# CSE 431/531 Analysis of Algorithms
# Problem Set 1

Chen Xu

Due Date: June 15, 2023 23:59 PM EST.

**Problem 1. (20%)**
Given the following ten functions:

1. $f_1(n) = n^2$
2. $f_2(n) = n \log n$
3. $f_3(n) = n^3$
4. $f_4(n) = 2^n$
5. $f_5(n) = n!$
6. $f_6(n) = \sqrt{n}$
7. $f_7(n) = \log n$
8. $f_8(n) = n^{2.5}$
9. $f_9(n) = 3^n$
10. $f_{10}(n) = n^{\log n}$

Arrange these functions in ascending order of growth rate, that is, the slowest-growing function first and the fastest-growing function last. Provide a detailed comparison and justification for your arrangement.

**Problem 2. (20%)**

(1) Prove that $O(an^3 + bn^2 + cn + d) = O(n^3)$ for any positive $a$.

(2) Prove that $O(g(n) + h(n)) = O(\max(g(n), h(n)))$ for arbitrary non-negative functions $g(n)$ and $h(n)$. Provide a detailed proof and explanation. Note: $max$ is taking the bigger one of $f(n), g(n)$ for any arbitrary $n$.

**Problem 3. (20%)**
Here is an implementation of an unknown algorithm "Heh".

```
function Heh(A[0..N-1], value) {
    l = 0
    h = N - 1
    while (l <= h) {
        m = l + ((h - l) / 2)
        if (A[m] > value)
            h = m - 1
        else if (A[m] < value)
            l = m + 1
        else
            return m
    }
    return "Heh"
}
```

(1) Rewrite this iterative version into the recursive version.

(2) Write the recursive function for the running time $T(n)$.

(3) Solve the recursion you gave in (2).

**Problem 4. (20%)**
Solve the following recurrence relations:

(1) $T_1(n) = 100T_1\left(\frac{n}{9}\right) + n^2$

(2) $T_2(n) = 4T_2\left(\frac{n}{2}\right) + n^3$

(3) $T_3(n) = 5T_3\left(\frac{n}{2}\right) + n^{\log_2 5}$

(4) $T_4(n) = T_4\left(\frac{n}{4}\right) + n!$

**Problem 5. (20%)**
An array $A = \{a_1, a_2, ..., a_n\}$ is called a "Mountain" if there is a $k(1 \leq k \leq n)$ such that $a_1 \leq a_2 \leq ... \leq a_k$ and $a_n \leq a_{n-1} \leq ... \leq a_k$. The $a_k$ is called the "summit". A median of the array is the middle element in the sorted sequence of the array. For simplicity let us assume our "Mountain" has odd number of elements.

(1) Design an algorithm to find the median of a "Mountain". (2) Prove your algorithm works. (3) Analyze the running time.