# CS182 Homework # 5

Maninder (Kaurman) Kaur

February 2025

## Problem 1

I, **Maninder (Kaurman) Kaur**, affirm that I have not given or received any unauthorized help on this assignment and that this work is my own. What I have submitted is expressed and explained in my own words. I have not used any online websites that provide a solution. I will not post any parts of this problem set to any online platform and doing so is a violation of course policy.

## Problem 2 ($4 \times 6$ points)

For each of the following statements, state whether they are true or false and explain briefly.

1. $n \log^2 n \in O(n\sqrt{n})$
2. $2025n^3 \in O(n^4)$
3. $\log n^{6n} \in O(n \log \log n)$
4. $n^3 - 2000n^2 - 1000n \in O(n^2)$
5. $1000n^3 + 100n^2 + 10n \in O(n^3 \log n)$
6. $\sqrt{n} \in O(0.999^n)$

**Solution**

1. **Statement:** $n \log^2 n \in O(n\sqrt{n})$. **Answer:** True. **Explanation:**
   (a) Compare growth rates: $\log^2 n$ vs. $\sqrt{n} = n^{1/2}$.
   (b) Logarithmic terms grow slower than polynomial terms.
   (c) Thus, $n \log^2 n$ is dominated by $n\sqrt{n}$.
2. **Statement:** $2025n^3 \in O(n^4)$. **Answer:** True. **Explanation:**
   (a) $n^3$ grows slower than $n^4$.
   (b) Constants like 2025 are irrelevant in asymptotic notation.
   (c) $2025n^3 \leq cn^4$ for some constant $c$.
3. **Statement:** $\log n^{6n} \in O(n \log \log n)$. **Answer:** False. **Explanation:**
   (a) Simplify: $\log n^{6n} = 6n \log n$.
   (b) Compare $6n \log n$ vs. $n \log \log n$.
   (c) $\log n$ grows faster than $\log \log n$, so the statement is false.
4. **Statement:** $n^3 - 2000n^2 - 1000n \in O(n^2)$. **Answer:** False. **Explanation:**
   (a) Dominant term: $n^3$.
   (b) $n^3$ is not bounded by $O(n^2)$.
5. **Statement:** $1000n^3 + 100n^2 + 10n \in O(n^3 \log n)$. **Answer:** True. **Explanation:**
   (a) Leading term: $1000n^3$.
   (b) $n^3 \log n$ grows faster than $n^3$.
   (c) Thus, $1000n^3 \leq cn^3 \log n$ for $n \geq 2$.

6. **Statement:** $\sqrt{n} \in O(0.999^n)$. **Answer:** False. **Explanation:**
   (a) $0.999^n$ decays exponentially.
   (b) Polynomial growth $\sqrt{n}$ cannot be bounded by an exponential decay.

# Problem 3 (6 × 4 points)

Determine how many times the BoilerUp() function is called in the following code snippets. Provide an exact count and a tight asymptotic bound in Big-O notation.

**Solution**

## (a) Function A

```
function A(n, m)
    for i = 0; i < n; i = i + 1 do
        BoilerUp()
        for j = 0; j < m; j = j + 1 do
            BoilerUp()
            BoilerUp()
        end for
    end for
end function
```

**Exact Count:**

$$n \times (1 + 2m) = n + 2nm$$

**Asymptotic Bound:** $\mathcal{O}(nm)$. **Explanation:**

1. Outer loop runs $n$ times.
2. Each iteration: 1 call outside inner loop.
3. Inner loop runs $m$ times, with 2 calls per iteration.

## (b) Function B

```
function B(n)
    for i = 1; i < n; i = i * 2 do
        for j = 1; j < n; j = j * 2 do
            BoilerUp()
        end for
    end for
    for i = n; i >= 0; i = i - 1 do
        BoilerUp()
    end for
end function
```

**Exact Count:**

$$(\log_2 n)^2 + (n + 1)$$

**Asymptotic Bound:** $\mathcal{O}(n)$. **Explanation:**

1. First nested loops: $i$ and $j$ double until $n$, resulting in $\log_2 n$ iterations each. Total: $(\log n)^2$.
2. Second loop: $n + 1$ iterations.

## (c) Function C

```
function C(n)
    i = 1
    while i <= n^2 do
        for j = 0; j < i; j = j + 1 do
            BoilerUp()
        end for
        i = i * 2
    end while
end function
```

**Exact Count:**

$$\sum_{k=0}^{\log_2(n^2)} 2^k = 2n^2 - 1$$

**Asymptotic Bound:** $\mathcal{O}(n^2)$. **Explanation:**

1. $i$ starts at 1 and doubles until $n^2$.
2. Total iterations: $\log_2(n^2) + 1$.
3. Sum of geometric series: $1 + 2 + 4 + \cdots + n^2 = 2n^2 - 1$.

## (d) Function D

```
function D(n)
    for i = n; i > 0; i = floor(i/2) do
        for j = i; j < n; j = j + 1 do
            BoilerUp()
        end for
    end for
end function
```

**Exact Count:**

$$\sum_{k=0}^{\lfloor \log_2 n \rfloor} (n - 2^k) \approx n \log n - n$$

**Asymptotic Bound:** $\mathcal{O}(n \log n)$. **Explanation:**

1. Outer loop: $\log_2 n$ iterations.
2. Inner loop: $n - i$ iterations per outer loop.

# Problem 4 (8 + 12 points)

## (a)

Show that $p(n) = n^3 + 4n^2 - 3n + 10n \log n$ is $O(n^3)$ and $O(n^4)$.

## (b)

Define $q(n) = 2^{\log_2(n^3)}$. Let $r(n) = p(n) - q(n)$. Prove or disprove that $r(n)$ is $O(n)$. **Solution**

## (a)

Show that $p(n) = n^3 + 4n^2 - 3n + 10n \log n$ is $O(n^3)$ and $O(n^4)$.

**Proof for $O(n^3)$:**

1. Dominant term: $n^3$.
2. All other terms ($4n^2$, $-3n$, $10n \log n$) are asymptotically smaller.
3. For large $n$, $p(n) \leq 2n^3$.

**Proof for $O(n^4)$:**

1. $n^3 \leq n^4$ for $n \geq 1$.
2. Thus, $p(n) \leq n^4 + 4n^4 + 3n^4 + 10n^4 = 18n^4$.

## (b)

Define $q(n) = 2^{\log_2(n^3)} = n^3$. Let $r(n) = p(n) - q(n) = 4n^2 - 3n + 10n \log n$. Prove or disprove that $r(n)$ is $O(n)$.

**Disproof:**

1. Dominant term in $r(n)$: $4n^2$.
2. $4n^2$ grows faster than $O(n)$.
3. Thus, $r(n) \notin O(n)$.

## Problem 5. (10 + 2 points)

**(a)** What is the output for the following merge sort algorithm after each recursive split and merge step? Please provide a diagram similar to the one below:

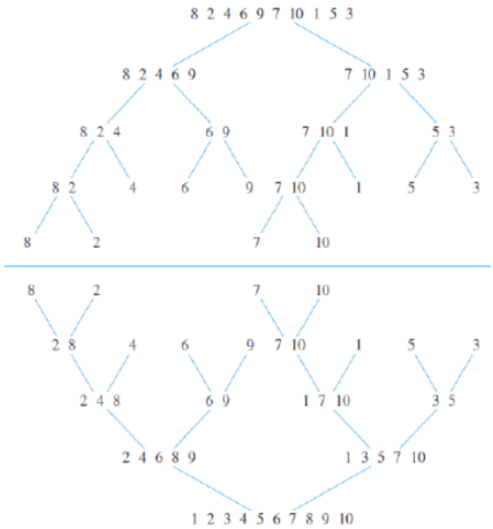


FIGURE 2 **The merge sort of 8, 2, 4, 6, 9, 7, 10, 1, 5, 3.**

**(b)** Briefly explain how merge sort uses the divide-and-conquer approach to sort the array.

**Solution (a) Merge Sort Steps:**

```
Original: [14, 10, 8, 15, 7, 11, 3]
Split into [14, 10, 8] and [15, 7, 11, 3]
Recursively split and merge:
Left half: [14, 10, 8] → [10, 14, 8] → [8, 10, 14]
Right half: [15, 7, 11, 3] → [7, 15, 3, 11] → [3, 7, 11, 15]
Merge: [8, 10, 14] + [3, 7, 11, 15] → [3, 7, 8, 10, 11, 14, 15]
```

**(b) Divide-and-Conquer:**

1. **Divide:** Split the array into halves.
2. **Conquer:** Recursively sort each half.
3. **Combine:** Merge the sorted halves.
4. Time complexity: $O(n \log n)$.

**Algorithm 1** MergeSort Algorithm

```
a = [14, 10, 8, 15, 7, 11, 3]
function MergeSort(a)
    if length(a) ≤ 1 then
        return a
    end if
    mid = length(a)/2
    left = MergeSort(a[0 : mid])
    right = MergeSort(a[mid :])
    return Merge(left, right)
end function
function Merge(left, right)
    result = []
    while left and right do
        if left[0] < right[0] then
            result.append(left.pop(0))
        else
            result.append(right.pop(0))
        end if
    end while
    return result + left + right
end function
```

# Problem 6 (12 + 8 points)

Imagine you are working in a sushi restaurant, and the sushi plates with unique prices are arranged on a rotating conveyor belt in sorted order by price. However, at the start of the day, the belt rotates multiple times, causing the order of sushi plates to change. For example, the sushi plates price array = [0,1,2,4,5,6,7] might become: [4,5,6,7,0,1,2] if it was rotated 4 times. Your task is to help the chef identify the cheapest sushi plate using an efficient algorithm.

(a) Write the pseudocode for your algorithm.

(b) Explain how your algorithm works and why it runs in $O(\log n)$ time.

**Solution (a) Pseudocode:**

```
function FindMin(arr)
    low = 0, high = len(arr) − 1
    while low < high do
        mid = ⌊(low + high)/2⌋
        if arr[mid] > arr[high] then
            low = mid + 1
        else
            high = mid
        end if
```

10: **end while**
11: **return** $arr$[low]
12: **end function**

**(b) Explanation:**

1. **Binary Search:** The algorithm uses binary search to find the pivot point.
2. **Comparison:** By comparing $arr$[mid] with $arr$[high], it determines which half contains the minimum.
3. **Time Complexity:** Each iteration halves the search space, resulting in $O(\log n)$ steps.