Tutorial 2 DAA

1. Using Figure 2.4 as a model, illustrate the operation of merge sort on the array $A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$.

2. Rewrite the MERGE procedure so that it does not use sentinels, instead stopping once either array L or R has had all its elements copied back to A and then copying the remainder of the other array back into A.

3. Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2 & \text{if } n = 2 , \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

is $T(n) = n \lg n$.

4. We can express insertion 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 the sorted array $A[1 \,..\, n-1]$. Write a recurrence for the worst-case running time of this recursive version of insertion sort.

5. Referring back to the searching problem (see Exercise 2.1-3), observe that if the sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. The ***binary search*** algorithm repeats this procedure, halving the size of the remaining portion of the sequence each time. Write pseudocode, either iterative or recursive, for binary search. Argue that the worst-case running time of binary search is $\Theta(\lg n)$.

6. Observe that the **while** loop of lines 5–7 of the INSERTION-SORT procedure in Section 2.1 uses a linear search to scan (backward) through the sorted subarray $A[1 \,..\, j-1]$. Can we use a binary search (see Exercise 2.3-5) instead to improve the overall worst-case running time of insertion sort to $\Theta(n \lg n)$?

7. Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.