Solution to Homework 2

Part A: Master Method

We are solving the recurrence relation:

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

Step 1: Standard Form

The recurrence is in the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where:

- $a = 4$,
- $b = 2$,
- $f(n) = n^3$.

Step 2: Compute p

$$p = \log_b a = \log_2 4 = 2$$

Step 3: Compare $f(n)$ with n^p

- $f(n) = n^3$,
- $n^p = n^2$.

Since $f(n)$ grows faster than n^p (because $3 > 2$), we use the third case of the Master Theorem.

Step 4: Apply the Master Theorem

Because $f(n) = \Omega(n^{p+\epsilon})$ for some $\epsilon > 0$ (e.g., $\epsilon = 1$), and the regularity condition holds, the solution is dominated by $f(n)$:

$$T(n) = \Theta(f(n)) = \Theta(n^3)$$

Part B: Iterative Substitution Method

We are solving the recurrence relation:

$$T(n) = 8T\left(\frac{n}{2}\right) + c n^2, \quad T(1) = 1$$

Step 1: Expand the recurrence

Substituting iteratively:

1. First substitution: $T(n) = 8T\left(\frac{n}{2}\right) + c n^2$

2. Second substitution: $T(n) = 8 \left(8T\left(\frac{n}{4}\right) + c \left(\frac{n}{2}\right)^2 \right) + c n^2$ $T(n) = 8^2 T\left(\frac{n}{4}\right) + 8c \frac{n^2}{4} + c n^2$
3. General pattern: $T(n) = 8^k T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k-1} 8^i c \frac{n^2}{4^i}$

Step 2: Stopping condition

We stop when $\frac{n}{2^k} = 1$, i.e., $k = \log_2 n$:

$$T(n) = 8^{\log_2 n} T(1) + \sum_{i=0}^{\log_2 n - 1} 8^i c \frac{n^2}{4^i}$$

Step 3: Simplify terms

1. First term:

$$8^{\log_2 n} = (2^3)^{\log_2 n} = n^3$$

Thus, $8^{\log_2 n} T(1) = n^3$.

2. Second term:

$$\sum_{i=0}^{\log_2 n - 1} 8^i \frac{n^2}{4^i} = n^2 \sum_{i=0}^{\log_2 n - 1} (2^2)^i (2^{-2i}) = n^2 \sum_{i=0}^{\log_2 n - 1} 1 = n^2 \log_2 n$$

Step 4: Combine results

$$T(n) = n^3 + c n^2 \log_2 n$$

Thus, the asymptotic bound is:

$$T(n) = \Theta(n^3)$$

Part C: Algorithm Analysis

Given Algorithm

```
Function ABC(A, n):
    for i = 1 to n do:
        for j = n downto i + 1 do:
            if A[j] < A[j - 1] then:
                SWAP(A[j], A[j - 1])
```

Step 1: Outer Loop Analysis

The outer loop runs from $i = 1$ to n, so it executes n iterations.

Step 2: Inner Loop Analysis

For each iteration of i, the inner loop runs from $j = n$ to $i+1$. The number of iterations for the inner loop is:

$$n - (i + 1) + 1 = n - i$$

Step 3: Total Iterations

The total number of iterations is the sum over all values of i:

$$\text{Total steps} = \sum_{i=1}^n (n - i) = n + (n - 1) + (n - 2) + \dots + 1$$

This is the sum of the first $n-1$ integers:

$$\text{Total steps} = \frac{n(n-1)}{2}$$

Step 4: Swap Operations

Each iteration of the inner loop involves at most one comparison and one swap. Therefore, the number of swaps is also:

$$\frac{n(n-1)}{2}$$

Thus, the algorithm has a time complexity of:

$$\Theta(n^2)$$
