PSET 1 Due on **Sunday September 01, 2024 at 11:59 pm ET**

CS 601: Algorithms and Complexity, Fall 2024 (Akash Kumar) Version: **1.0**

Submitted by <**name**>

Grading Policies:

(a) **Total points: 100**

(b) There are four problems in this pset (each of which have one or more subparts). Please solve/attempt them all.

(c) You are allowed –indeed you are encouraged– to, collaborate with other students in the class. Please list all the people you collaborated with.

(d) You may use online resources as well.

(e) I cannot emphasize this point enough. **With all the freedom given in the points above, it is imperative that you write your solutions yourself.** Please do not copy/paste what your friends wrote or what electronic material you find online. This defeats the purpose of solving a problem set. Indeed, **if you are caught engaging in unseemly practices, severe action will be taken.**

(f) Please write your pseudocode as you see those written in textbooks. Confusing/inscrutable psuedocodes will be penalized with no room for a regrade.

(g) Please write your mathematical claims rigorously and unambiguously. Again, follow the style laid out in textbook or in the class if you are unsure. Ambiguous or unclear claims/proofs will be penalized. Your TAs are not here to debug your proofs/algorithms. If your writeup does not convince them of the correctness, you will rightfully lose points.

---

1 (20 PTS.) UNDERSTANDING BIG-OH

1.A. (10 PTS PTS.) Arrange the following functions in increasing order according to big-Oh. That is, you will report the function which has the smallest growth rate as the first item in your list, followed by the function with second smallest growth rate and so on. Do not justify your answer.

(a) $a(n) = 10^n$.
(b) $b(n) = n^{1/5}$.
(c) $c(n) = \sqrt{\log n}$.
(d) $d(n) = \log \log n$.
(e) $e(n) = \sqrt{n}$.
(f) $f(n) = 2^{\sqrt{\log n}}$.
(g) $g(n) = n^{1/\log\log n}$.
(h) $h(n) = 2^n$.
(i) $i(n) = n$.
(j) $j(n) = 2^{n^2}$.

1.B. (10 PTS PTS.) In the following problem, you are given two functions in each row of the table denoted f and g. Indicate whether $f = O(g)$, $f = \Omega(g)$ or $f = \Theta(g)$.

	f(n)	g(n)
(i)	$\log_2 n$	$\log_{10} n$
(ii)	$n^{1/3}$	$(\log_2 n)^2$
(iii)	n^2	2^{n^2}
(iv)	2^n	$2^{\sqrt{n}}$
(v)	$2^{\sqrt{\log n}}$	$\log \log n$

Solution:

1.A.

1.B.

2 (30 PTS.) USING INDUCTION

2.A. (5 PTS.) $2^{2n} - 1$ is divisible by 3, for integers $n > 0$.

2.B. (10 PTS.) Consider the following sequence:

$$A_1 = 1, A_2 = 1, A_3 = 1$$
$$A_n = A_{n-1} + A_{n-2} + A_{n-3}$$

Prove that $\forall n \in \mathbb{Z}^+$, it is the case that $A_n < 2^n$.

2.C. (15 PTS.) Let $F_0 = 0, F_1 = 1, F_2 = 1, \ldots$ denote the Fibonacci Sequence. Show that $F_1^2 + F_2^2 + F_3^2 + \ldots + F_n^2 = F_{n+1} \cdot F_n$. For this problem, I will even allow solutions that do not use induction.

Solution:

2.A.

2.B.

2.C.

3 (30 PTS.) THE BROKERAGE FIRM

A brokerage firm periodically looks at how a particular stock has done in the last n days and wishes to know what the maximum possible profit would have been if someone had bought and sold a share during those n days. Let A be an array of n numbers where $A[i]$ represents the value of a share of the stock on day i. We wish to find $\max_{1 \leqslant j < k \leqslant n} A[k] - A[j]$, where $A[k] - A[j]$ represents the profit made when buying on day j and selling on day k.

3.A. (10 PTS.) Describe an $O(n^2)$ time algorithm.

3.B. (20 PTS.) Describe an $O(n)$ time divide-and-conquer algorithm. Carefully state what each recursive call returns and how this information is used. Give the recurrence relation underlying the algorithm.

Solution:

3.A.

3.B.

4 (20 PTS.) SIGNIFICANT INVERSION PROBLEM

Recall the problem of counting the number of inversions. Given a sequence of n distinct numbers $a_1, a_2, \ldots a_n$ and we define an inversion to be a pair $i < j$ such that $a_i > a_j$. Let's call a pair a *significant inversion* if $i < j$ and $a_i > 2a_j$

4.A. (10 PTS.) Show that the number of significant inversions of the form $i \leqslant n/2 < j$ in the list $L = a_1, a_2, \ldots a_n$ is equal to the number of regular inversions of the form $i \leqslant n/2 < j$ in the list $L_0 = a_1, a_2 a_{n/2}, 2 \cdot a_{n/2+1} \ldots, 2 \cdot a_n$.

4.B. (10 PTS.) Give an $O(nlogn)$ algorithm to count the number of significant inversions in a list.

Solution:

4.A.

4.B.