Exercise Session 03

Exercise 1.
Consider the array $A = [3, 41, 52, 26, 38, 57, 49, 9]$. Give the state of the array after five sub-calls of the algorithm MERGE are performed during the execution of the call MERGE-SORT$(A, 1, 8)$.

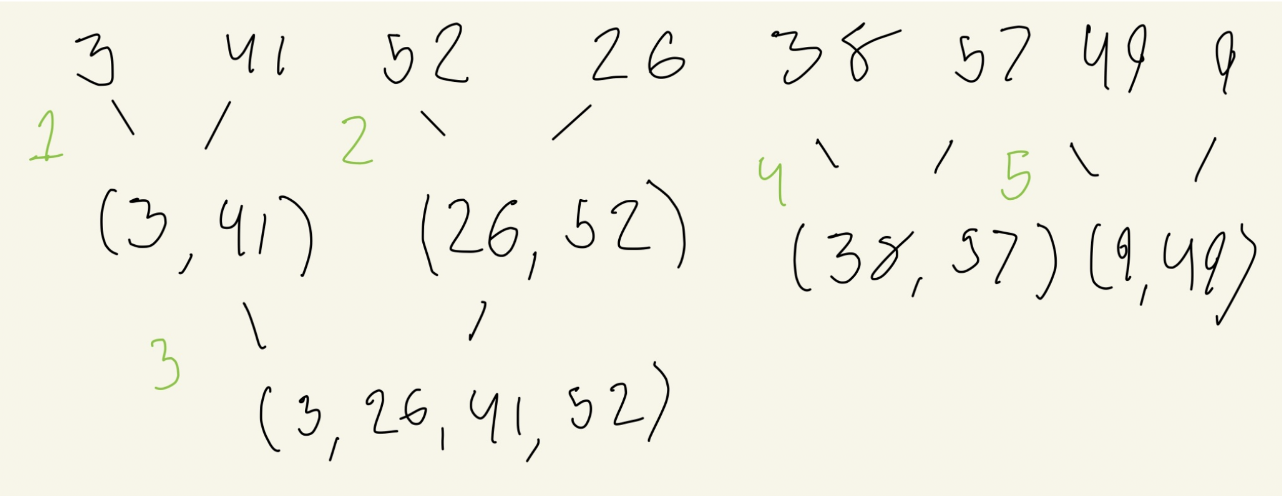


Figure 1: MergeSort after 5 iterations of Merge

Exercise 2.
CLRS-3 2.3–3. Use mathematical induction to show that when n is an exact power of 2 (that is, $n = 2^k$ for some $k \in \mathbb{N} \setminus \{0\}$), the solution of the following recurrence is $T(n) = n \lg n$

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

Base case $n = 2$

$$\begin{aligned} T(n) &= nlog(n) \\ T(2) &= 2(2) \\ T(2) &= 2 \end{aligned}$$

Induction hypothesis: Assume that $l = 2^k$ then $T(2^k) = 2^k log(2^k)$
Induction step: show that $T(l) = l \cdot log(l) = 2^{k+1} log(2^{k+1})$

$$
\begin{aligned}
l &= 2^{k+1} \\
T(l) &= 2T(l/2) + l \\
T(2^{k+1}) &= 2T(\frac{2^{k+1}}{2}) + 2^{k+1} \\
&= 2T(2^k) + 2^{k+1} \\
&= 2(2^k \cdot log(2^k)) + 2^{k+1} \\
&= 2^{k+1} \cdot log(2^k) + 2^{k+1} \\
&= 2^{k+1}(log(2^k) + 1) \\
&= 2^{k+1}(k + 1) \\
&= 2^{k+1}(log(2^{k+1}) \\
&= 2^{k+1} \cdot log(2^{k+1}) \\
&= l \cdot log(l)
\end{aligned}
$$

```
INSERTIONSORT(A, p)
1  if p > 1
2      INSERTIONSORT(A, p − 1)
3      // Insert A[p] into the sorted sequence A[1 . . p − 1]
4      key = A[p]
5      i = p − 1
6      while i > 0 and A[i] > key
7          A[i + 1] = A[i]
8          i = i − 1
9      A[i + 1] = key
```

CLRS-3 2.3–4. 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.

$$
T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n-1) + \Theta(n) & \text{if } n > 1 \end{cases}
$$

In the base case that there is only a single element in the list the time complexity is $\Theta(1)$ as it is trivially sorted. Otherwise InsertionSort recursively sorts the current element. Moreover InsertionSort is called recursively $\Theta(n-1)$ until we hit the base case $\Theta(n)$

CLRS-3 2.3–6. Observe that the while loop of lines 5–7 of the INSERTION-SORT procedure uses a linear search to scan (backward) through the sorted subarray $A[1\,..\,j-1]$. Can we use a binary search instead to improve the overall worst-case running time of insertion sort to $\Theta(nlgn)$?
Answer:Insertion sort makes use of shifting elements no matter what and searching with a $\Theta(nlgn)$ algorithm will still lead to a overall time complexity of $\Theta(n^2)$

Exercise 3.

Consider the problem of finding the smallest element in a nonempty array of numbers $A[1\,..\,n]$.

(a) Write an *incremental* algorithm that solves the above problem and determine its asymptotic worst-case running time.

```
function findSmallest(A) {
    smallest = A[1]
    for (index = 2 to A.length) {
        if(A[index] < smallest)
        smallest = A[index]
    }
    return smallest
}
```

Figure 2: Caption

$$T(n) = c_1 + c_2 n + (c_3 + c_4)(n - 1) + c_5 = \Theta(n)$$

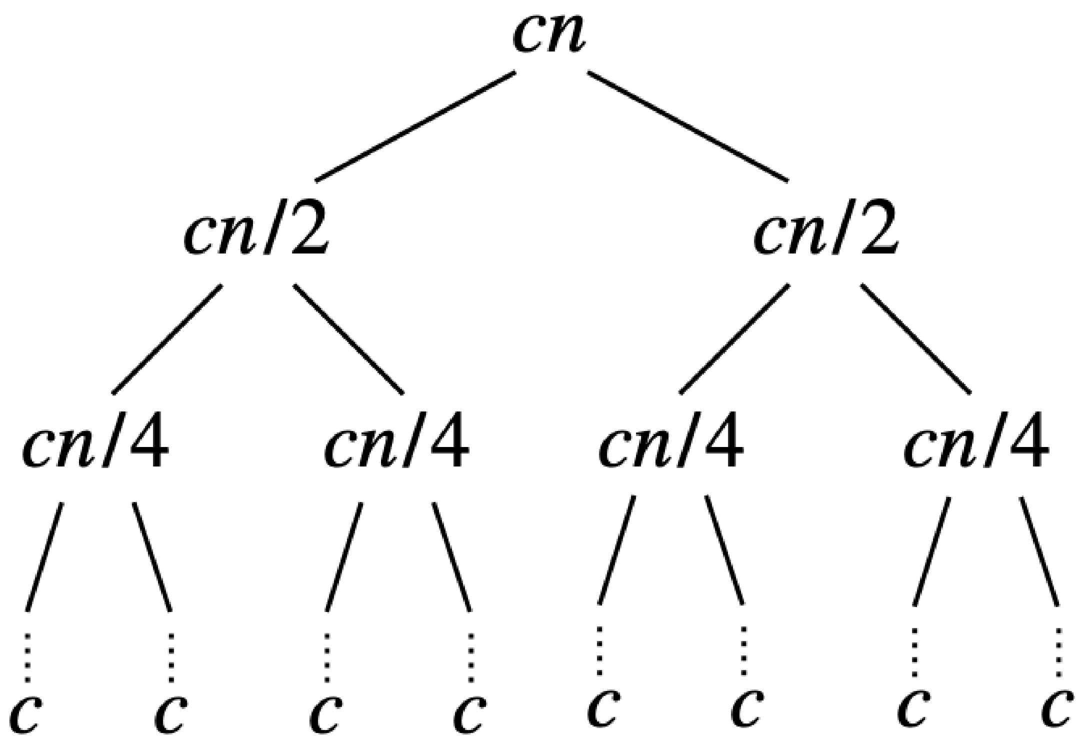
(b) Write a *divide-and-conquer* algorithm that solves the above problem and determine its asymptotic worst-case running time.

```
function findSmallest(A, l, r)
  if (l >= r) return A[l];
  else
    let m = Math.floor((l + r) / 2);
    let min1 = findSmallest(A, l, m);
    let min2 = findSmallest(A, m + 1, r);
    return Math.min(min1, min2);
```

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n/2) + \Theta(1) & \text{if } n > 1 \end{cases}$$

(c) Assume that the length of A is a power of 2. Write a recurrence describing how many comparison operations (among elements of A) your divide-and-conquer algorithm performs, and solve the recurrence using the recursion-tree method.

$$C(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2C(n/2) + 1 & \text{if } n = 2^k \text{ and } k \geq 1 \end{cases}$$



> **Remark**: count ONLY the comparisons performed among elements in A. E.g., a comparison like $i \leq A.length$ shall not be counted, whereas $A[i] \leq k$ where k is a variable storing some element of A shall be counted. Moreover, if you use expressions like $\min(A[i], A[j])$ for some indices i, j, that also counts as 1 comparison.

Hint: A full binary tree with n leaves has $n-1$ internal nodes (see CLRS-3 B.5-3 pp.1177–1179, or CLRS-4 B.5.3 pp.1173–1175).

★ Exercise 4.

[CLRS-3 2.3–7] 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.