Problem 1. (15 points [analysis of iterative pseudocode])

Give a tight bound (using the big-theta notation) on the number of \( H_i \)'s produced by the following method as a function of \( n \). For simplicity, you can assume \( n \) to be a power of two.

\[
\text{Algorithm LOOP1 (} n : \text{integer)} \\
i \leftarrow n \\
\text{while } i \geq 1 \text{ do} \\
\quad \text{for } j \leftarrow i \text{ to } 2i - 1 \text{ do} \\
\quad \quad \text{print "Hi"} \\
\quad i \leftarrow i/2 \\
\]

Answer: When \( i = n \), the print statement is executed \( n \) times because \( j \) ranges from \( n \) to \( 2n - 1 \). When \( i = n/2 \) the print statement is executed \( n/2 \) times because \( j \) ranges from \( n/2 \) to \( n - 1 \). \ldots When \( i = 2 \) the print statement is executed twice because \( j \) ranges from 2 to 3. When \( i = 1 \) the print statement is executed once because \( j \) ranges from 1 to 1. The total is \( n + n/2 + \ldots + 2 + 1 \in \Theta(n) \).

Problem 2. (15 points [recurrence relations])

Solve exactly (that is, without using any asymptotic notation) the following recurrence relation by iterative substitutions

\[
T(n) = \begin{cases} 
1 & n = 1 \\
3T\left(\frac{n}{2}\right) + 2 & n > 1 
\end{cases}
\]

Answer: We have

\[
T(n) = 3T(n/2) + 2 \\
= 3(3T(n/4) + 2) + 2 \\
= 3^2T(n/4) + 3 \times 2 + 2 \\
= 3^3T(n/8) + 3^2 \times 2 + 3 \times 2 + 2 \\
\ldots \\
= 3^iT(n/2^i) + (3^{i-1} + 3^{i-2} + \ldots + 1) \times 2 \\
= 3^iT(n/2^i) + 3^i - 1 \\
\]

now we set \( n/2^i = 1 \) which is \( i = \log_2 n \) and we get

\[
T(n) = 3^{\log_2 n}T(1) + 3^{\log_2 n} - 1 \\
= 2n^{\log_2 3} - 1 \\
\]

Problem 3. (15 points [recurrence relations])
Using the Master method, give an asymptotic tight bound for $T(n)$ in the following recurrence relation

$$T(n) = \begin{cases} 
1 & n = 1 \\
T\left(\frac{n}{2}\right) + n \log_2 n & n > 1 
\end{cases}$$

**Answer:** Case 3 of Master theorem applies. First note that $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$. The first condition for case 3 is $n \log_2 n \in \Omega(n^\epsilon)$ which is satisfied for $\epsilon = 1$. The second condition is $af(n/b) \leq \delta f(n)$, which translates to

$$\frac{n}{2} \log_2 \frac{n}{2} = \frac{n}{2} \log_2 n - \frac{n}{2} \log_2 2 = \frac{n}{2} \log_2 n - \frac{n}{2} \leq \delta n \log_2 n$$

The last inequality is satisfied by $\delta = 1/2 < 1$.

The conclusion is $T(n) = \Theta(n \log n)$.

**Problem 4.** (15 points [writing/solving recurrence relations])
Consider the following variant for the closest-pair algorithm we discussed in class.

**Algorithm** Closest-Pair ($A$ : array of 2D points)

1. **if** $|A| \leq 2$ **then return** the pair in $O(1)$ time
2. **sort** set $A$ by $x$ coordinate
3. $dL \leftarrow$ Closest-Pair($A[1 \ldots n/2]$)
4. $dR \leftarrow$ Closest-Pair($A[n/2 + 1 \ldots n]$)
5. $d \leftarrow \min(dL, dR)$
6. $S \leftarrow \{p \in A : |p.x - A[n/2].x| \leq d\}$ be the points in the vertical strip around the median
7. $dS \leftarrow$ the closest-pair distance of points in $S$, computed in $O(n \log n)$ time
8. **return** $\min(d, dS)$

Assume each basic operation on points (computing distance, comparing coordinates, . . .) takes $O(1)$ time. Define $T(n)$ to be the worst-case running time of Closest-Pair($A$) on any array $A$ of $n$ points. Write a recurrence relation for $T(n)$ and give a tight bound on $T(n)$.

**Answer:** The recurrence relation for $T(n)$ is

$$T(n) = 2T(n/2) + n \log n \quad \text{for } n \geq 3$$

and

$$T(n) = 1 \quad \text{for } n \leq 2$$

By Master theorem Case 2 ($k = 1$) we have $T(n) \in \Theta(n \log^2 n)$.

**Problem 5.** (20 points [divide & conquer])
Another version of Strassen’s algorithm uses the following identities. To compute

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} E & F \\ G & H \end{bmatrix} \begin{bmatrix} I & J \\ K & L \end{bmatrix}$$

first compute the following values:
\[ s_1 = G + H \quad m_1 = s_2 s_6 \]
\[ s_2 = s_1 - E \quad m_2 = EI \]
\[ s_3 = E - G \quad m_3 = FK \]
\[ s_4 = F - s_2 \quad m_4 = s_3 s_7 \]
\[ s_5 = J - I \quad m_5 = s_1 s_5 \]
\[ s_6 = L - s_5 \quad m_6 = s_4 L \]
\[ s_7 = L - J \quad m_7 = H s_8 \]
\[ s_8 = s_6 - K \]

then

\[ A = m_2 + m_3 \]
\[ B = m_1 + m_2 + m_5 + m_6 \]
\[ C = m_1 + m_2 + m_4 - m_7 \]
\[ D = m_1 + m_2 + m_4 + m_5 \]

1. Analyze the time complexity \( T(n) \) of this new algorithm assuming that the input are two \( n \times n \) matrices

2. Compare \( T(n) \) against that of the original Strassen. Is it faster, slower, or the same?

**Answer:**

1. Since there are 7 multiplications and 18 additions, the recurrence relation is

\[
T(n) = \begin{cases} 
  c & n = 2 \\
  7 T \left( \frac{n}{2} \right) + 18n^2 & n > 2 
\end{cases}
\]

2. The solution of the recurrence relation is \( T(n) \in O(n^\log_2 7) \), which is the same time complexity as Strassen’s.

**Problem 6.** (20 points [divide & conquer])

Suppose you are given an array \( A = \{a_1, a_2, \ldots, a_n\} \) of \( n \) distinct integers. You are told that the sequence of values \( a_1, a_2, \ldots, a_n \) is unimodal, that is for some index \( p \in [1, n] \), the values in the array increase up to position \( p \), and then decrease the remainder of the way until position \( n \). Give an algorithm to find the position \( p \) in \( O(\log n) \) time. You can assume \( n \) to be a power of 2.

**Answer:** Compare the elements \( A[n/2], A[n/2 - 1] \) and \( A[n/2 + 1] \) to decide whether to search on the left, on the right, or whether we are done. More specifically


The algorithm has the same structure of binary search, its recurrence relation is \( T(n) = T(n/2) + O(1) \), which has solution \( O(\log n) \).