COMP 3711 – Design and Analysis of Algorithms
2022 Fall Semester – Written Assignment 2
Distributed: September 30, 2022
Due: October 17, 2022, 23:59

Your solution should contain
(i) your name, (ii) your student ID #, and (iii) your email address
at the top of its first page.

Some Notes:

- Please write clearly and briefly. In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.
- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page. ***You must acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- The term *Documented Pseudocode* means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.
- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.
- Please make a *copy* of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.
- Submit a SOFTCOPY of your assignment to Canvas by the deadline. If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

1. (10 points; from textbook) Let $A[1..n]$ be an array of n distinct integers. For any $i, j \in [1, n]$ such that $i < j$, if $A[i] > A[j]$, we call the pair (i, j) an inversion. Suppose that we select a permutation of A uniformly at random. Derive the expected number of inversions in A in this random permutation? Use indicator random variables.

2. (10 points; from textbook) Let $\textsc{Random}(1, k)$ be a procedure that draws an integer uniformly at random from $[1, k]$ and returns it. We assume that a call of $\textsc{Random}$ takes $O(1)$ worst-case time. The following recursive algorithm $\textsc{Random-Sample}$ generates a random subset of $[1, n]$ with $m \leq n$ distinct elements. Prove that $\textsc{Random-Sample}$ returns a subset of $[1, n]$ of size m drawn uniformly at random.

```
RANDOM-SAMPLE(m, n)
  if m = 0 then
    return ∅
  else
    S ← RANDOM-SAMPLE(m − 1, n − 1)
    i ← RANDOM(1, n)
    if i ∈ S then
      return S = S ∪ {n}
    else
      return S = S ∪ {i}
    end if
    return S
  end if
```

3. (10 points) Let $A[1..n]$ be an array of n distinct integers. This problem is about rearranging the elements of A so that it becomes an array representation of a binary min-heap. Assume that n is one less than a power of 2. Recall that A can be viewed as an array representation of a binary tree with $A[1]$ being the root at level 0. The children of the root is at level 1, and so on. For all $i \in [1, n]$, we use $\ell(i)$ to denote the level of $A[i]$. Let $L = \max\{\ell(i) : i \in [1, n]\}$. The height of $A[i]$ is defined as $L - \ell(i)$. So the root has height L and its children has height $L - 1$.

   (a) Let h be any integer in the range $[0, L]$. Derive the indices of the elements of A that have height h. Derive the number of the elements of A that have height h.

   (b) Consider an element $A[i]$. Suppose that the subtrees rooted at the children of $A[i]$ are already binary min-heaps. Describe how you can update the subtree rooted at $A[i]$ so that this subtree becomes a binary min-heap in time bounded by the height of $A[i]$.

   (c) Use (b) to design a recursive algorithm that turns $A[1..n]$ into a binary min-heap. Explain the correctness of your algorithm. Show that the running time of your algorithm is $O(n)$.

4. (10 points) You are given a list of n intervals $I_1, I_2, \ldots, I_n$ on the real line, Each I_j is denoted by $[s_j, e_j]$, where $s_j, e_j \in \mathbb{R}$ such that $s_j < e_j$. That is, s_j and e_j are the left and right endpoints of I_j, respectively. A time instance $t \in \mathbb{R}$ *hits* an interval I_j if $t \in [s_j, e_j]$. Note that a single time instance t can hit several intervals as long as t belongs to every one of them.

   Describe a greedy algorithm that finds a smallest set of time instances that hit all n intervals $I_1, I_2, \ldots, I_n$. Explain the correctness of your algorithm. Derive the running time of your algorithm.

5. (10 points) Let $A_1, \ldots, A_n$ be n computer jobs such that A_i takes t_i time units to finish. Assume that if $i \neq j$, then $t_i \neq t_j$. Let σ be a permuation of $1, 2, \ldots n$, that is, $\sigma(i)$ is an integer in $[1, n]$ and if $i \neq j$, then $\sigma(i) \neq \sigma(j)$. Suppose that we run these n jobs on a single CPU machine one after another starting at time zero in this order $A_{\sigma(1)}, A_{\sigma(2)}, \ldots, A_{\sigma(n)}$.

   (a) For every $j \in [1, n]$, the completion time of $A_{\sigma(j)}$ is the time at which $A_{\sigma(j)}$ completes. Express the completion time of $A_{\sigma(j)}$ in terms of $t_{\sigma(i)}$ for $i \in [1, j]$.

   (b) The total completion time is the sum of the completion times of $A_{\sigma(1)}, A_{\sigma(2)}, \ldots, A_{\sigma(n)}$. Show that the total completion time is minimized if $t_{\sigma(1)} < t_{\sigma(2)} < \cdots < t_{\sigma(n)}$.