

Sorting

2020 Spring: AP Computer Science A

February 5th, 2020

Today

- **Sorting**
- **Selection sort**
- **Bubble sort**
- **Insertion sort**
- **Merge sort**
- **Quick sort**

- **Sorting Java objects**

Sorting

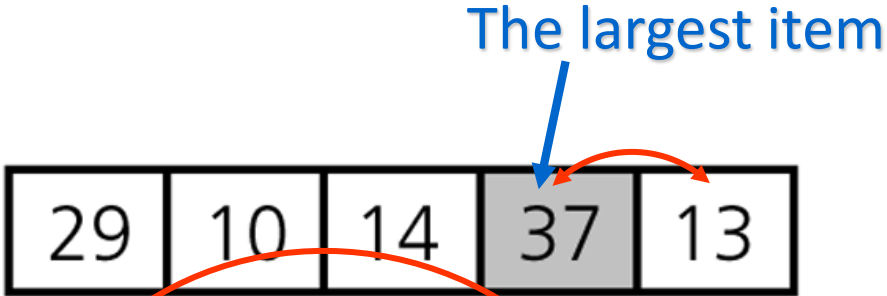
- We will try to sort an array of integers in increasing order
- Given an array of integers of size n , the returned array must satisfy
 - $\text{arr}[0] \leq \text{arr}[1] \leq \dots \leq \text{arr}[n-1]$

Selection Sort

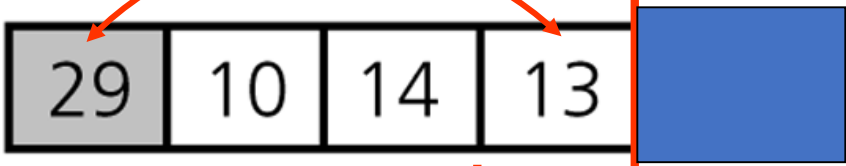
- 각 루프마다
 - 최대 원소를 찾는다
 - 최대 원소와 맨 오른쪽 원소를 교환한다
 - 맨 오른쪽 원소를 제외한다
- 하나의 원소만 남을 때까지 위의 루프를 반복

Finding the Recursive Structure

Initial array:



After 1st swap:



After 2nd swap:



After 3rd swap:



After 4th swap:



✓ 수행시간: $(n - 1) + (n - 2) + \dots + 2 + 1 = \Theta(n^2)$
 Worst case
 Average case

```

selectionSort(A[], n)      ▷ 배열 A[1 ... n]을 정렬한다
{
    for last ← n downto 2 { ----- ①
        A[1 ... last] 중 가장 큰 수 A[k]를 찾는다; ----- ②
        A[k] ↔ A[last]; ▷ A[k]와 A[last]의 값을 교환 ----- ③
    }
}

```

✓ 수행시간:

- ①의 **for** 루프는 $n - 1$ 번 반복
- ②에서 가장 큰 수를 찾기 위한 비교횟수: $n - 1, n - 2, \dots, 2, 1$
- ③의 교환은 상수 시간 작업

✓ $(n - 1) + (n - 2) + \dots + 2 + 1 = \Theta(n^2)$

Selection Sort의 작동 예

정렬할 배열이 주어짐

8	31	48	73	3	65	20	29	11	15
---	----	----	----	---	----	----	----	----	----

가장 큰 수를 찾는다 (73)

8	31	48	73	3	65	20	29	11	15
---	----	----	----	---	----	----	----	----	----

73을 맨 오른쪽 수(15)와 자리 바꾼다

8	31	48	15	3	65	20	29	11	73
---	----	----	----	---	----	----	----	----	----

①의 첫번째 loop

맨 오른쪽 수를 제외한 나머지에서 가장 큰 수를 찾는다 (65)

8	31	48	15	3	65	20	29	11	73
---	----	----	----	---	----	----	----	----	----

65를 맨 오른쪽 수(11)와 자리 바꾼다

8	31	48	15	3	11	20	29	65	73
---	----	----	----	---	----	----	----	----	----

①의 두번째 loop

맨 오른쪽 두 수를 제외한 나머지에서 가장 큰 수를 찾는다 (48)

8	31	48	15	3	11	20	29	65	73
---	----	----	----	---	----	----	----	----	----

...

8을 맨 오른쪽 수(3)와 자리 바꾼다

8	3	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

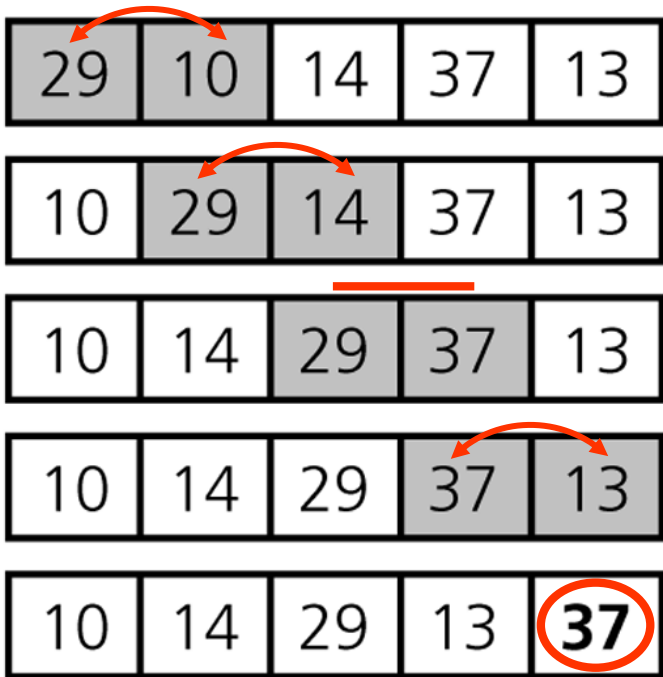
최종 배열

3	8	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

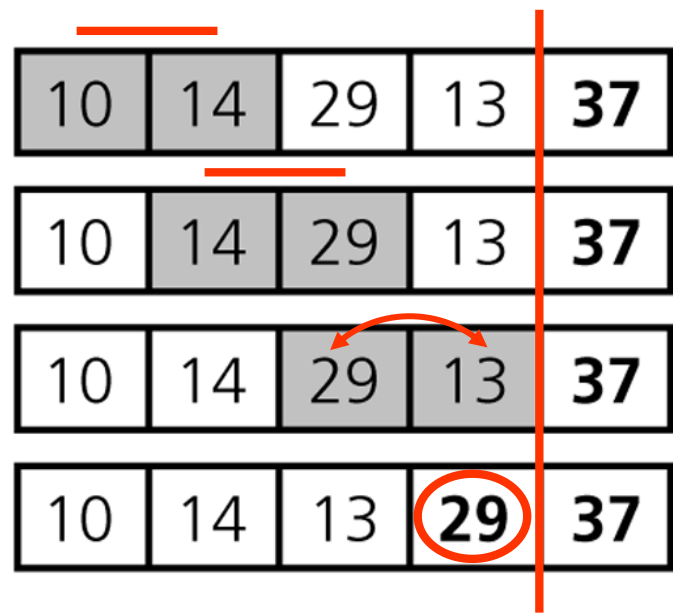
Bubble Sort

(a) Pass 1

Initial array:



(b) Pass 2



✓ 수행시간: $(n - 1) + (n - 2) + \dots + 2 + 1 = \Theta(n^2)$
 Worst case
 Average case


```

bubbleSort(A[], n)    ▷ A[1 ... n]을 정렬한다
{
    for last ← n downto 2 ----- ①
        for i ← 1 to last-1 ----- ②
            if (A[i] > A[i+1])
                then A[i] ↔ A[i+1]; ▷ 원소 교환 -- ③
}

```

✓ 수행시간:

- ①의 **for** 루프는 $n - 1$ 번 반복
- ②의 **for** 루프는 각각 $n - 1, n - 2, \dots, 2, 1$ 번 반복
- ③은 상수 시간 작업

✓ $(n - 1) + (n - 2) + \dots + 2 + 1 = \Theta(n^2)$

Bubble Sort의 작동 예

정렬할 배열이 주어짐

3	31	48	73	8	11	20	29	65	15
---	----	----	----	---	----	----	----	----	----

왼쪽부터 시작해 이웃한 쌍들을 비교해간다

3	31	48	73	8	11	20	29	65	15
---	----	----	----	---	----	----	----	----	----

순서대로 되어 있지 않으면 자리 바꾼다

3	31	48	8	73	11	20	29	65	15
---	----	----	---	----	----	----	----	----	----

3	31	48	8	11	73	20	29	65	15
---	----	----	---	----	----	----	----	----	----

3	31	48	8	11	20	73	29	65	15
---	----	----	---	----	----	----	----	----	----

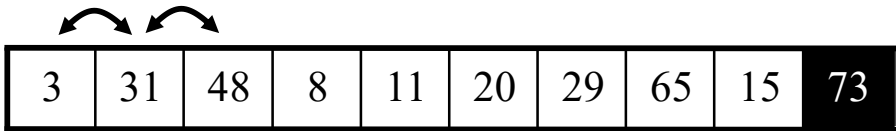
...

3	31	48	8	11	20	29	65	15	73
---	----	----	---	----	----	----	----	----	----

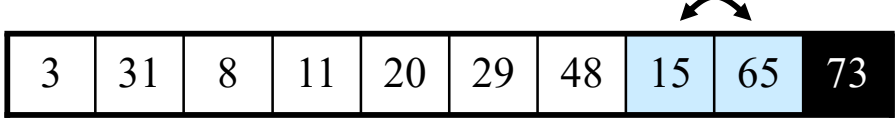
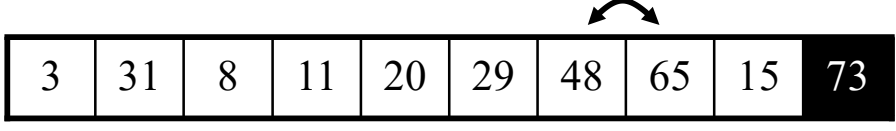
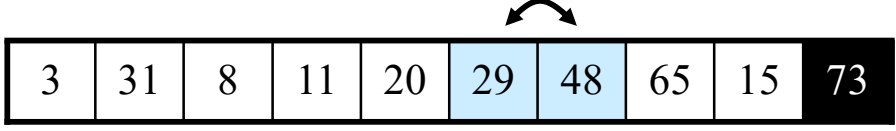
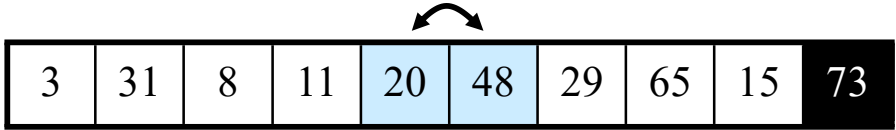
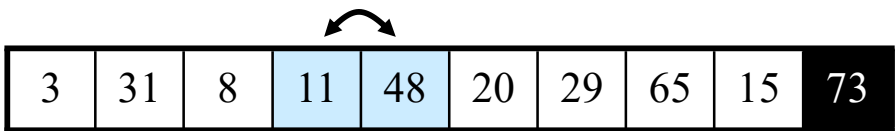
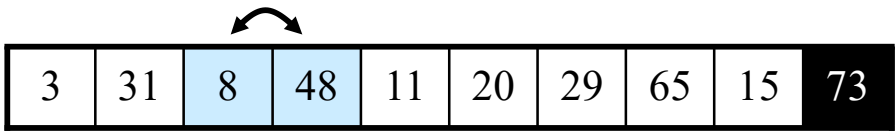
맨 오른쪽 수(73)를 대상에서 제외한다

3	31	48	8	11	20	29	65	15	73
---	----	----	---	----	----	----	----	----	----

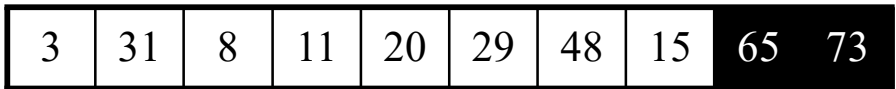
왼쪽부터 시작해 이웃한 쌍들을 비교해간다



순서대로 되어 있지 않은 경우에는 자리 바꾼다



맨 오른쪽 수(65)를 대상에서 제외한다



앞의 작업을 반복하면서 계속 제외해 나간다

...

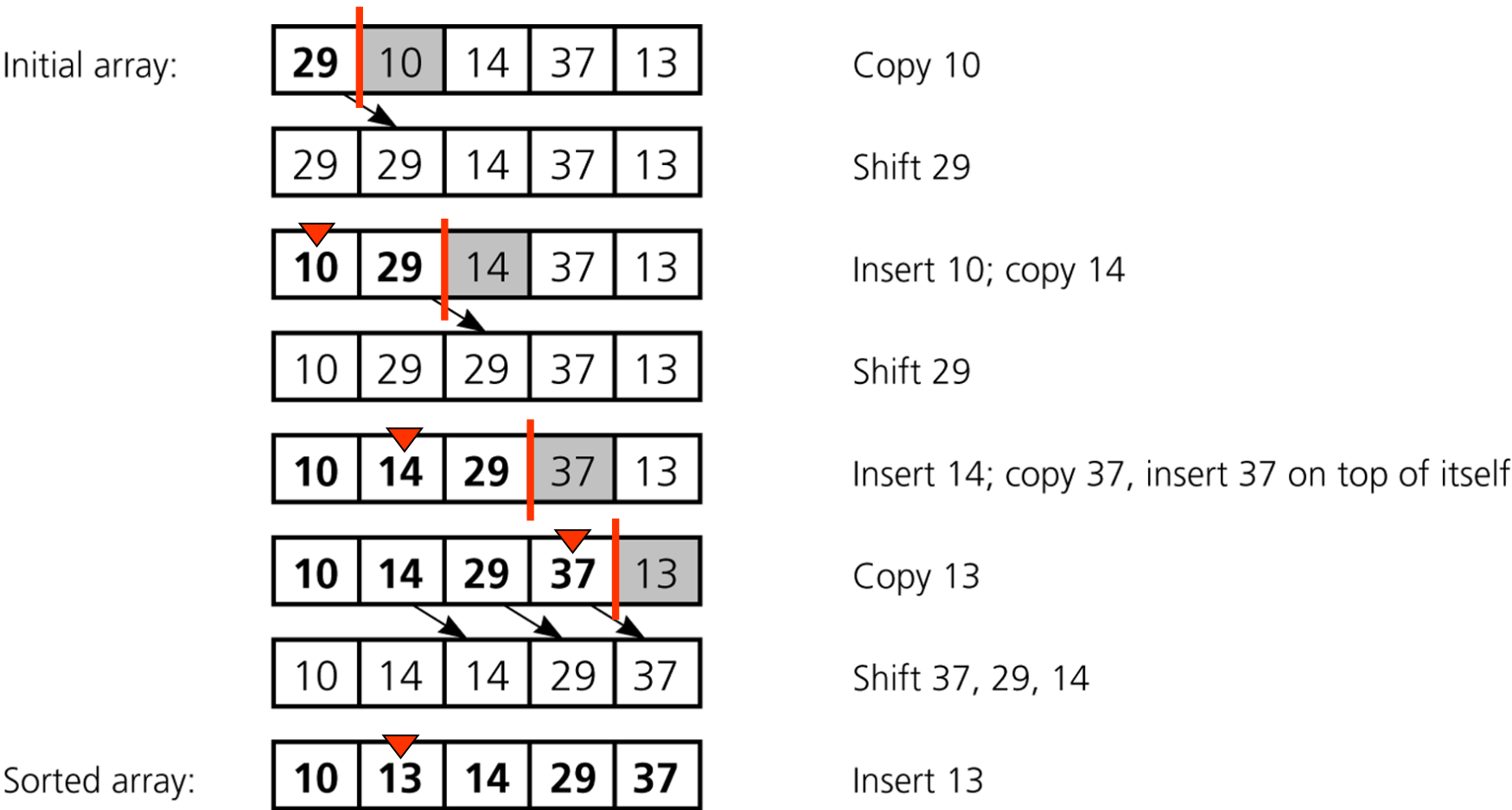
3	8	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

두개짜리 배열의 처리를 끝으로 정렬이 완료된다

3	8	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

3	8	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

Insertion Sort



✓ 수행시간: $\Theta(n^2)$

Worst case: $1 + 2 + \dots + (n - 2) + (n - 1)$

Average case: $\frac{1}{2} (1 + 2 + \dots + (n - 2) + (n - 1))$

insertionSort(A[], n) \triangleright A[1 ... n]을 정렬한다

```
{
    for i ← 2 to n ----- ①
        A[1 ... i]의 적당한 자리에 A[i]를 삽입한다; ---- ②
}
```

✓ 수행시간:

- ①의 **for** 루프는 $n - 1$ 번 반복
- ②의 삽입은 최악의 경우 $i - 1$ 회 비교

✓ Worst case: $1 + 2 + \dots + (n - 2) + (n - 1) = \Theta(n^2)$

✓ Average case: $\frac{1}{2} (1 + 2 + \dots + (n - 2) + (n - 1)) = \Theta(n^2)$

Merge Sort

`mergeSort(A[], p, r)` ▷ `A[p ... r]`을 정렬한다

```
{
    if (p < r) then {
        q ← (p+r)/2; ----- ①    ▷ p, r의 중간 지점 계산
        mergeSort(A, p, q); ----- ②    ▷ 전반부 정렬
        mergeSort(A, q+1, r); ----- ③    ▷ 후반부 정렬
        merge(A, p, q, r); ----- ④    ▷ 병합
    }
}
```

```
merge(A[ ], p, q, r) {
    정렬되어 있는 두 배열 A[p ... q]와 A[q+1 ... r]을 합하여
    정렬된 하나의 배열 A[p ... r]을 만든다.
}
```

Mergesort의 작동 예

정렬할 배열이 주어짐

31	3	65	73	8	11	20	29	48	15
----	---	----	----	---	----	----	----	----	----

배열을 반반으로 나눈다

31	3	65	73	8	11	20	29	48	15
----	---	----	----	---	----	----	----	----	----

— (1)

각각 독립적으로 정렬한다

3	8	31	65	73	11	15	20	29	48
---	---	----	----	----	----	----	----	----	----

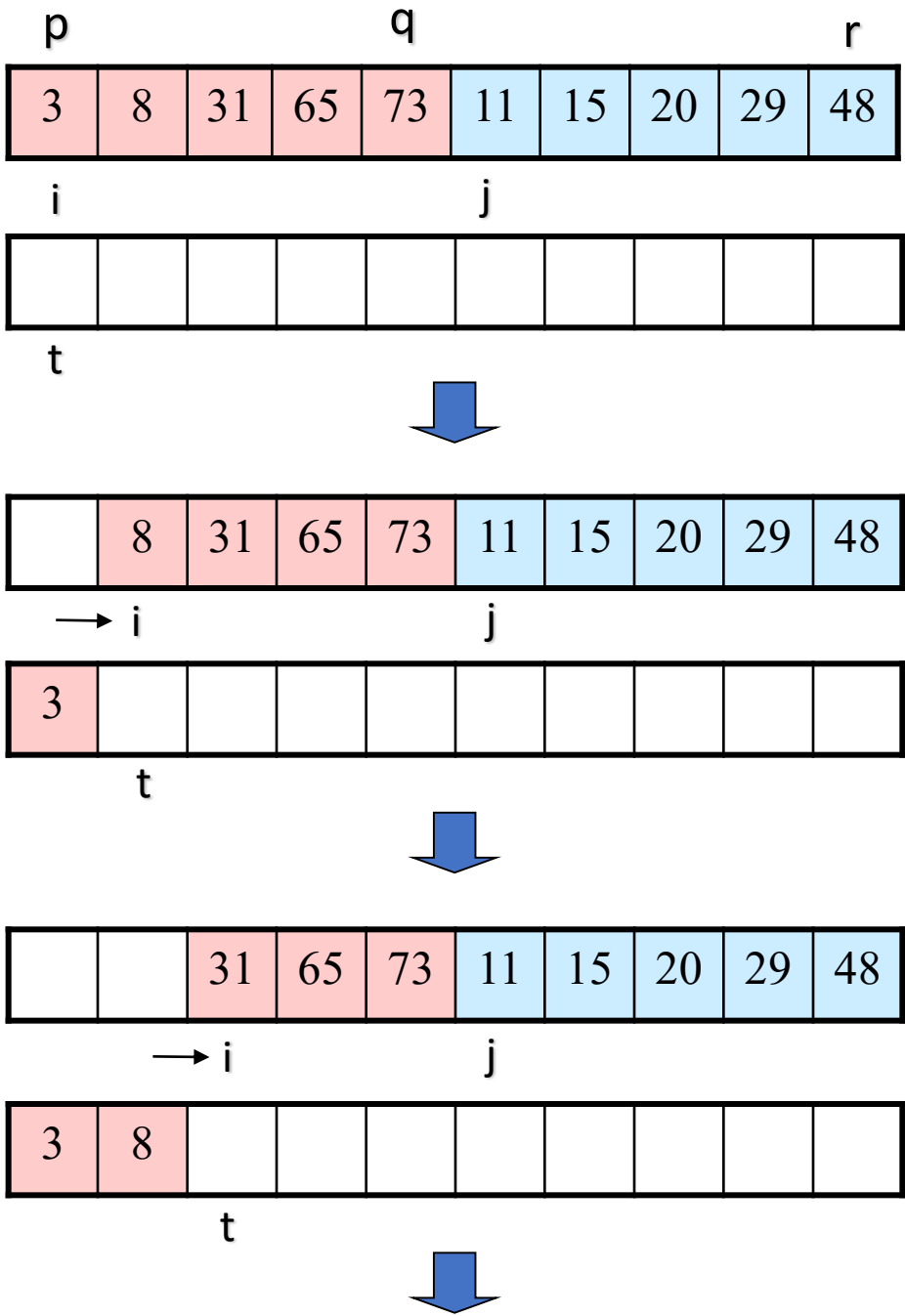
— (2) (3)

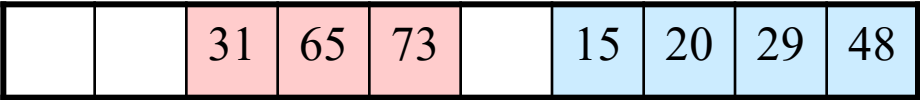
병합한다 (정렬완료)

3	8	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

— (4)

Merge의 작동 예

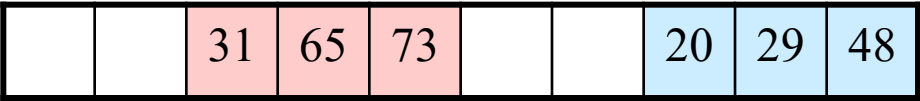




i → j



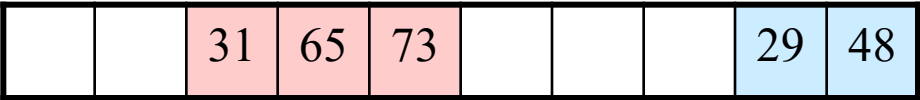
t



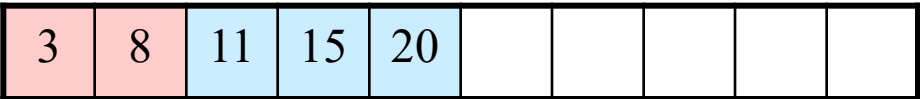
i → j



t

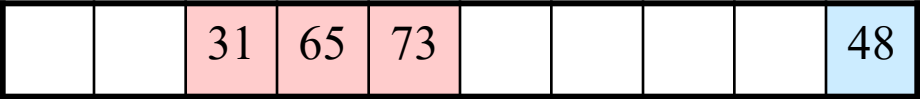


i → j

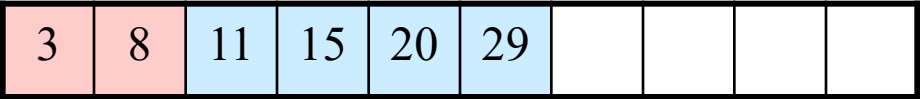


t

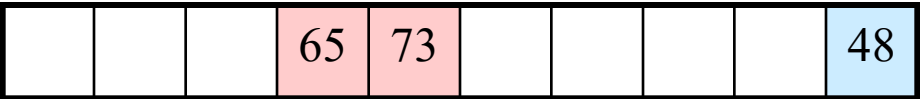




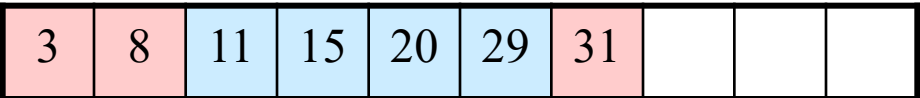
i → j



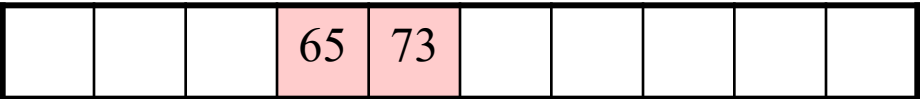
t



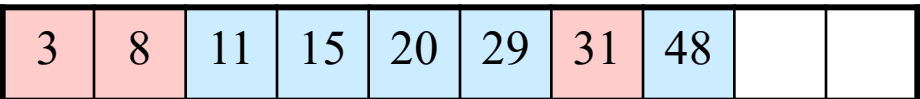
→ i j



t

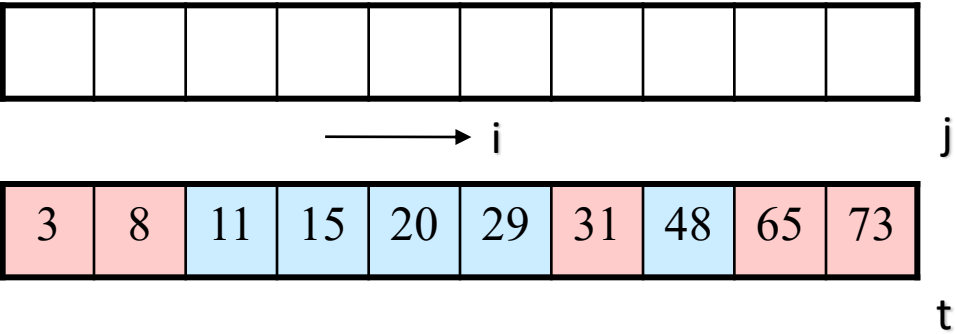


i → j




t





Animation (Mergesort)



1 2 3 4 6 7 8 9

✓ 수행시간: $\Theta(n \log n)$

Quicksort

```

quicksort(A[], p, r)  ▷ A[p ... r]을 정렬한다
{
    if (p < r) then {
        q = partition(A, p, r);  ▷ 분할
        quicksort(A, p, q-1);    ▷ 왼쪽 부분배열 정렬
        quicksort(A, q+1, r);    ▷ 오른쪽 부분배열 정렬
    }
}

```

```

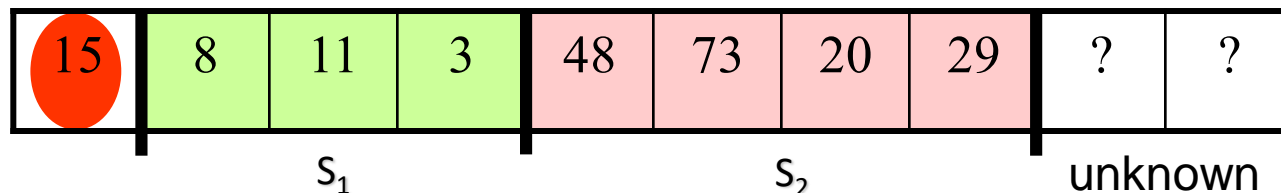
partition(A[], p, r) {
    배열 A[p ... r]의 원소들을
    A[p](또는 아무거나 하나)을 기준으로 양쪽으로 재배치하고
    A[p]가 자리한 위치를 return 한다;
}

```

```

partition(A[], p, r) {
    x ← A[p]; // pivot
    lastS1 ← p; // sets S1 & S2 are empty
    for i ← p+1 to r
        if (A[i] < x) {
            lastS1++;
            A[lastS1] ↔ A[i];
        }
    A[lastS1] ↔ A[p];
    return lastS1;
}

```



Animation (Quicksort)



1 2 3 4 5 6 8 9

- ✓ 평균 수행시간: $\Theta(n \log n)$
- ✓ 최악의 경우 수행시간: $\Theta(n^2)$

Quicksort의 작동 예

정렬할 배열이 주어짐. 첫번째 수를 기준으로 삼는다.

15	8	48	73	11	3	20	29	65	31
----	---	----	----	----	---	----	----	----	----

기준보다 작은 수는 기준의 왼쪽에 나머지는
기준의 오른쪽에 오도록 재배치한다

3	8	11	15	48	73	20	29	65	31
---	---	----	----	----	----	----	----	----	----

 — (a)

기준(31) 왼쪽과 오른쪽을 각각 독립적으로 정렬한다 (정렬완료)

3	8	11	15	20	29	31	48	65	73
---	---	----	----	----	----	----	----	----	----

 — (b)

Partition의 예

