2 (Ref. Page 117 in the textbook)

Statement:

Recall the `Partition` subroutine employed by Quicksort. You are told that the following array has just been partitioned around some pivot element:

A = { 3, 1, 2, 4, 5, 8, 7, 6, 9 }.

Which of the elements could have been the pivot element? (List all that apply; there could be more than one possibility.)

Problem 2 Quick Sort *2 points*

Problem 5.8 - Part 1 (Ref. Page 118 in the textbook)

Statement:

Implement the `QuickSort` algorithm in Python, and evaluate its empirical performance when the pivot is always the first element in the array.

One approach is to keep track of the number of comparisons between input array elements made by `QuickSort`. For several different input arrays, determine and plot the number of comparisons as the input array grows in size.

Problem 3 Heap Sort *2 points*

Convert any input array A into your own `Min-Heap` data structure in Python. Then, sort the entries in A by implementing the `HeapSort` algorithm in Python, and evaluate its empirical performance for different input sizes.