# Data structure [A02]

김종규, PhD

2017-03-13

# Sorting concept

- 다음에 주어진 10 개의 수 중 가장 작은 수는?

```
x = [40694 ,
73593 ,
13612 ,
65541 ,
19386 ,
2347 ,
26723 ,
42533 ,
27999 ,
96272 ]
```

- 네 번째로 작은 수는?

# Sorting concept

- 다음에 주어진 10 개의 수 중 가장 작은 수는?

```
x = [ 2347 ,
13612 ,
19386 ,
26723 ,
27999 ,
40694 ,
42533 ,
65541 ,
73593 ,
96272 ]
```

- 네 번째로 작은 수는?

⟶ 어떻게 쉽게 알 수 있을까? ⟶ 쭉 읽어보니까 점점

# Sorting idea

- Sorting 되어 있다면?

```
x = [1, 2, 3, 4]
```

⟶ 쉽게 확인 가능

# Sort testing algorithm

```
def is_sorted(x)
  n = len(x)
  sorted = True
  for i in range(n-1):
    if x[i] > x[i+1]:
      sorted = False
  return sorted
```

# Sorting idea

- Sorting 되어 있지 않은 것을 발견할 때 순서를 바꾸면?

```
def test_sort(x)
  n = len(x)
  for i in range(n-1):
    if x[i] > x[i+1]:
      (x[i+1],x[i])=(x[i],x[i+1])
```

- 예를 들어 다음의 경우

```
2 1 3 4
```

```
1 2 3 4
```

$\longrightarrow$ 이걸로 충분할까?

# Sorting idea

- 다음에 적용하면?

```
4  3  2  1

4     3  2  1
3  4     2  1
3  2  4     1
3  2  1  4
```

⟶ 여러번 반복해야 한다.

⟶ 가장 작은 값은? 반드시 한 칸 자리를 옮긴다

⟶ 가장 큰 값은? 반드시 가장 마지막으로 간다

# Sorting algorithm

- 이 과정을 한 번 더 수행하면? $\longrightarrow$ 가장 큰 값은 맨 뒤에, 그 다음으로 큰 값은 바로 앞에 오게 된다.
- 이 과정을 n 번 반복한다면? $\longrightarrow$ Sorting 완료

```
    4 3 2 1
1: 3 2 1 4
2: 2 1 3 4
3: 1 2 3 4
4: 1 2 3 4
```

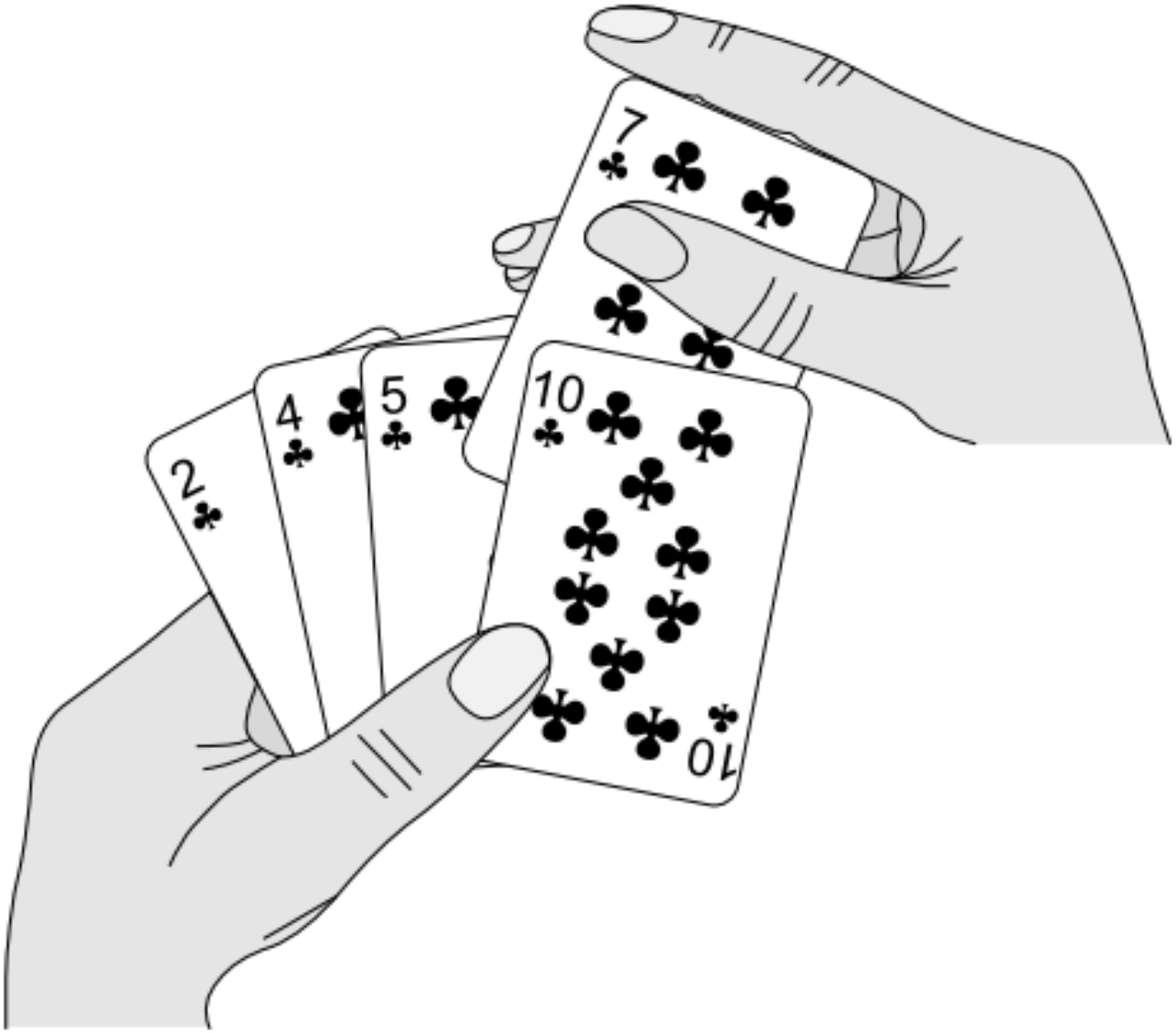
# Sorting algorithm

```
  def test_sort(x)
1   n = len(x)
2   for j in range(n):
3     for i in range(n-1):
4       if x[i] > x[i+1]:
5         (x[i+1],x[i])=(x[i],x[i+1])
```

$\longrightarrow$ Bubble sort

- 비교는 몇 번? $\longrightarrow n^2$

# Insertion sort: concept



# Insertion sort: example

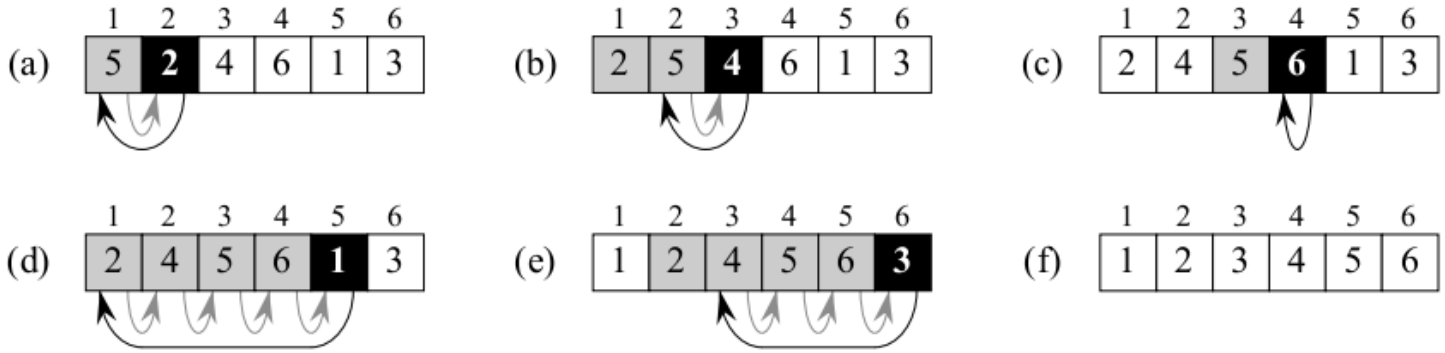



그림: Insertion sort

# Insertion sort: pseudocode



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

그림: Insertion sort

# C 언어

```
void insertion_sort(int A[], int nelem)
{
  for (int j = 1; j < nelem; j++) {
    int key = A[j];
    int i = j - 1;
    while (i >= 0 && A[i] > key) {
      A[i + 1] = A[i];
      i = i - 1;
    }
    A[i + 1] = key;
  }
}
```

# Bubble sort: pseudocode



**ALGORITHM 4 The Bubble Sort.**

**procedure** *bubblesort*($a_1, \ldots, a_n$ : real numbers with $n \geq 2$)
**for** $i := 1$ **to** $n - 1$
    **for** $j := 1$ **to** $n - i$
        **if** $a_j > a_{j+1}$ **then** interchange $a_j$ and $a_{j+1}$
{$a_1, \ldots, a_n$ is in increasing order}

그림: Bubble sort

# Comparing two algorithms

- Bubble sort 와 Insertion sort 중 어떤 것이 더 좋을까?
- Best case: Insertion sort
- Worst case: ??

# Sorting algorithm

```
  def test_sort(x)
1   n = len(x)
2   for j in range(n):
3     for i in range(n-1):
4       if x[i] > x[i+1]:
5         (x[i+1],x[i])=(x[i],x[i+1])
```

$\longrightarrow$ Bubble sort

- 비교는 몇 번? $\longrightarrow n^2$

# Cost model

- $c_1$: `n = len(x)`
- $c_2$: `for j in range(n):`
- $c_3$: `  for i in range(n-1):`
- $c_4$: `    if x[i] > x[i+1]:`
- $c_5$: `      (x[i+1],x[i])=(x[i],x[i+1])`
- Best: $c = c_1 + (n+1)c_2 + n^2c_3 + n(n-1)c4 + n0c_5$
- Worst:

  $c = c_1 + (n+1)c_2 + n^2c_3 + n(n-1)c4 + n(n-1)c_5$

# Analysis of insertion sort: primitive cost model

| INSERTION-SORT$(A)$ | *cost* | *times* |
|---|---|---|
| 1 **for** $j = 2$ **to** $A.length$ | $c_1$ | $n$ |
| 2 $key = A[j]$ | $c_2$ | $n-1$ |
| 3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$. | 0 | $n-1$ |
| 4 $i = j-1$ | $c_4$ | $n-1$ |
| 5 **while** $i > 0$ and $A[i] > key$ | $c_5$ | $\sum_{j=2}^{n} t_j$ |
| 6 $A[i+1] = A[i]$ | $c_6$ | $\sum_{j=2}^{n} (t_j - 1)$ |
| 7 $i = i-1$ | $c_7$ | $\sum_{j=2}^{n} (t_j - 1)$ |
| 8 $A[i+1] = key$ | $c_8$ | $n-1$ |

그림: Insertion sort

# Analysis of insertion sort: mathematical model

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

그림: Insertion sort; cost

# Analysis of insertion sort: example (best case)

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\
&= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \; .
\end{aligned}
$$

그림: Insertion sort; best cost

# Insertion sort: example (worst case)

- `6 5 4 3 2 1`
  - j = 2 $\longrightarrow$ while 문은 1 회
  - j = 3 $\longrightarrow$ whlie 문은 2 회
  - j = 4 $\longrightarrow$ whlie 문은 3 회
  - j = 5 $\longrightarrow$ whlie 문은 4 회
  - j = 6 $\longrightarrow$ whlie 문은 5 회

$\longrightarrow$ $t_j = j - 1$

# Insertion sort – Worst case

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
&\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n \\
&\quad - (c_2 + c_4 + c_5 + c_8) \; .
\end{aligned}
$$

그림: Insertion sort – Worst case

# Wrap-up

- We introduced the concept of sorting
- We introduced two different sorting algorithms
- We analyzed the cost of sorting algorithms
- But it's hard to choose when the worst case is considered
- We will learn more about analysis