Be aware that your homework should be your own work. It is a matter of intellectual honesty to write your homework strictly by yourself. Using solutions from any other source is not allowed.

1. Use mathematical induction to show that when \( n \) is a power of 2, \( T(n) = n \log n \) is the solution of the recurrence relation

\[
T(n) = \begin{cases} 
2 & \text{if } n = 2 \\
2T\left(\frac{n}{2}\right) + n & \text{if } n = 2^k \text{ for } k > 1.
\end{cases}
\]

2. Suppose we are comparing implementations of INSERT-SORT and MERGE-SORT on the same machine. For input of size \( n = 2^k \) for \( k \geq 1 \), INSERT-SORT runs in \( 8n^2 \) steps, while MERGE-SORT runs in \( 64n \log n \) steps. For which value of \( n \) does INSERT-SORT beat MERGE-SORT?

3. We can express INSERT-SORT as a recursive procedure as follows. In order to sort \( A[1...n] \), we recursively sort \( A[1...n-1] \) and then insert \( A[n] \) into sorted array \( A[1...n-1] \).

   (a) Write the pseudocode for this recursive version of INSERT-SORT, name it INSERT-SORT-RECUR.

   (b) Write a recurrence for the running time of INSERT-SORT-RECUR.

   (c) Find the solution of this recurrence relation.

   (d) Is INSERT-SORT-RECUR more expensive than INSERT-SORT?

4. In this exercise, we consider a SELECTION-SORT algorithm. To sort \( n \) numbers stored in array \( a \), we first find the smallest element of \( A \) and exchanging it with the element in \( a[1] \). Then find the second smallest element of \( a \), and exchange it with \( a[2] \). Continue in this manner for the first \( n-1 \) element of \( a \).

   (a) Write a pseudocode for the SELECTION-SORT algorithm.

   (b) Analyze the running times.

5. Given an array \( s = \langle s[1], s[2], \ldots, s[n] \rangle \), and \( n = 2^d \) for some \( d \geq 1 \). We want to find the minimum and maximum values in \( s \). We do this by comparing elements of \( s \).

   (a) The “obvious” algorithm makes \( 2n - 2 \) comparisons. Explain.

   (b) Can we do it better? Carefully specify a more efficient divide-and-conquer algorithm.

   (c) Let \( T(n) \) = the number of comparisons your algorithm makes. Write a recurrence relation for \( T(n) \).

   (d) Show that your recurrence relation has as its solution \( T(n) = 3n/2 - 2 \).

6. Let \( S \) be an array of \( n \) integers such that \( S[1] < S[2] < \cdots < S[n] \). (1) Specify an \( O(\log n) \) algorithm which either finds an \( i \in \{1, 2, \ldots, n\} \) such that \( S[i] = i \) or else determine that there is no such \( i \). (2) Justify the correctness and running time of your algorithm.