# Homework 4: Recursion and Big-O

Priya Natarajan
pnatarajan@cs.example.edu

---

**1.** Give the tightest big-O bound on the running time of the following function of $n$, the length of the input array, and justify your answer.

---

```
def mystery(arr):
    total = 0
    i = 0
    while i < len(arr):
        j = i
        while j < len(arr):
            total += arr[i] * arr[j]
            j += 1
        i += 1
    return total
```

The outer loop runs $n$ times, and for each outer iteration the inner loop runs at most $n$ times, so the total work is

$$\sum_{i=0}^{n-1} (n - i) = \frac{n(n+1)}{2} = O(n^2).$$

No early exit is possible, so this bound is tight: $\Theta(n^2)$.

---

**2.** Write a recurrence for the running time of the following recursive function and solve it using the Master Theorem.

---

```
def divide_and_conquer(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left  = divide_and_conquer(arr[:mid])
    right = divide_and_conquer(arr[mid:])
    return merge(left, right)   # merge is O(n)
```

The recurrence is $T(n) = 2T(n/2) + O(n)$, with $a = 2$, $b = 2$, and $f(n) = \Theta(n)$. Since $n^{\log_b a} = n^{\log_2 2} = n^1$ matches $f(n) = \Theta(n)$, this is Master Theorem Case 2, so

$$T(n) = \Theta(n \log n).$$

---

**3.** Prove, by induction on $n$, that the recursion tree for `divide_and_conquer` on an input of size $n$ has exactly $\lceil \log_2 n \rceil + 1$ levels.

**Base case.** For $n = 1$, the function returns immediately with no recursive call, so the tree has exactly one level, and $\lceil \log_2 1 \rceil + 1 = 0 + 1 = 1$.

**Inductive step.** Assume the claim holds for all inputs of size strictly less than $n$. On input of size $n > 1$, the function makes two recursive calls on inputs of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$, each with at most $\lceil \log_2 \lceil n/2 \rceil \rceil + 1$ levels by the inductive hypothesis. The tree for $n$ is one level deeper than the larger of its two children, giving $\lceil \log_2 \lceil n/2 \rceil \rceil + 2 = \lceil \log_2 n \rceil + 1$ levels, which completes the induction.

**4.** Short answer: why does the recursion in Question 2 terminate?

1. Every recursive call is made on a strictly smaller sub-array (size $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$, both $< n$ for $n > 1$).
2. There is a base case (`len(arr) <= 1`) that makes no further recursive call.

Together these guarantee the recursion depth is finite and bounded by $\lceil \log_2 n \rceil + 1$, from Question 3.