Introduction to Algorithms

Date: 3/10 (Tuesday)

Instructor: 유준수

Assignment

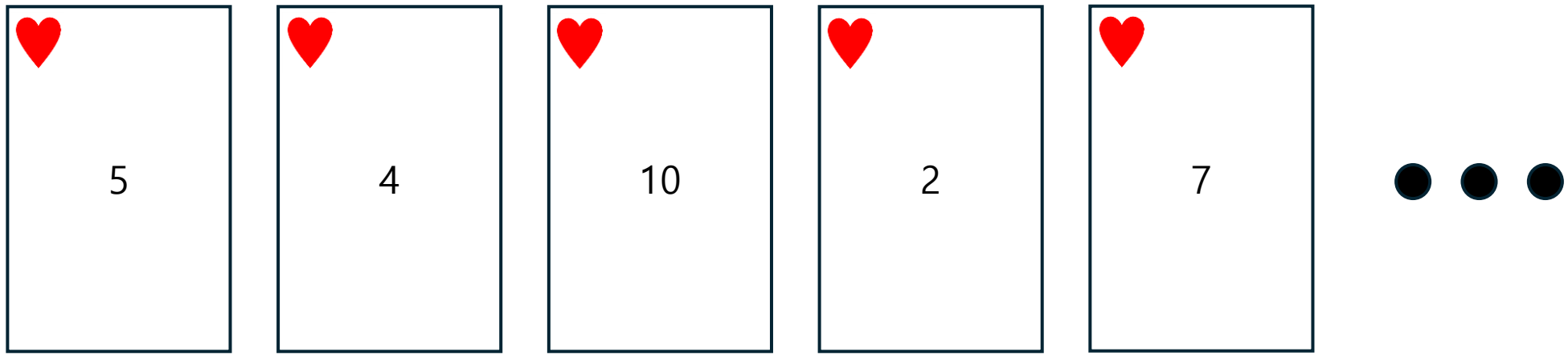
- Read 2.2, 2.3.1
- Problems:
  - 2.1절 –

Review

- 1.1 Algorithms
- 1.2 Algorithms as a technology

Summary

Insertion Sort



Pseudocode

Correctness

- Loop Invariant
- Init/Main/Ter

```
INSERTION-SORT(A, n)
1  for i = 2 to n
2      key = A[i]
3      // Insert A[i] into the sorted subarray A[1 : i − 1].
4      j = i − 1
5      while j > 0 and A[j] > key
6          A[j + 1] = A[j]
7          j = j − 1
8      A[j + 1] = key
```

2.1-4

Consider the *searching problem*:

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$ stored in array $A[1:n]$ and a value x.

Output: An index i such that x equals $A[i]$ or the special value NIL if x does not appear in A.

Write pseudocode for *linear search*, which scans through the array from beginning to end, looking for x. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

Think about 2 min…

Pseudocode

Proof of Correctness

Chapter 2. The Role of Algorithms in Computing

- 2.1 Insertion sort
- 2.2 Analyzing algorithms
- 2.3 Designing algorithms

Model Assumption: RAM

Calculate Time Complexity (hard way): Insertion Sort

```
INSERTION-SORT(A, n)
1  for i = 2 to n
2      key = A[i]
3      // Insert A[i] into the sorted subarray A[1 : i − 1].
4      j = i − 1
5      while j > 0 and A[j] > key
6          A[j + 1] = A[j]
7          j = j − 1
8      A[j + 1] = key
```

Best case analysis

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=2}^{n} t_i + c_6 \sum_{i=2}^{n} (t_i - 1) + c_7 \sum_{i=2}^{n} (t_i - 1) + c_8(n-1) .$$

Worst-case analysis

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=2}^{n} t_i + c_6 \sum_{i=2}^{n} (t_i - 1) + c_7 \sum_{i=2}^{n} (t_i - 1) + c_8(n-1) \; .$$

Average-case analysis

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=2}^{n} t_i + c_6 \sum_{i=2}^{n} (t_i - 1) + c_7 \sum_{i=2}^{n} (t_i - 1) + c_8(n-1) \; .$$

Summary of time complexity table (insertion sort)

Case	t_i	Time Complexity
Best-case		
Worst-case		
Avg-case		

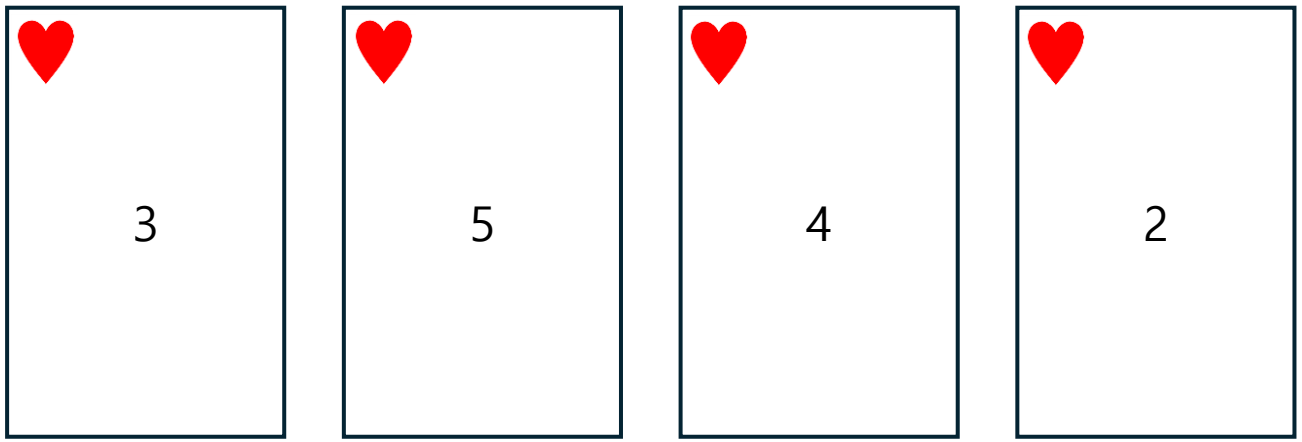
Order of Growth (Rate of growth)

Why do we ignore the constant term?

Chapter 2. The Role of Algorithms in Computing

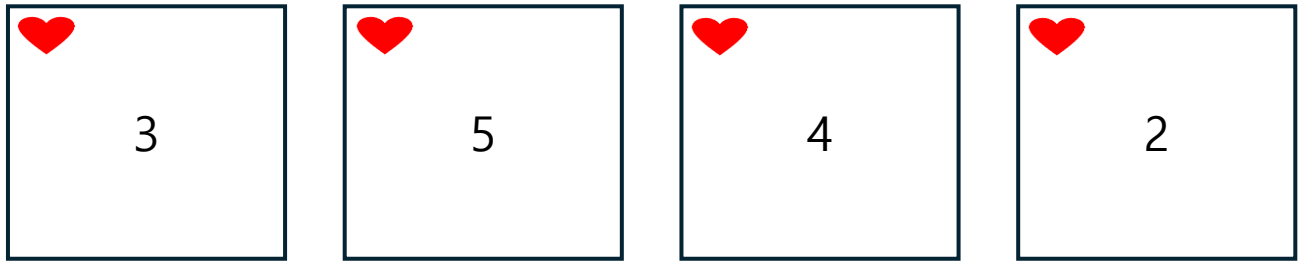
- 2.1 Insertion sort
- 2.2 Analyzing algorithms
- 2.3 Designing algorithms

2.3.1 Divide & Conquer Method



Incremental Approach:
Insertion Sort

2.3.1 Divide & Conquer Method: Example of Merge Sort



Number of Comparisons for $n = 4$

Algorithm	$number\ of\ comparsions$
Insertion Sort	
Merge Sort	

General problem solving strategy of Divide and Conquer Method

Divide the problem into one or more subproblems that are smaller instances of the same problem.

Conquer the subproblems by solving them recursively.

Combine the subproblem solutions to form a solution to the original problem.

Question?