الجامعة المصرية اليابانية للعلوم والتكنولوجيا E-JUST Egypt - Japan University of Science and Technology エジプト日本科学技術大学	**Assignment 5: Spring 2020**
Department	Computer Science and Engineering
Submission due	Tue, May 4th, 2021by 9:30am
Date	Apr 28th, 2021
Course Title	CSE 326: Analysis and Design of Algorithms
Instructor	Prof. Walid Gomaa
Allowed Equipment	None

Answer all of the following questions

1. The *median-of-three* heuristic examines the first, last, and middle element of the array, and uses the median of those three elements as the pivot of quicksort. Prove that quicksort with the median-of-three heuristic requires $\Omega(n^2)$ time to sort an array of size n in the worst-case. Specifically, for any integer n, describe a permutation of the integers 1 through n, such that in every recursive call to median-of-three-quicksort, the pivot is always the second smallest element of the array.

2. What is the running time of quicksort when all elements of the array have the same value?

3. Suppose that the splits at every level of quicksort are in proportion $1-\alpha$ to α, where $0 < \alpha \leq 1/2$ is a constant. Show that the minumum depth of a leaf in the recursion tree is approximately $-\log n/\log \alpha$ and the maximum depth is approximately $-\log n/\log(1-\alpha)$.

4. Suppose we are using quicksort to process data that we receive from a connection in a networked system. We want to cover our system from the possibility of being 'sabotaged' by hostile connections— we could receive data that is specifically crafted to cause quicksort to have its worst-case performance and thus make our system consume excessive resources and time (rendering it unable to efficiently respond to other connections). This can be considered as an adversarial attack.

   (a) Assuming that quicksort simply chooses the first element as the pivot (instead of the median of first, last, and middle), what is the arrangement of data that produces the worst-case performance in quicksort? (that is, if you were the attacker trying to sabotage the system, what data would you have to send?)

   (b) Suggest a simple strategy (hopefully requiring no more than linear time) to avoid the problem. That is, a strategy to guarantee that quick sort will run in $\mathcal{O}(n \log n)$ most of the time, regardless of input data, even if this input data is maliciously created. Notice that, you're not allowed to change the way quicksort selects the pivot (in fact, you will hopefully suggest a strategy that works regardless of how quicksort selects the pivot).

5. Consider the sorting problem where all the numbers are not known exactly. Instead, for each number, you know an interval on the real line to which it belongs. That is, you are given n closed intervals of the form $[a_i, b_i]$, where $a_i \leq b_i$. Assume that no interval contains any other interval. That is, if $a_i \leq a_j$ then $b_i \leq b_j$. You are asked to *fuzzy-sort* these intervals, i.e., produce a permutation $\langle i_1, \ldots, i_n \rangle$ of the intervals such that for all j, there exists a $c_j \in [a_{i_j}, b_{i_j}]$ satisfying $c_1 \leq c_2 \leq \cdots \leq c_n$.

   Suppose that each interval is guaranteed to overlap at least $d-1$ other intervals. Give an $\mathcal{O}(n \log \frac{n}{d})$ algorithm to fuzzy-sort n intervals with d degrees of overlap. Thus, if $d = \Theta(1)$, the algorithm runs in $\mathcal{O}(n \log n)$ time, and if $d = \Theta(n)$, the algorithm runs in $\mathcal{O}(n)$.