COMP319 Algorithms
Lecture 3
Simple Sorting Methods

Instructor: Gil-Jin Jang

Simple sorting methods

Insertion/selection/bubble sort

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

정렬의 정의

SORTING, DEFINITION

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

The Sorting Problem

- **Input:**
  - A sequence of n numbers $a_1, a_2, \ldots, a_n$
- **Output:**
  - (Ascending, 오름차순) A permutation (reordering) $a_1', a_2', \ldots, a_n'$ of the input such that $a_1' \leq a_2' \leq \cdots \leq a_n'$
  - (Descending, 내림차순) $a_1' \geq a_2' \geq \cdots \geq a_n'$

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

정렬 알고리즘의 개요

- 많은 정렬 알고리즘 존재
  - 단순하지만 비효율적인 방법 -- 삽입정렬, 선택정렬, 버블정렬 등
  - 복잡하지만 효율적인 방법 -- 퀵정렬, 히프정렬, 합병정렬, 기수정렬 등
- 정렬 알고리즘의 평가
  - 효율성: 비교회수 및 이동회수
  - 비교와 이동 회수는 비례하지 않음
- 모든 경우에 최적인 알고리즘은 없음
  - 응용에 맞추어 선택
  - 숫자와 문자열 비교, 숫자와 구조체 이동 고려

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Sorting Algorithm Comparison

- Insertion/Selection/Bubble sort
  - Advantages: using less extra memory
  - Disadvantages: $T(n) = T(n\text{-}1)+cn$ ➔ O(n^2)

- *Merge sort*
- *Quicksort*
- *Heapsort*

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

삽입 정렬

INSERTION SORT

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

삽입정렬 Insertion Sort

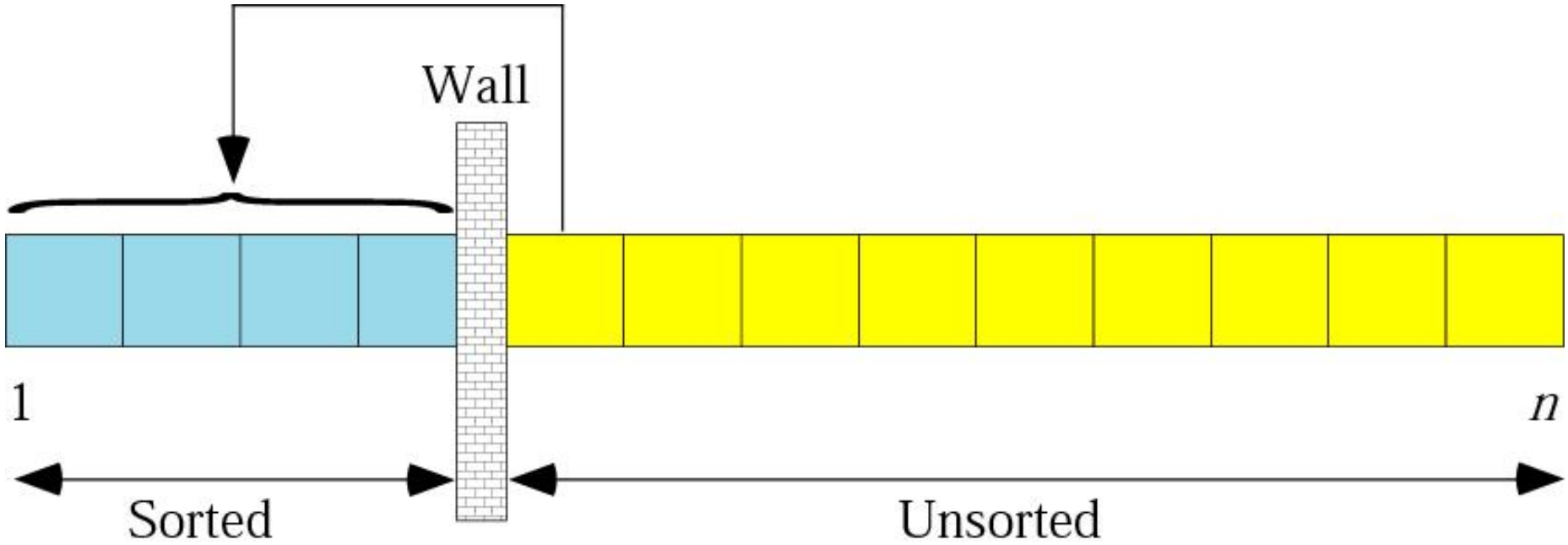
- 삽입정렬은 정렬되어 있는 부분에 새로운 레코드를 적절한 위치에 삽입하는 과정을 반복

삽입정렬의 과정

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Insertion sort

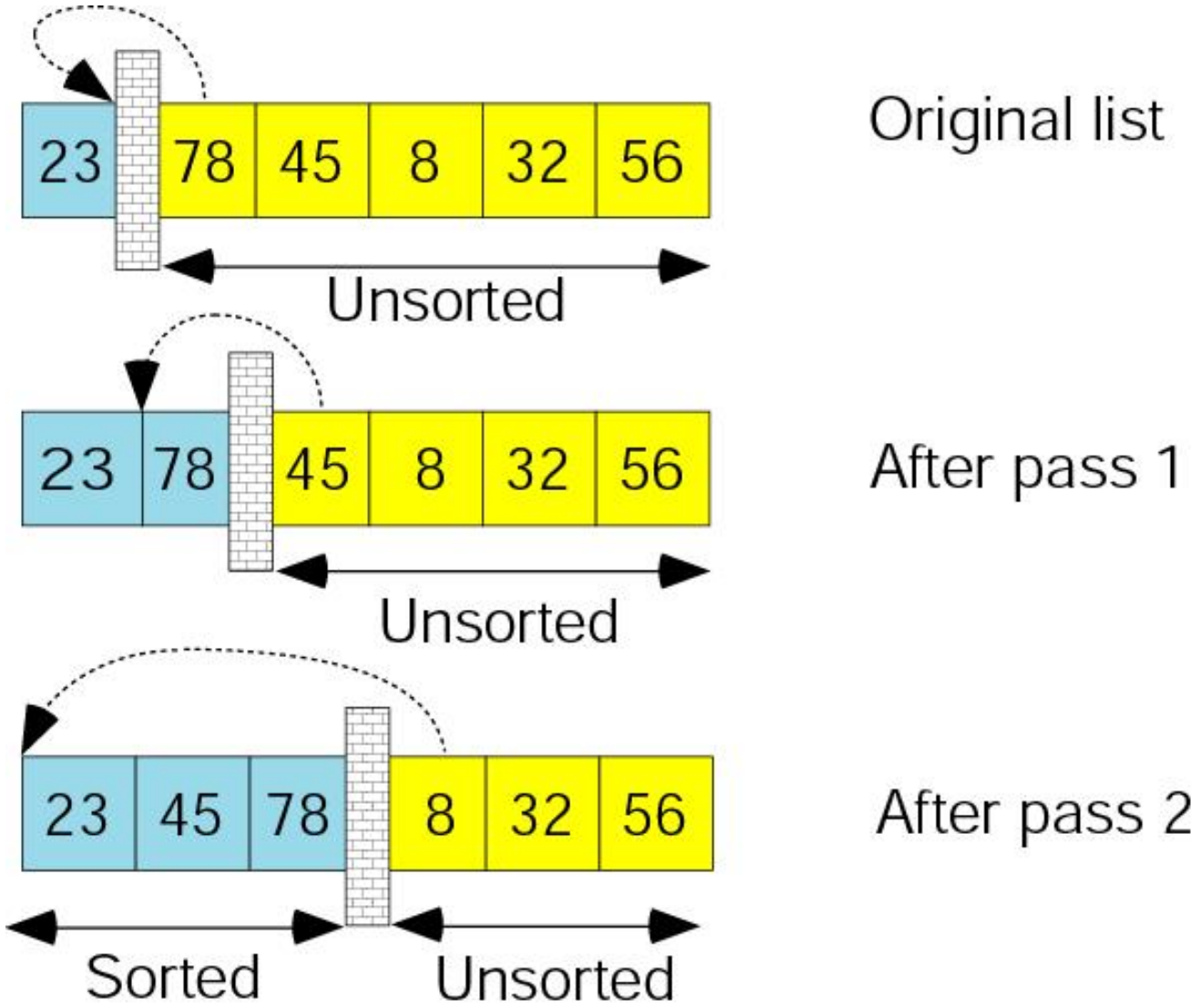
Figure 8-17



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Example of insertion sort

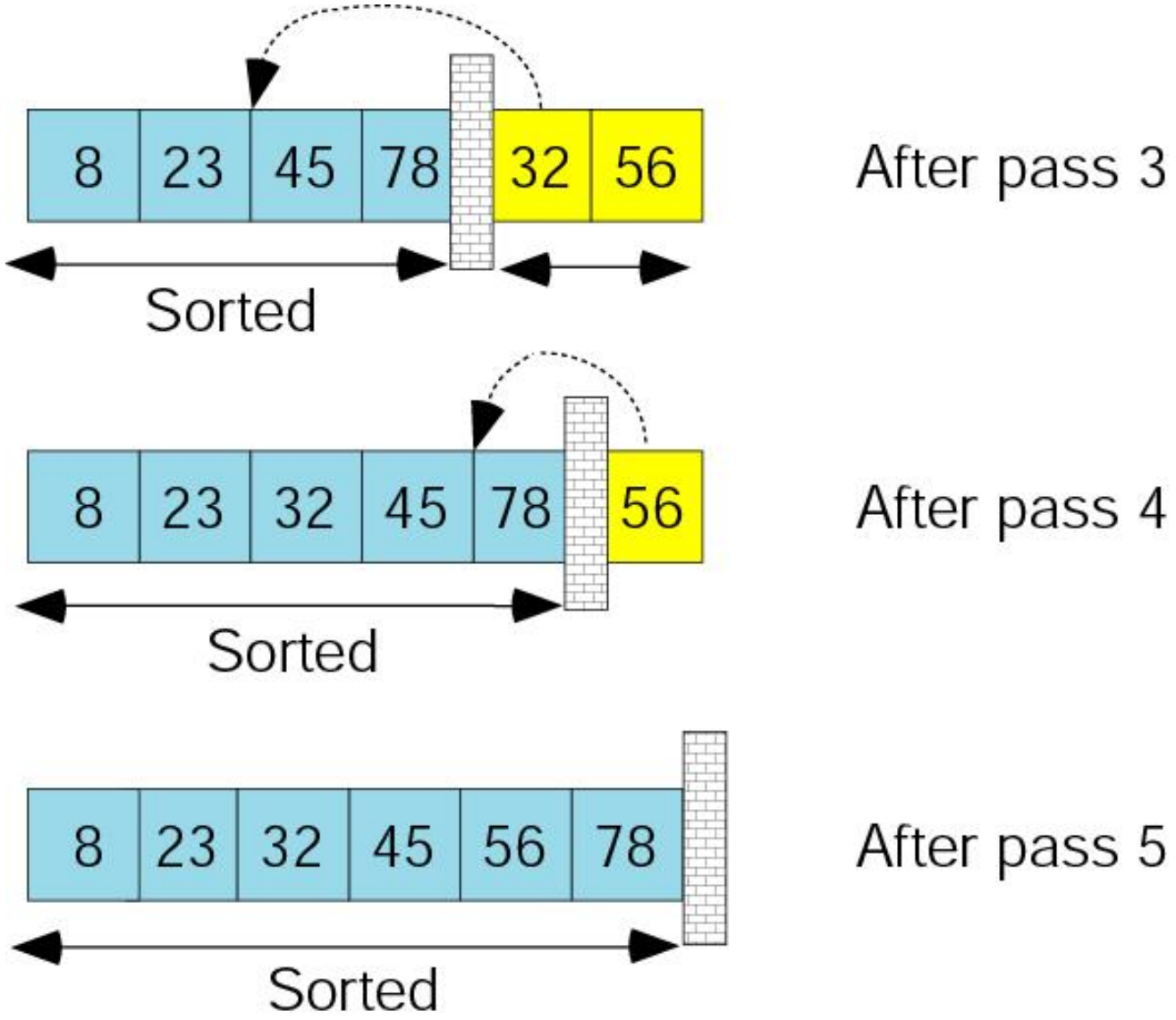
Figure 8-18: part I



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Example of insertion sort

Figure 8-18: part II



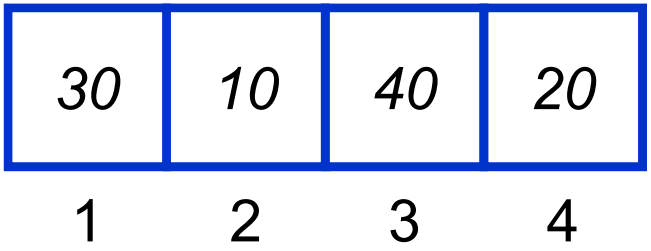
Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Pseudo code: Insertion Sort

```
/* Pseudo code: not an actual code,
   index starts from 1 */
InsertionSort(A, n) {
  for i = 2 to n {
      key = A[i]
      j = i - 1;
      while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
      }
      A[j+1] = key
  }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



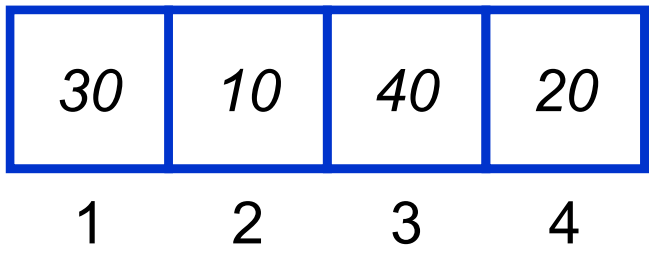
$i = \varnothing \quad j = \varnothing \quad key = \varnothing$

$A[j] = \varnothing \qquad A[j+1] = \varnothing$

```
InsertionSort(A, n) {
    for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
        }
        A[j+1] = key
    }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

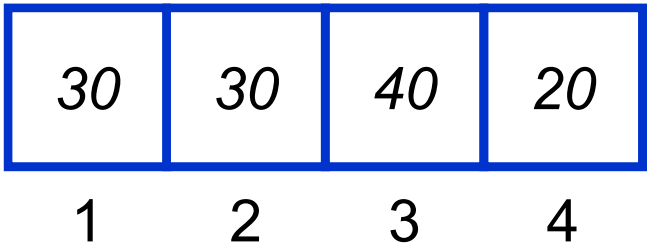


i = 2	j = 1	key = 10
A[j] = 30		A[j+1] = 10

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

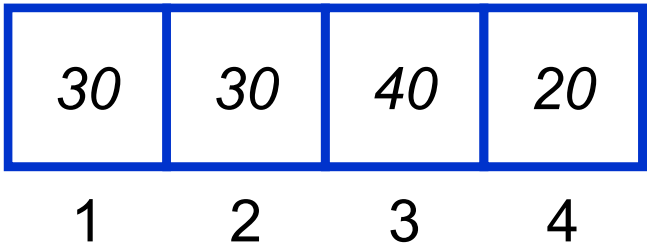


i = 2	j = 1	key = 10
A[j] = 30		A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



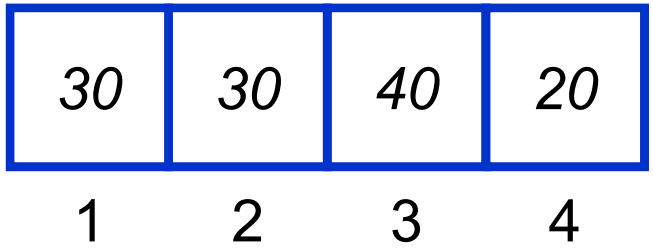
$i = 2 \quad j = 1 \quad key = 10$

$A[j] = 30 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

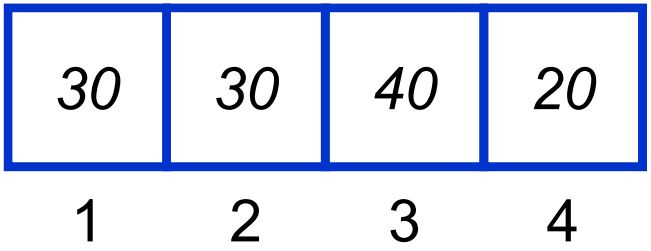


$i = 2 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \qquad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

An Example: Insertion Sort

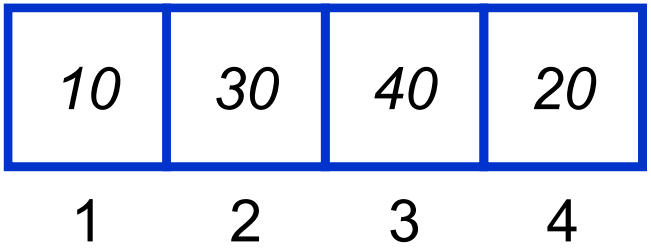


$i = 2 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \qquad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

An Example: Insertion Sort

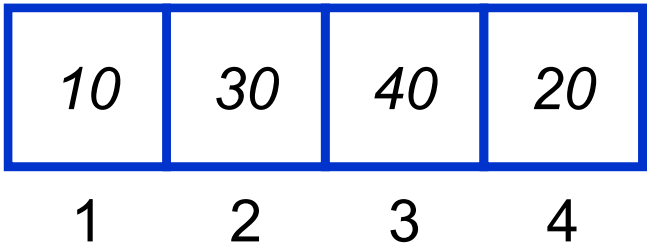


$i = 2 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \qquad A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

An Example: Insertion Sort



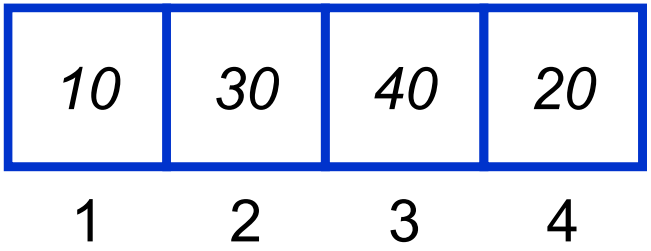
$i = 3 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \qquad A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



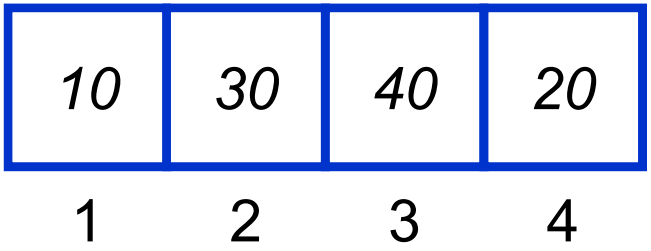
$i = 3 \quad j = 0 \quad key = 40$

$A[j] = \varnothing \qquad A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



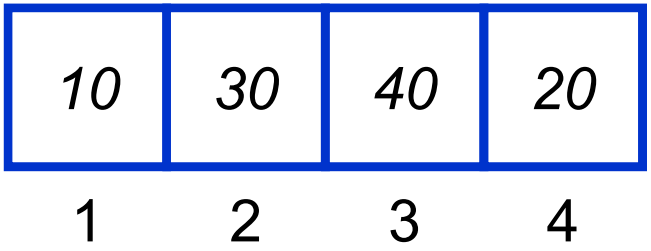
$i = 3 \quad j = 0 \quad key = 40$

$A[j] = \varnothing \qquad A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

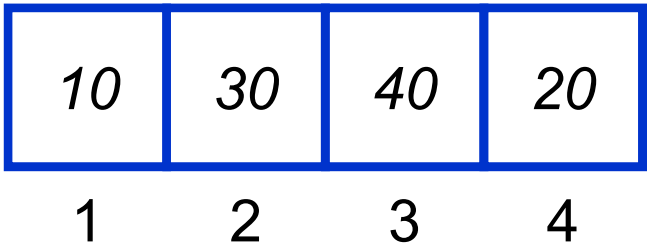


i = 3	j = 2	key = 40
A[j] = 30		A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

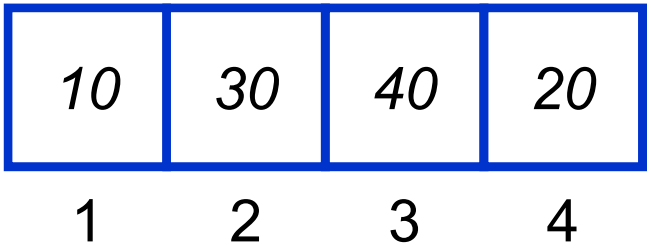


i = 3 j = 2	key = 40
A[j] = 30	A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

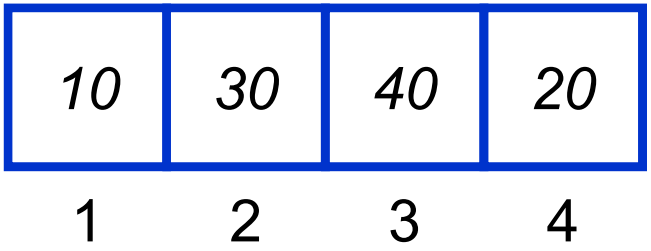


i = 3	j = 2	key = 40
A[j] = 30		A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

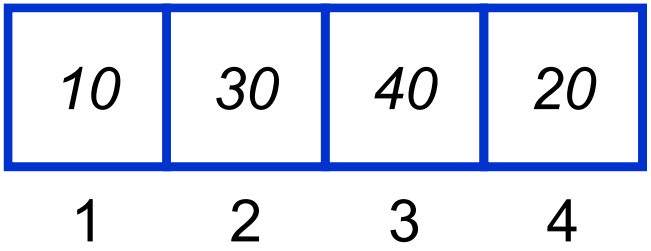


i = 4	j = 2	key = 40
A[j] = 30		A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



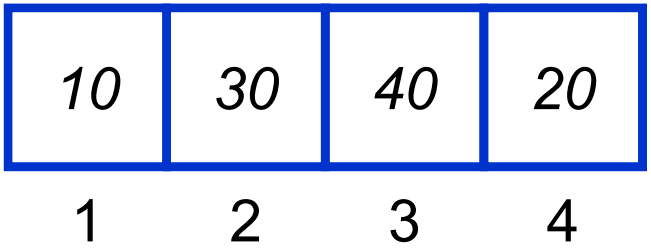
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



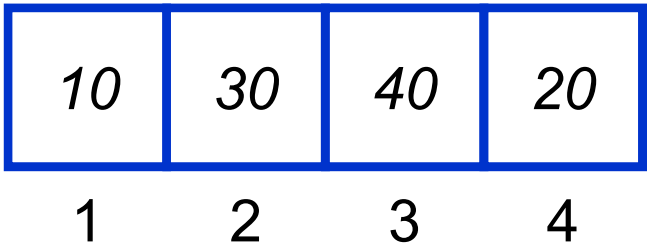
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



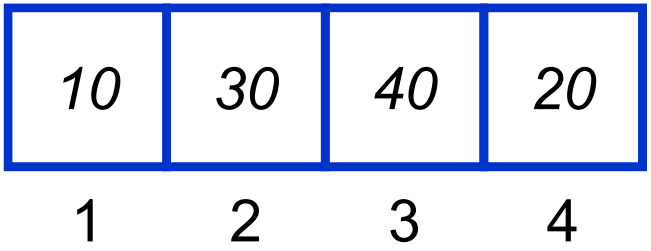
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 20$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



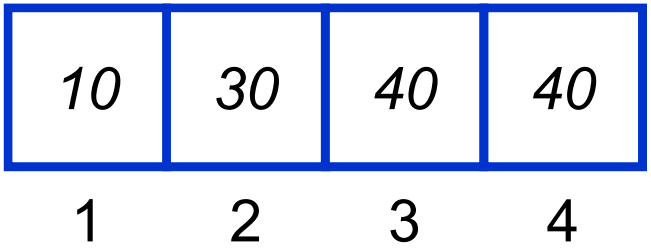
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 20$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



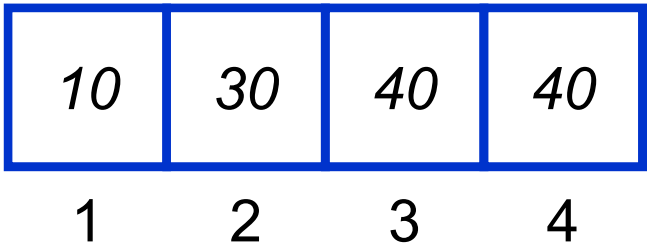
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



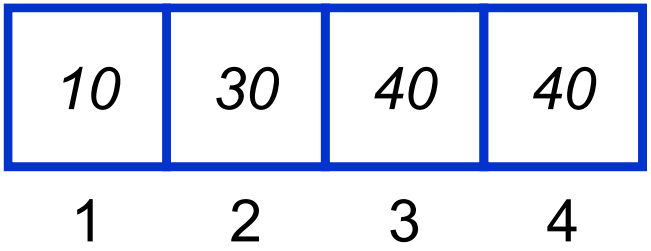
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



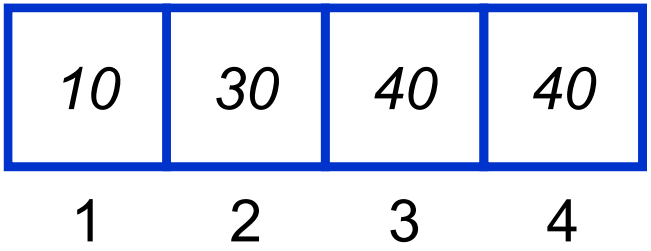
$i = 4 \quad j = 3 \quad key = 20$

$A[j] = 40 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

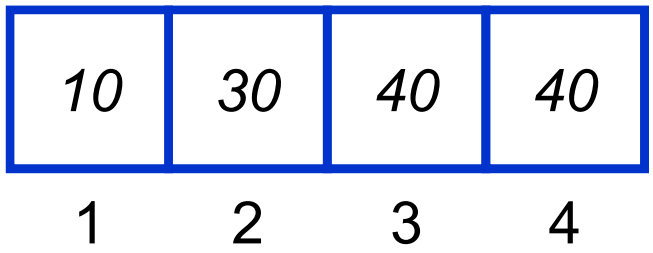


$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

An Example: Insertion Sort



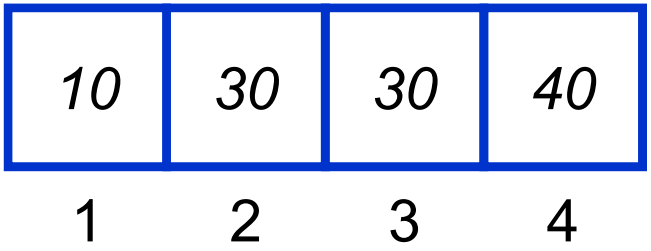
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

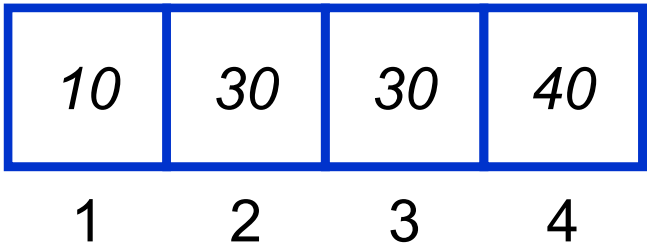


i = 4 j = 2	key = 20
A[j] = 30	A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



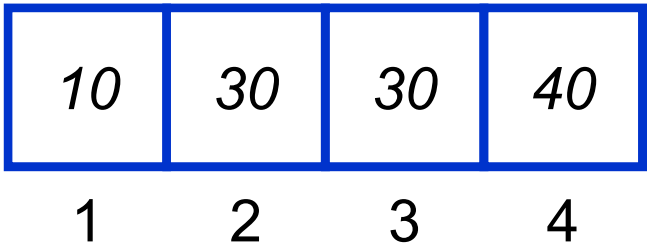
$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort

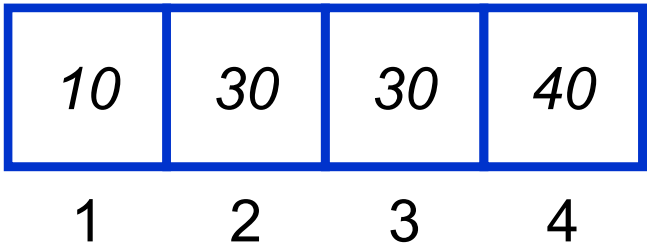


$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

An Example: Insertion Sort



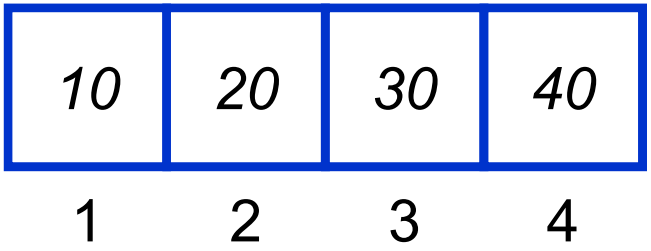
$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



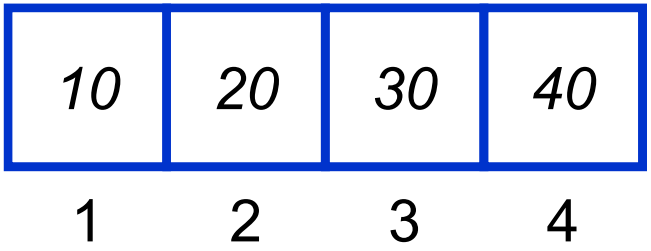
$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 20$

```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

An Example: Insertion Sort



$i = 4 \quad j = 1 \quad key = 20$

$A[j] = 10 \quad A[j+1] = 20$

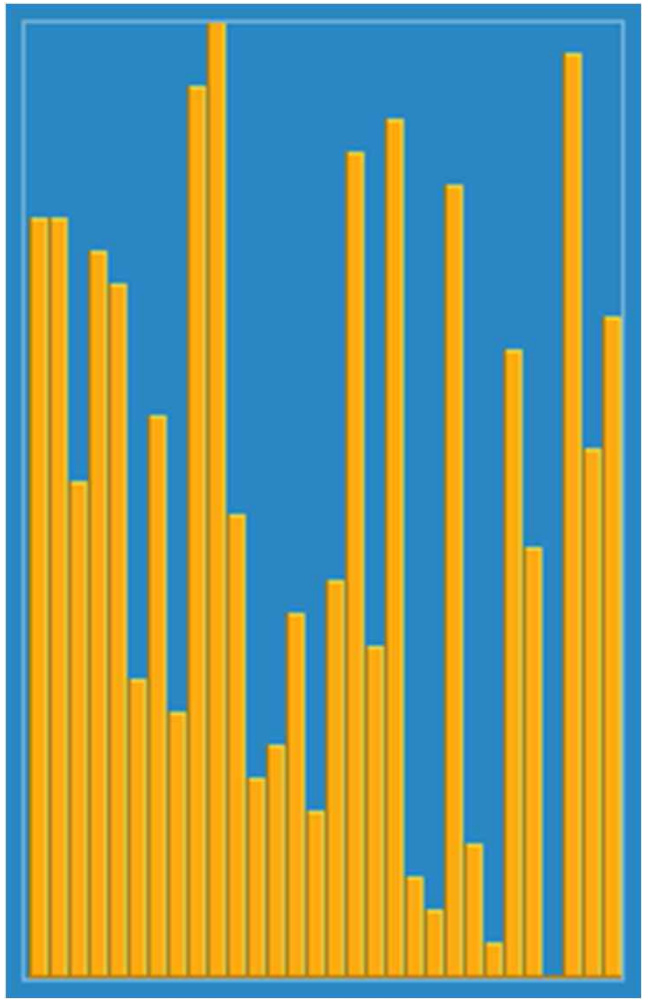
```
InsertionSort(A, n) {
   for i = 2 to n {
       key = A[i]
       j = i - 1;
       while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
       }
       A[j+1] = key
   }
}
```

Done!

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Animating Insertion Sort

6 5 3 1 8 7 2 4



Picture credit: Wikipedia, https://en.wikipedia.org/wiki/Insertion_sort

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Insertion Sort C code

```
/* source: https://ko.wikipedia.org/ */
void insertion_sort ( int *data, int n ) {
  int i, j, remember;
  for ( i = 1; i < n; i++ ) {
    remember = data[(j=i)];
    while ( --j >= 0 && remember < data[j] ) {
      data[j+1] = data[j];
      data[j] = remember;
    }
  }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

선택 정렬

SELECTION SORT

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

선택정렬 Selection Sort

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값을 선택하여 배열의 첫번째 요소와 교환
- 방법 1: 2개의 리스트 고려

리스트 1	리스트 2	설명
(5, 3, 8, 1, 2, 7)	()	초기 상태
(5, 3, 8, 2, 7)	(1)	리스트 1의 가장 작은 원소를 선택 => 리스트2로 이동
(5, 3, 8, 7)	(1, 2)	
(5, 8, 7)	(1, 2, 3)	
...	...	

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

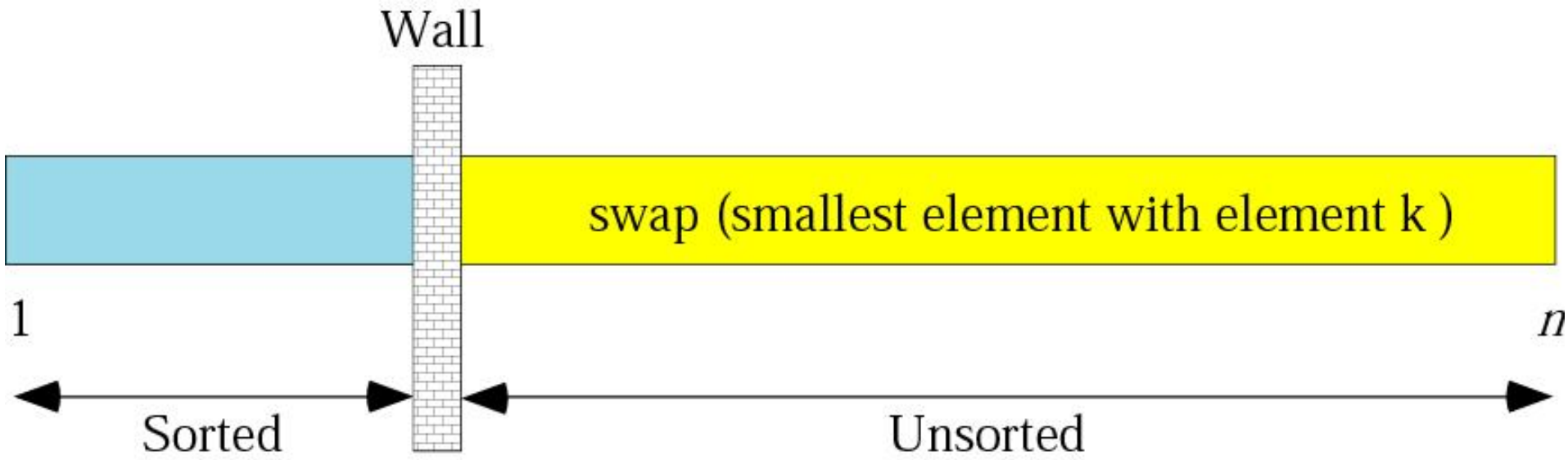
선택정렬(selection sort)

- 방법 2: 한 개의 리스트(배열)로 해결
  1. 가장 작은 원소를 첫번째 배열 원소와 교환
  2. 두번째 작은 원소를 두번째 배열 원소와 교환
  3. …

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

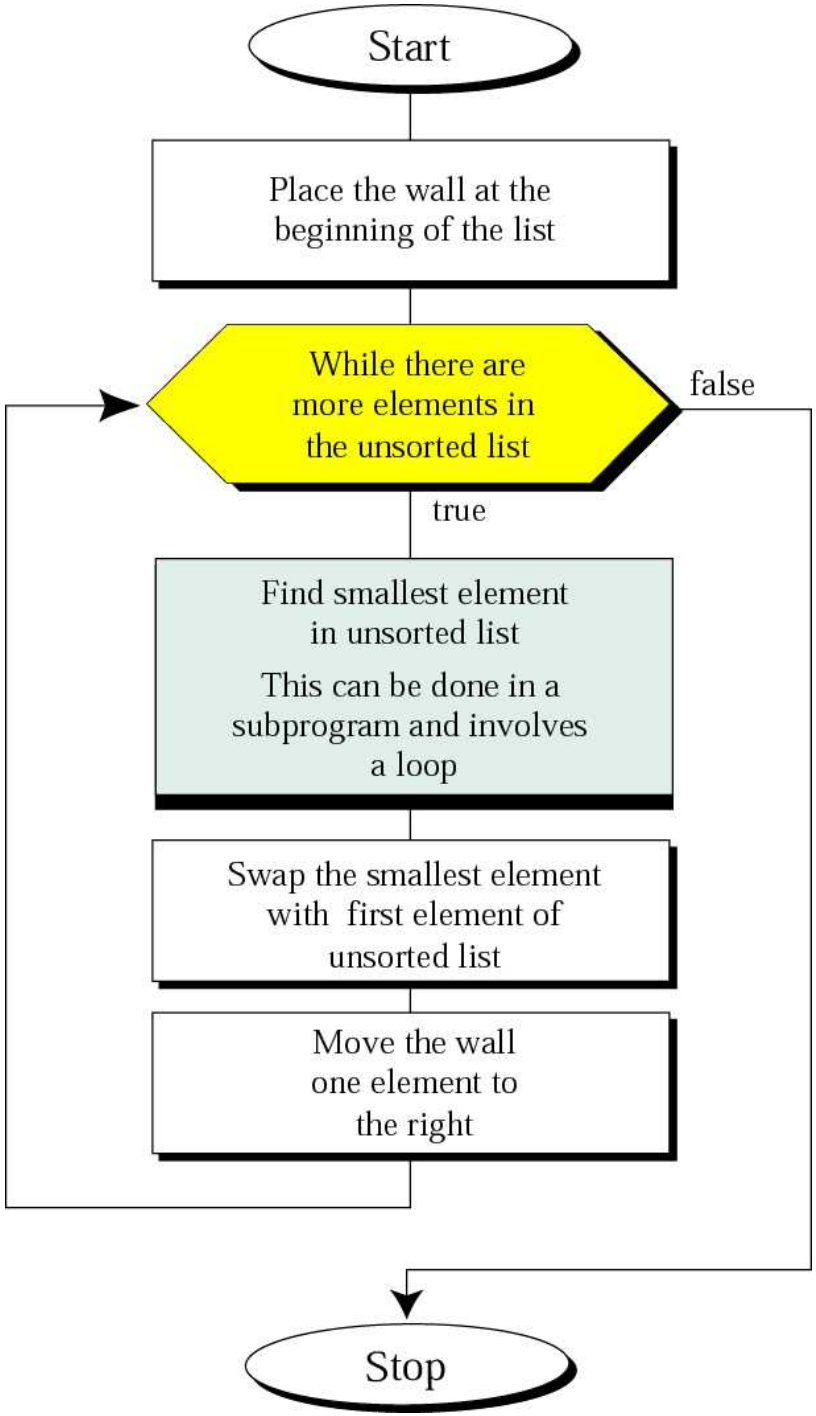
Selection sort

Figure 8-12



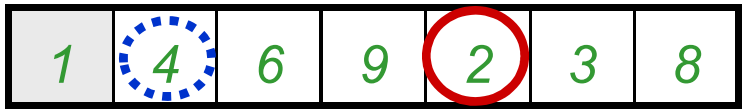
Selection sort algorithm

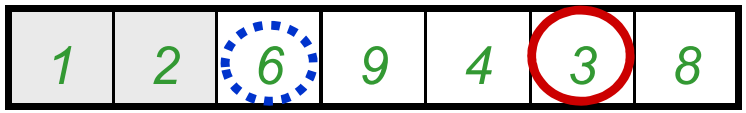
Figure 8-14

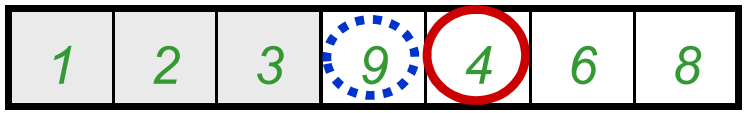


Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Example







정렬할 배열이 주어짐

가장 작은 수를 찾는다 (1)

1을 맨 왼쪽 수(8)와 자리 바꾼다

맨 왼쪽 수를 제외한 나머지에서 가장 큰 수를 찾는다 (2)

2를 맨 왼쪽 수(4)와 자리 바꾼다

맨 왼쪽 두 수를 제외한 나머지에서 가장 큰 수를 찾는다 (3)

. . .

정렬이 완료된 최종 배열

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Selection Sort

Alg.: SELECTION-SORT*(A)*

```
n ← length[A]
for j ← 1 to n - 1
    do smallest ← j
        for i ← j + 1 to n
            do if A[i] < A[smallest]
                then smallest ← i
        exchange A[j] ↔ A[smallest]
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Analysis of Selection Sort

Alg.: SELECTION-SORT(*A*)	*cost*	*times*
n ← length[*A*]	c_1	1
for j ← 1 to n - 1	c_2	n
do smallest ← j	c_3	n-1
for i ← j + 1 to n	c_4	$\sum_{j=1}^{n-1}(n-j+1)$
do if *A*[i] < *A*[smallest]	c_5	$\sum_{j=1}^{n-1}(n-j)$
then smallest ← i	c_6	$\sum_{j=1}^{n-1}(n-j)$
exchange *A*[j] ↔ *A*[smallest]	c_7	n-1

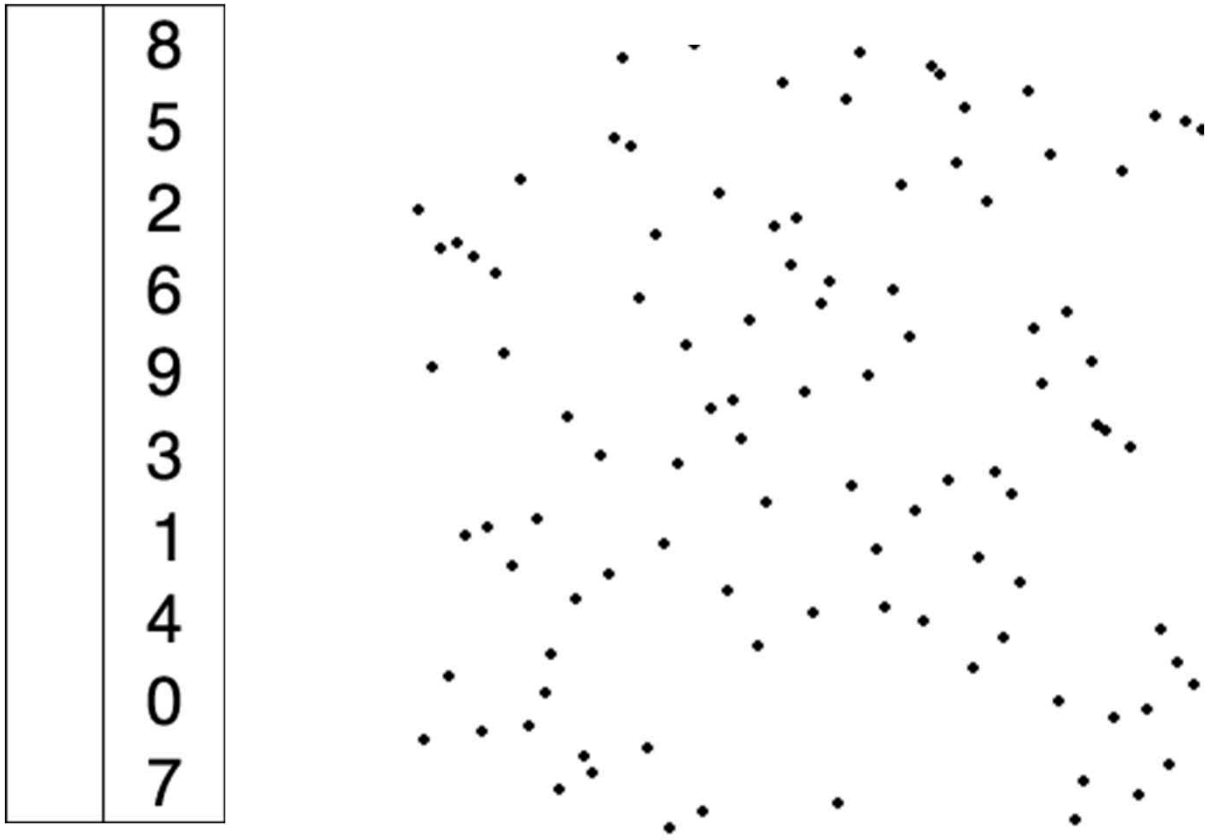
≈$n^2/2$ comparisons

≈n exchanges

$$\begin{aligned} T(n) &= c_1 + c_2 n + c_3(n-1) + c_4 \textstyle\sum_{j=1}^{n-1}(n-j+1) \\ &\quad + c_5 \textstyle\sum_{j=1}^{n-1}(n-j) + c_6 \sum_{j=2}^{n-1}(n-j) + c_7(n-1) \\ &\propto n^2 \end{aligned}$$

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Animating Selection Sort



Picture credit: Wikipedia, https://en.wikipedia.org/wiki/Selection_sort

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Selection Sort C code

```
/* source: https://ko.wikipedia.org/ */
void selectionSort(int *list, const int n) {
  int i, j, indexMin, temp;
  for (i = 0; i < n - 1; i++) {
    indexMin = i;
    for (j = i + 1; j < n; j++) {
      if (list[j] < list[indexMin]) {
        indexMin = j;
      }
    }
    temp = list[indexMin];
    list[indexMin] = list[i];
    list[i] = temp;
  }
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

거품 정렬

BUBBLE SORT

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

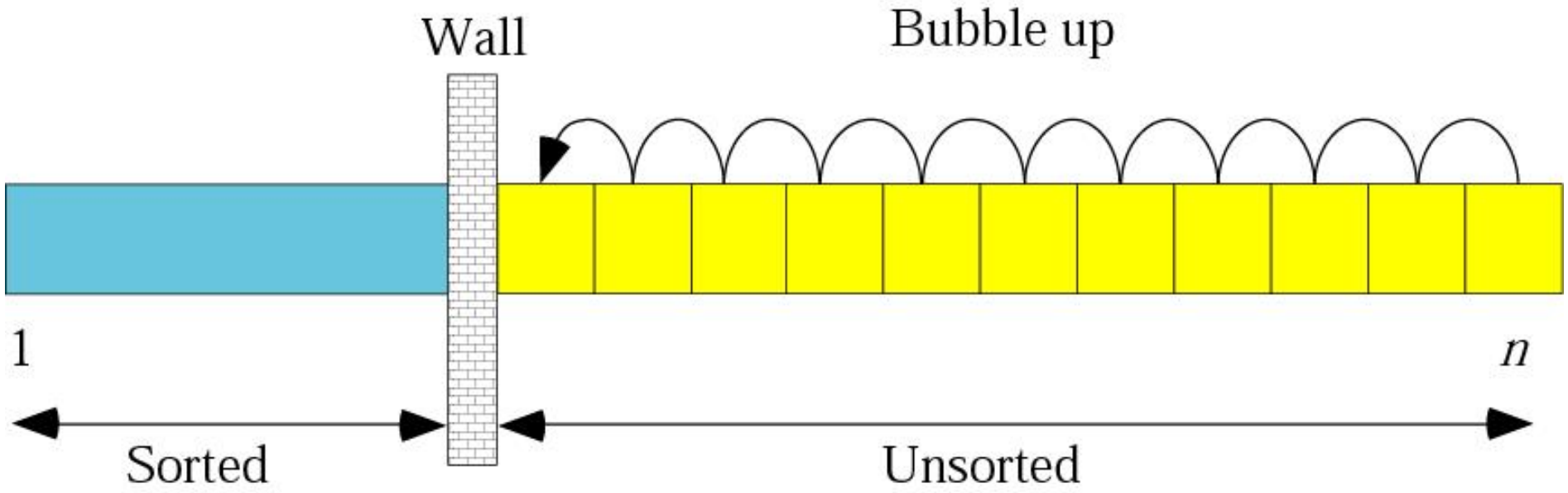
거품정렬(bubble sort)

- 인접한 레코드가 순서대로 되어 있지않으면 교환
- 전체가 정렬될때까지 비교/교환계속

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Bubble sort

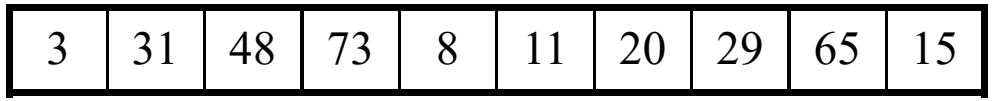
Figure 8-15



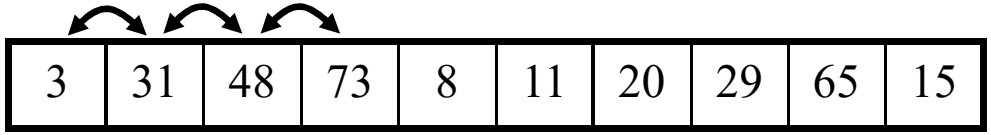
Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

버블정렬의 작동 예

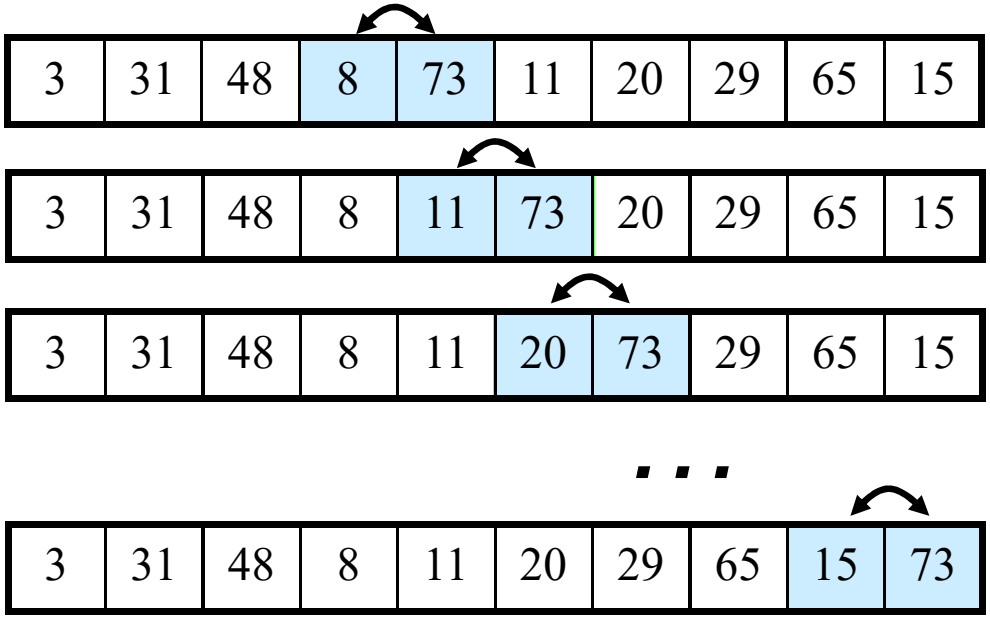
정렬할 배열이 주어진다



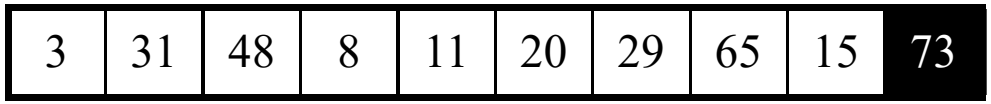
왼쪽부터 시작해 이웃한 쌍들을 비교해 나간다



순서대로 되어 있지 않으면 자리 바꾼다

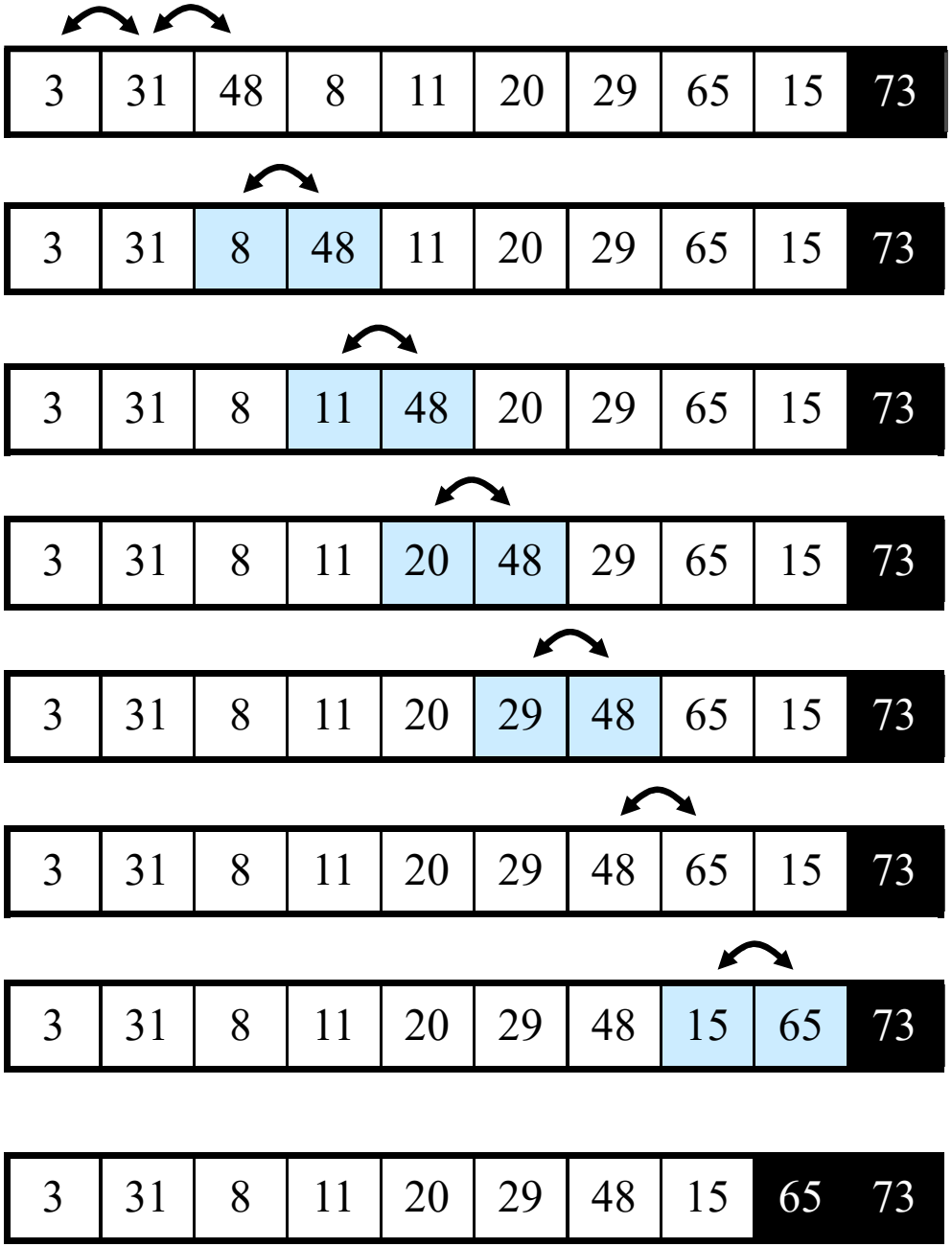


맨 오른쪽 수(73)를 대상에서 제외한다



Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

버블정렬의 작동 예



왼쪽부터 시작해 이웃한 쌍들을 비교해간다

순서대로 되어 있지 않은 경우에는 자리 바꾼다

맨 오른쪽 수(65)를 대상에서 제외한다

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

버블정렬의 작동 예

앞의 작업을 반복하면서 계속 제외해 나간다

. . .

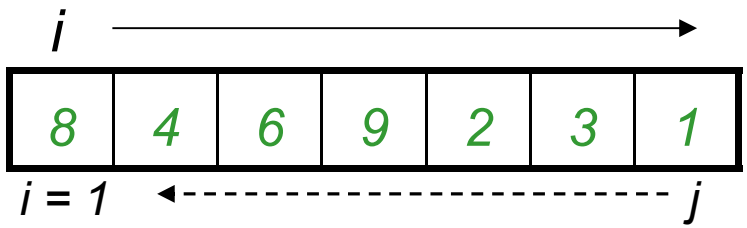
두개짜리 배열의 처리를 끝으로 정렬이 완료된다

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Bubble Sort

Alg.: BUBBLESORT(A)

for i ← 1 **to** length[A]
  do for j ← length[A] **downto** i + 1
    do if A[j] < A[j -1]
      then exchange A[j] ↔ A[j-1]



Easier to implement, but slower than Insertion sort

Bubble-Sort Running Time

Alg.: BUBBLESORT(A)

for $i \leftarrow 1$ **to** length[A] c_1

do for $j \leftarrow$ length[A] **downto** $i + 1$ c_2

Comparisons: $\approx n^2/2$ **do if** $A[j] < A[j-1]$ c_3

Exchanges: $\approx n^2/2$ **then** exchange $A[j] \leftrightarrow A[j-1]$ c_4

$$
\begin{aligned}
T(n) &= c_1(n+1) + c_2\sum_{j=1}^{n}(n-j+1) \\
&\quad + c_3\sum_{j=1}^{n}(n-j) + c_4\sum_{j=1}^{n}(n-j) \\
&\approx c_1 n + (c_2 + c_3 + c_4)\sum_{j=1}^{n}(n-j) \\
&= c_1 n + (c_2 + c_3 + c_4)\cdot\frac{n(n-1)}{2} \\
&\propto n^2
\end{aligned}
$$

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Animating Bubble Sort

Picture credit: Wikipedia, https://en.wikipedia.org/wiki/Selection_sort

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Bubble Sort C code

```
/* source: https://ko.wikipedia.org/ */
int bubble_sort(int arr[], int n) {
  int i, j, temp;
  for (i=n-1; i>0; i--) {
    for (j=0; j<i; j++) {
      if (arr[j] > arr[j+1]) {
        temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
      }
    }
  }
  return *arr;
}
```

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Sorting Algorithm Comparison

- Insertion/Selection/Bubble sort
  - Advantages: easy to understand, by increasing SORTED REGION one item at a time
  - Disadvantages: *n*^2에 비례하는 연산량
    - $T(n) = T(n-1)+cn$ ➔ O(*n*^2) … next class
- *Merge sort*
- *Quicksort*
- *Heapsort*

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I

Simple sorting methods

END OF LECTURE 3

Slide credit: J. Lillis, UIC's CS 201 Data Structures and Discrete Mathematics I