RA 15

8.12 Analyzing time complexity of recursive algorithms

- The time complexity of an algorithm $T(n)$ indicating the number of atomic operations performed by the algorithm on an input of size n.

Example

```
Factorial (n)
If (n=0), Return(1)
r := Factorial(n-1)
Return (r*n)
```

if $n=0$, two atomic operations are performed (conditional and return)

Recursive call to Factorial on input $n-1$ is $T(n-1)$.

Initial Value: $T(0) = 2$

Recurrence relation: for $n \geq 1$, $T(n) = T(n-1) + 5$

Examples of simplifying Recurrence relations

- $T(n) = 3 \cdot T(\frac{n}{2}) + 7n \rightarrow T(\frac{n}{2}) + \Theta(n)$
  - ↳ INCORRECT, CANNOT Remove scaler on $T(\frac{n}{2})$ term

- $T(n) = T(\lfloor \frac{n}{2} \rfloor) + 5n^2 + 10 \rightarrow T(n) = T(\frac{n}{2}) + \Theta(n^2)$
  - ↳ Correct

- $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lfloor \frac{n}{2} \rfloor) + \Theta(n) \longrightarrow T(n) = 2 \cdot T(\frac{n}{2}) + \Theta(n)$
  - ↳ Correct

8.13 Divide and Conquer Algorithms: Introduction and mergesort

- A divide and conquer algorithm solves a problem recursively by breaking the original input into smaller subproblems of roughly equal size. They follow 3 basic steps:
  1. Break the input into smaller subproblems of the same type on smaller inputs.
  2. Solve each subproblem recursively.
  3. Combine the solutions of the subproblems to obtain a solution to the original problem.

Merge Sort

- Merge Sort sorts the list $(17, 3, 9, 11, 21, 4, 7, 1)$ in 4 basic steps:
  1. Break list into two sub-lists: $(17, 3, 9, 11)$ and $(21, 4, 7, 1)$
  2. Make a recursive call to MergeSort with input $(17, 3, 9, 11)$, which returns $(3, 9, 11, 17)$
  3. Make a recursive call to MergeSort with input $(21, 4, 7, 1)$, which returns $(1, 4, 7, 21)$
  4. Merge the sorted sub-lists to obtain $(1, 3, 4, 7, 9, 11, 17, 21)$.

- Merge Sort is $\Theta(n \log n)$
- Merge Sort uses quene data structure:
  - New items can be added to one end called the back.
  - Items can be removed from the other end of the quene called the front.

Example: Quene B: $(5, 8, 7)$

↳ Remove (B): $(8, 7)$

↳ Add (9): $(8, 7, 9)$

8.14 Divide and Conquer algorithms: Binary Search

- Pseudocode for recursive binary search

```
RecBinarySearch(low, high, A, x)

if (low = high AND a_low = x), Return(low)
if (low = high AND a_low ≠ x), Return(-1)

mid := ⌊(low + high)/2⌋

if (x ≤ a_mid), then high := mid
if (x > a_mid), then low := mid + 1

Return[RecBinarySearch(low, high, A, x)]
```

- If size of list is even, then the two subranges are both $\frac{n}{2}$: $(mid - low + 1) = (high - mid) = \frac{n}{2}$
- If size of list is odd, then the size of first subrange is $(mid - low + 1) = \lceil \frac{n}{2} \rceil$, and the size of the second subrange is $(high - mid) = \lfloor \frac{n}{2} \rfloor$.