Merge Sort

㈜한컴에듀케이션 이주현

Merge Sort

❖ 개요

- 합병 정렬 또는 병합 정렬이라고 번역된다.
- 폰 노이만(John von Neumann)이 1945년 개발.
- 원소들 간의 비교를 통하여 정렬하는 **비교기반정렬** 알고리즘.
- 원소들 중에 같은 값이 있는 경우 정렬 후에도

  이들의 순서가 유지되는 **안정 정렬**에 속한다.
- 정렬의 과정은 분할 -〉 정복 -〉 합병(결합, 병합) -〉 복사로 이루어진다.

  대표적인 분할 정복 알고리즘의 한 예이다.
- N개의 데이터를 정렬할 때, **시간복잡도는 O(N * logN)이 보장**된다.
- 데이터가 배열에 저장된 경우 N크기의 추가적인 배열이 필요하다.

Merge Sort

❖ 폰노이만(John von Neumann)

- 수학자, 물리학자, 발명가, 컴퓨터 공학자.
- 20세기의 수학자들 가운데 가장 중요한 인물로 거론되는 인물.

  인류사가 시작된 이래 가장 위대한 천재중 하나로,

  당대 그 어떤 수학자도 폰 노이만을 능가하는 자가 없었을 정도였다.
- **폰 노이만 구조** : 현재와 같은 CPU, 메모리,

  프로그램 구조를 갖는 범용 컴퓨터 구조의 확립.
- 게임이론의 창시자.
- 1903/12/28 ~ 1957/2/18 (향년 53년 52일)
- 헝가리 -〉 미국

Merge Sort

❖ 안정 정렬(Stable Sort)

이름	과목 수
Neumann	4
Turing	1
Euler	3
Gauss	1
Archimedes	5
Knuth	2
Newton	3

Merge Sort

❖ 안정 정렬(Stable Sort)

과목수가 같은 경우 초기 순서를 유지하고 있다.

이름	과목 수
Neumann	4
Turing	1
Euler	3
Gauss	1
Archimedes	5
Knuth	2
Newton	3

이름	과목 수
Turing	**1**
Gauss	**1**
Knuth	2
Euler	**3**
Newton	**3**
Neumann	4
Archimedes	5

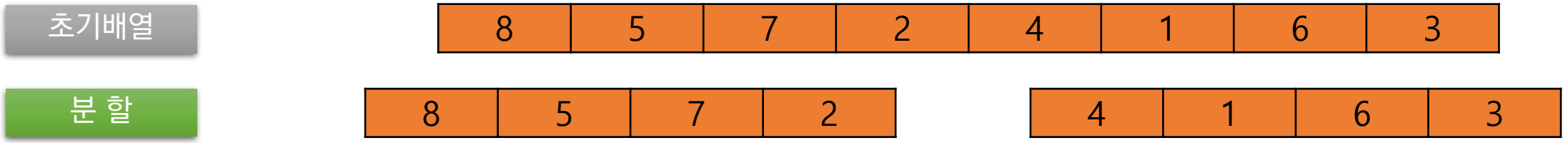
Merge Sort

❖ 정렬과정

초기배열

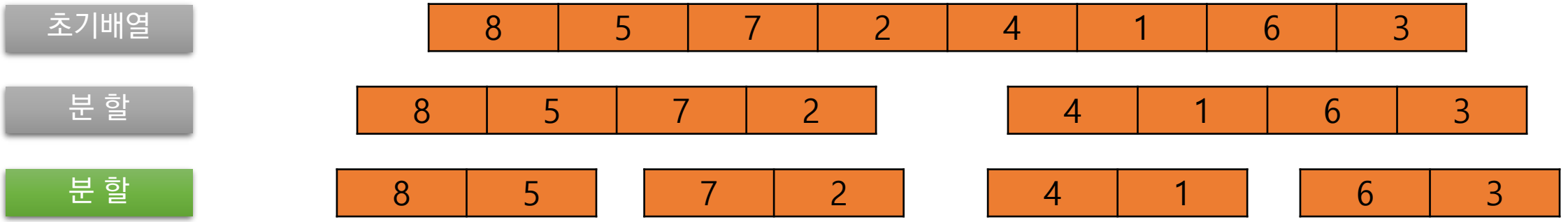
Merge Sort

❖ 정렬과정



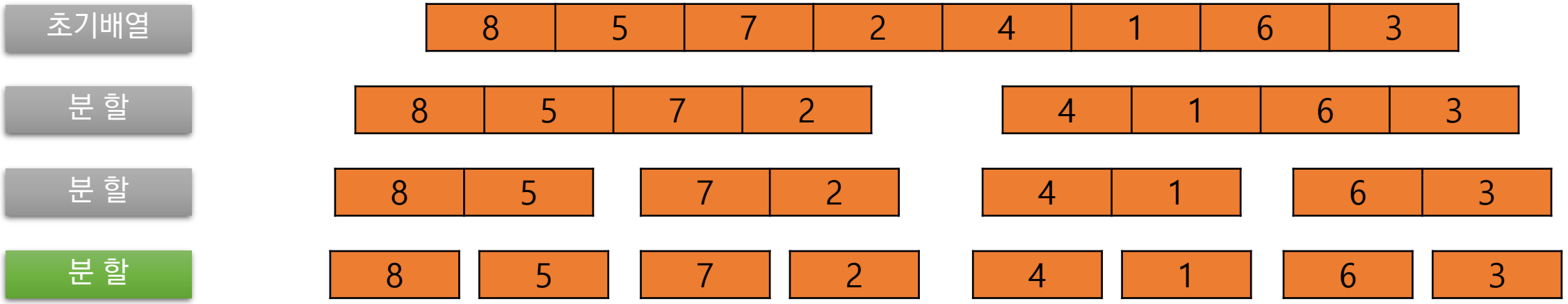
Merge Sort

❖ 정렬과정



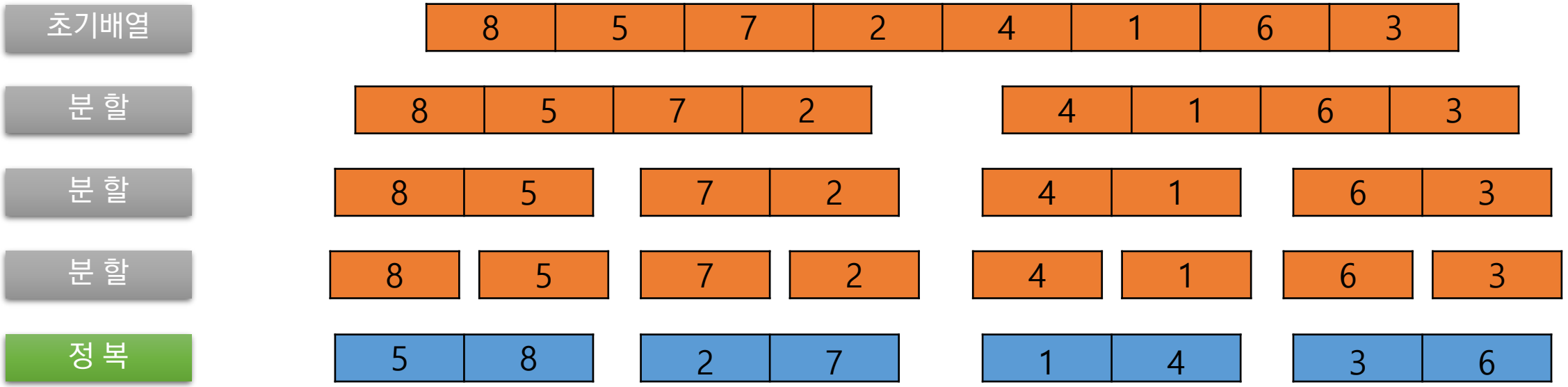
Merge Sort

❖ 정렬과정



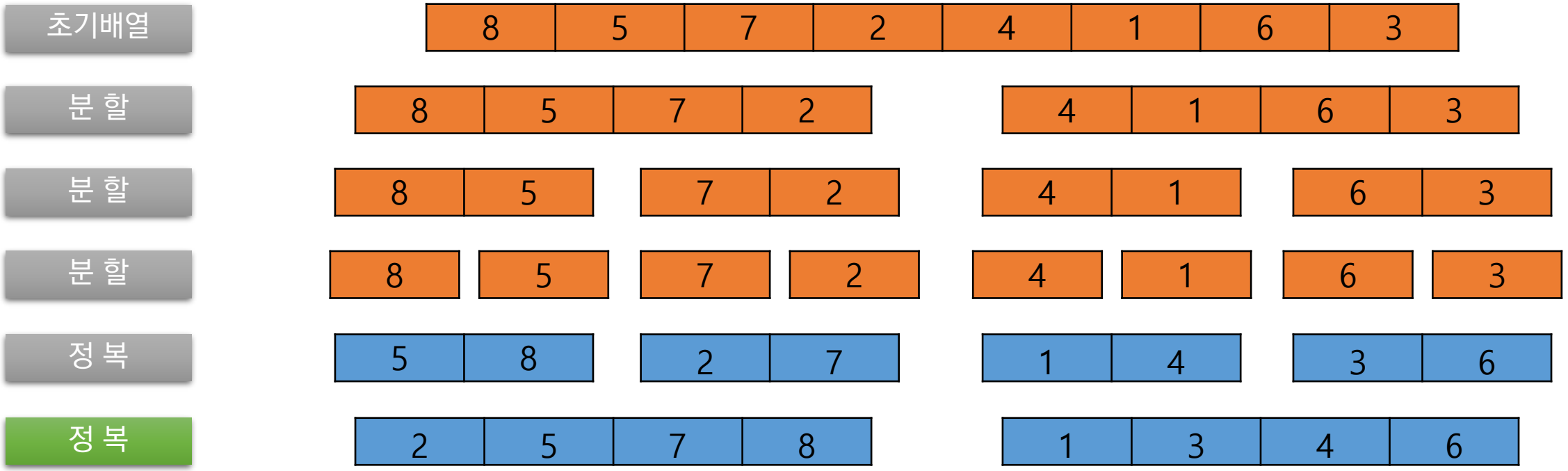
Merge Sort

❖ 정렬과정



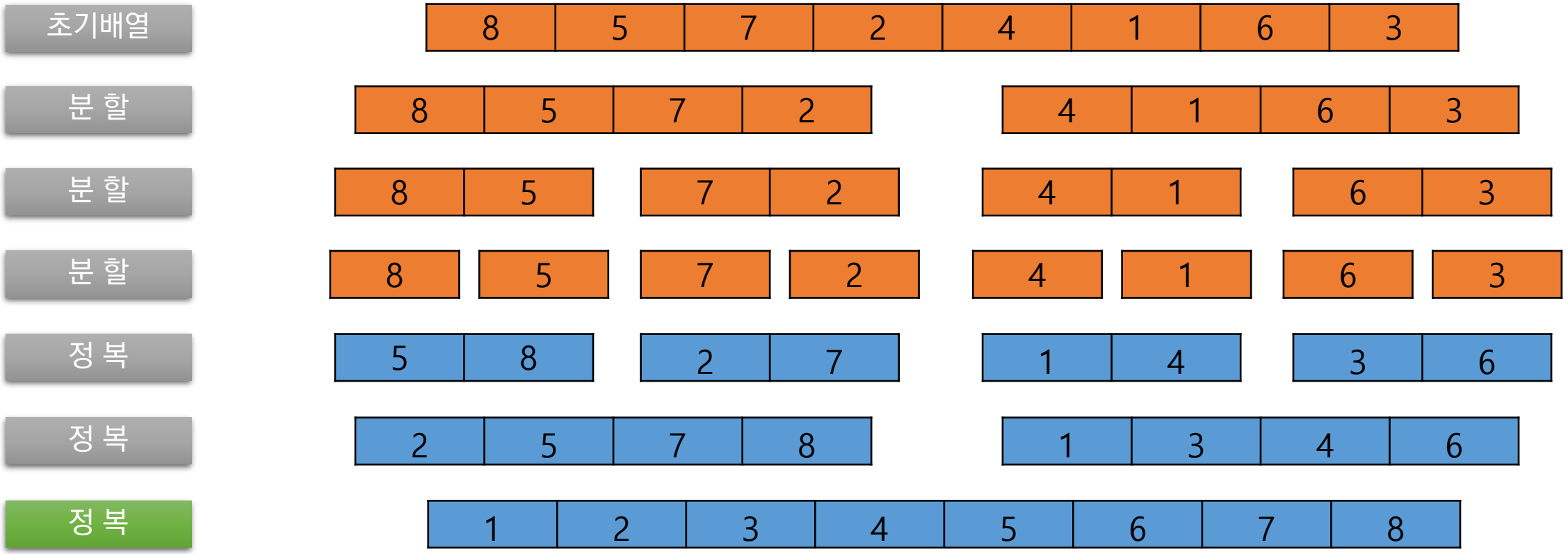
Merge Sort

❖ 정렬과정



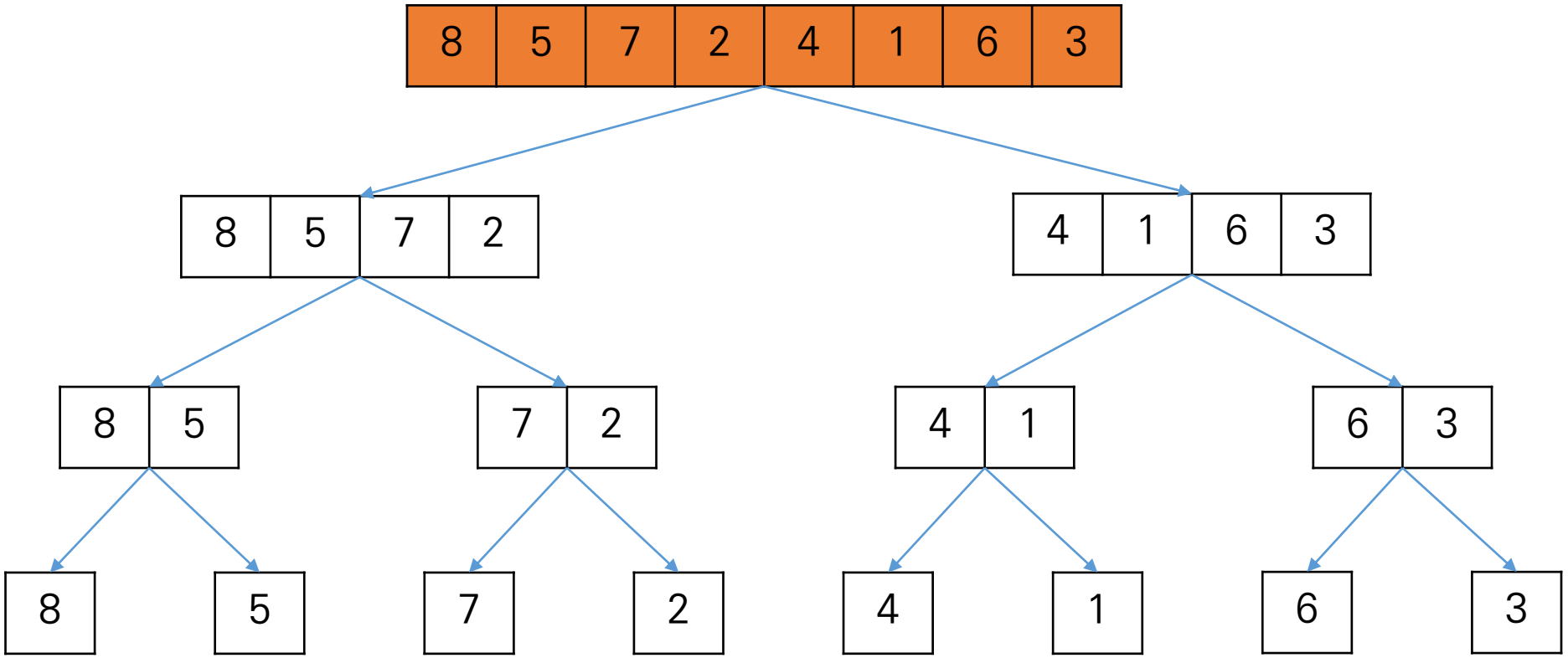
Merge Sort

❖ 정렬과정



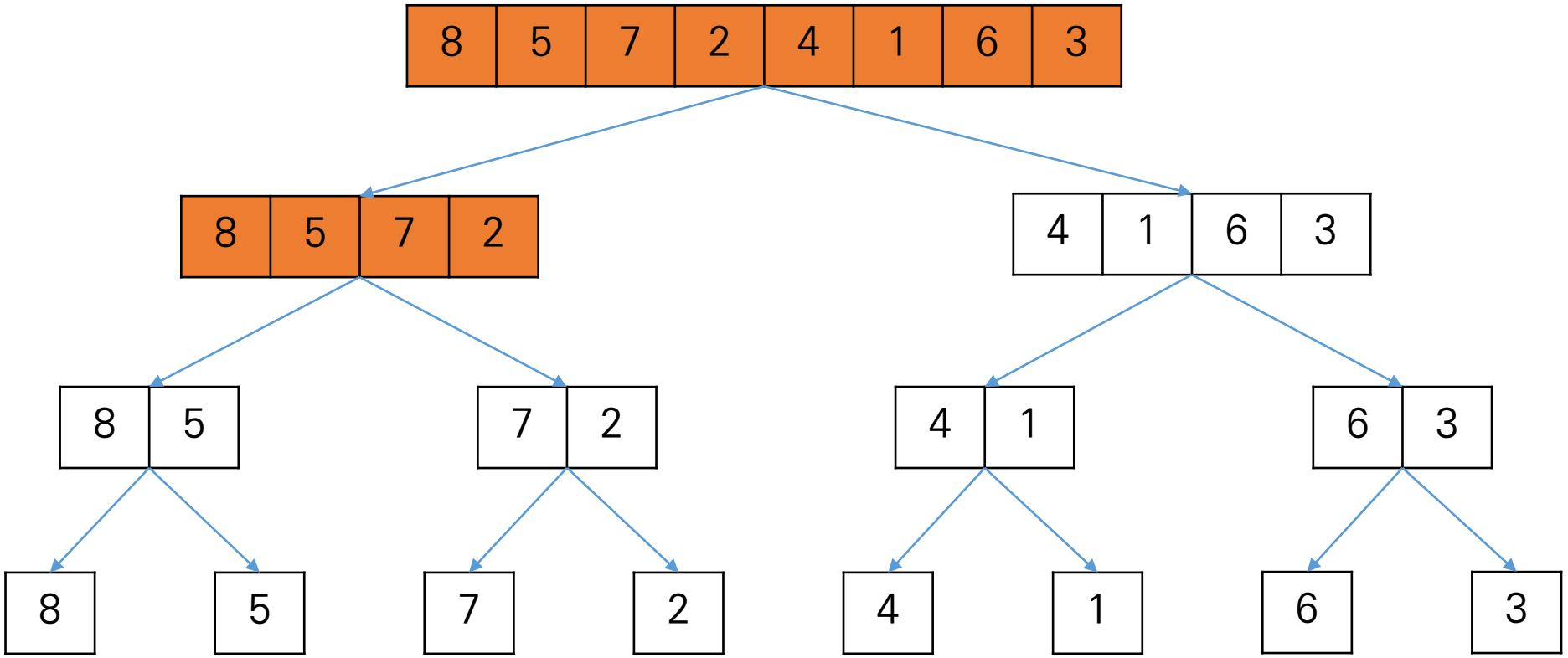
Merge Sort

❖ 정렬 과정을 트리로 표현



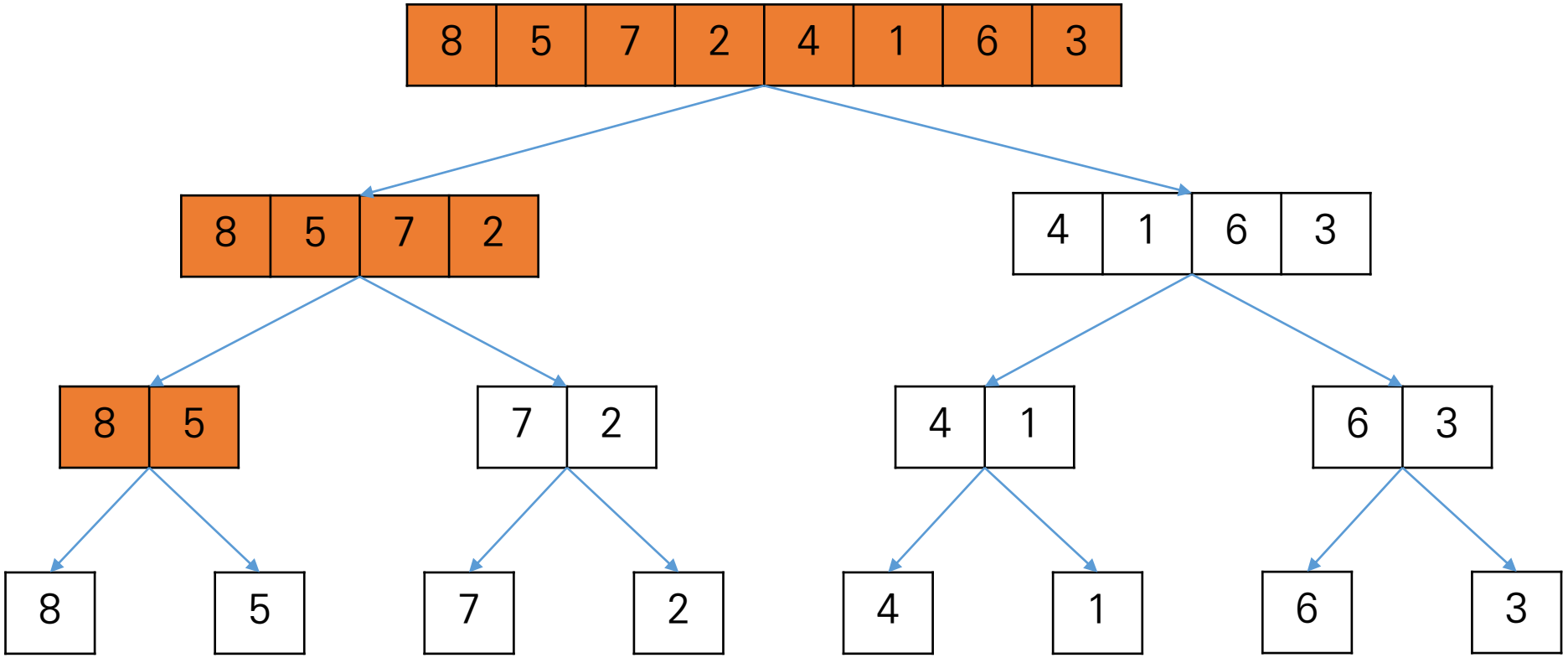
Merge Sort

❖ 정렬 과정을 트리로 표현



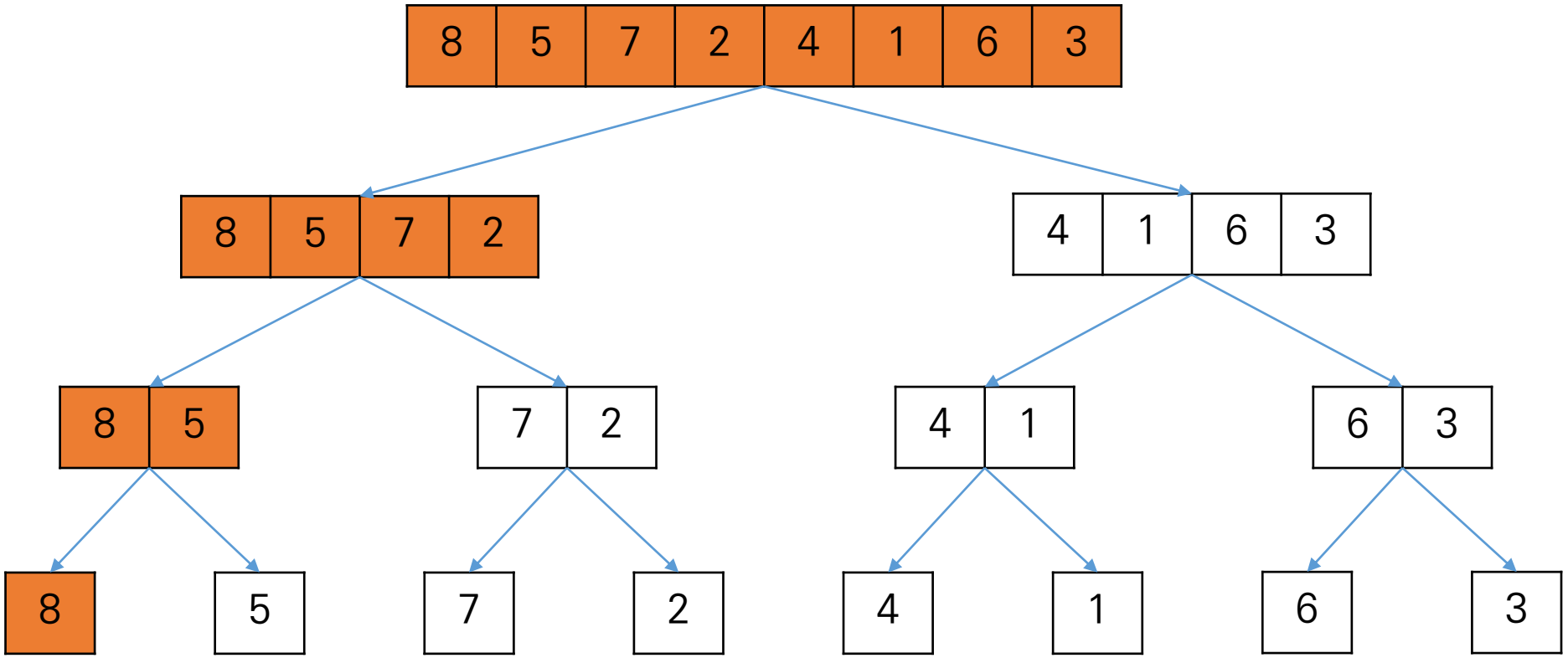
Merge Sort

❖ 정렬 과정을 트리로 표현



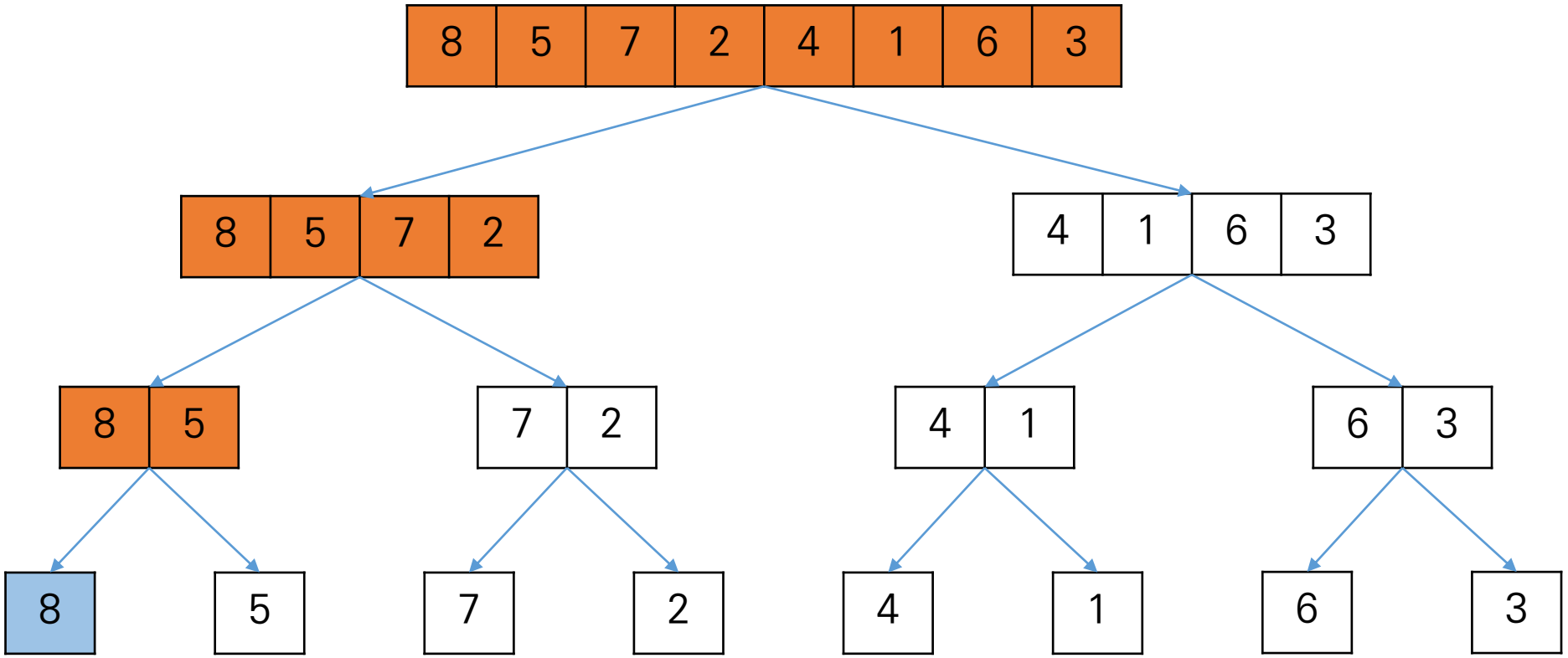
Merge Sort

❖ 정렬 과정을 트리로 표현



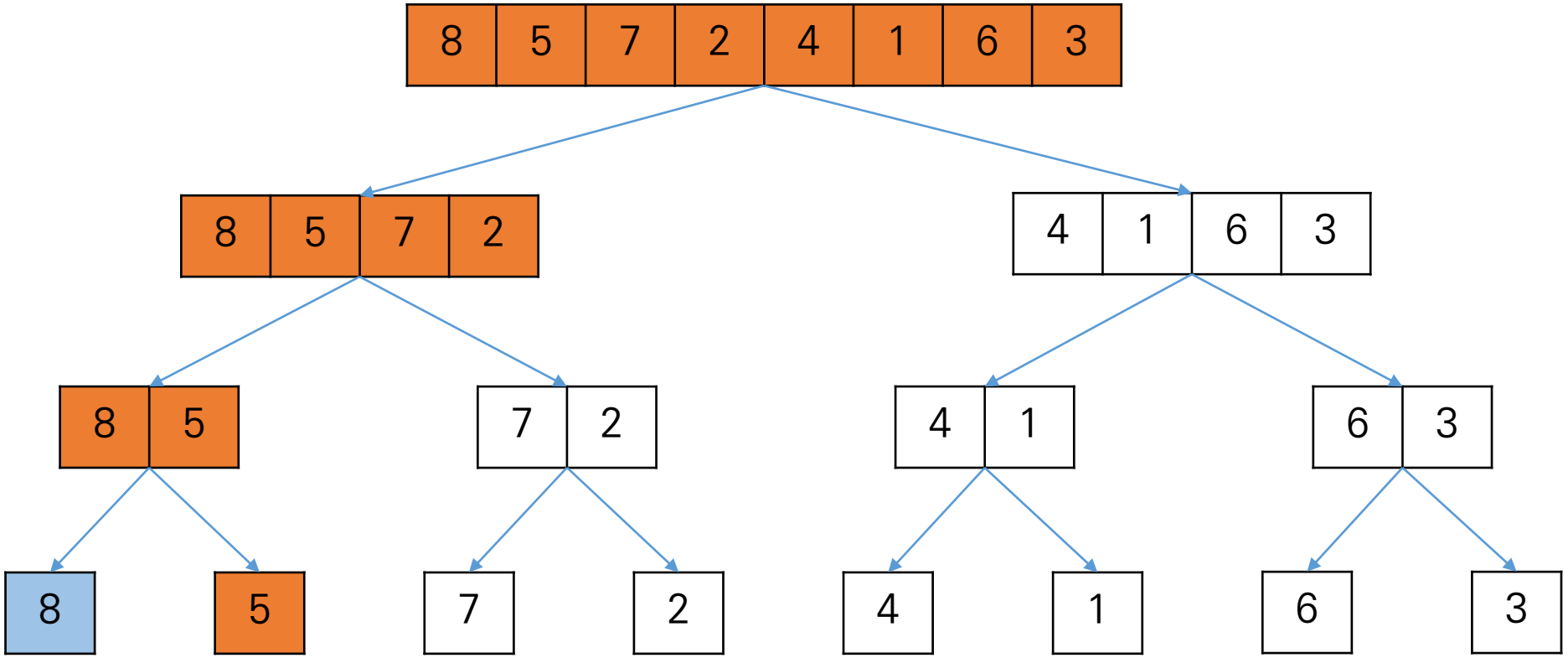
Merge Sort

❖ 정렬 과정을 트리로 표현



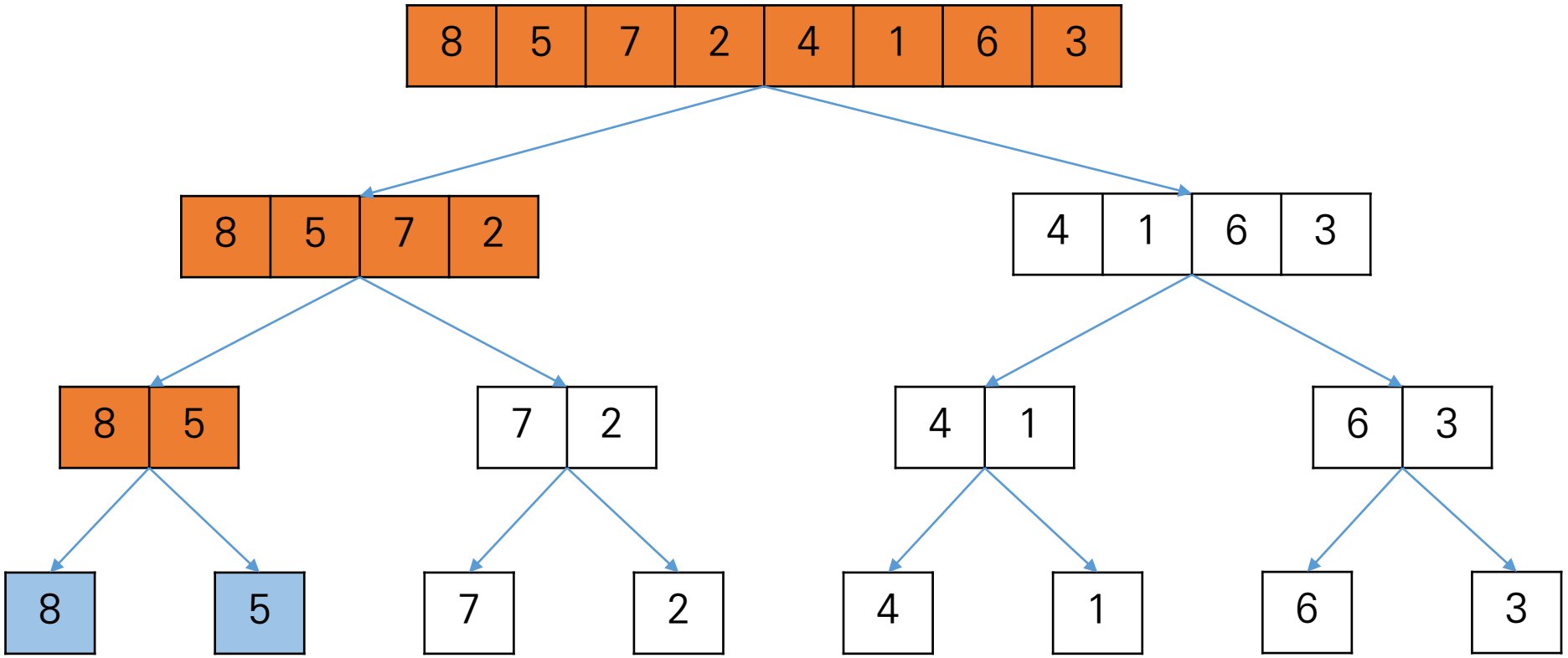
Merge Sort

❖ 정렬 과정을 트리로 표현



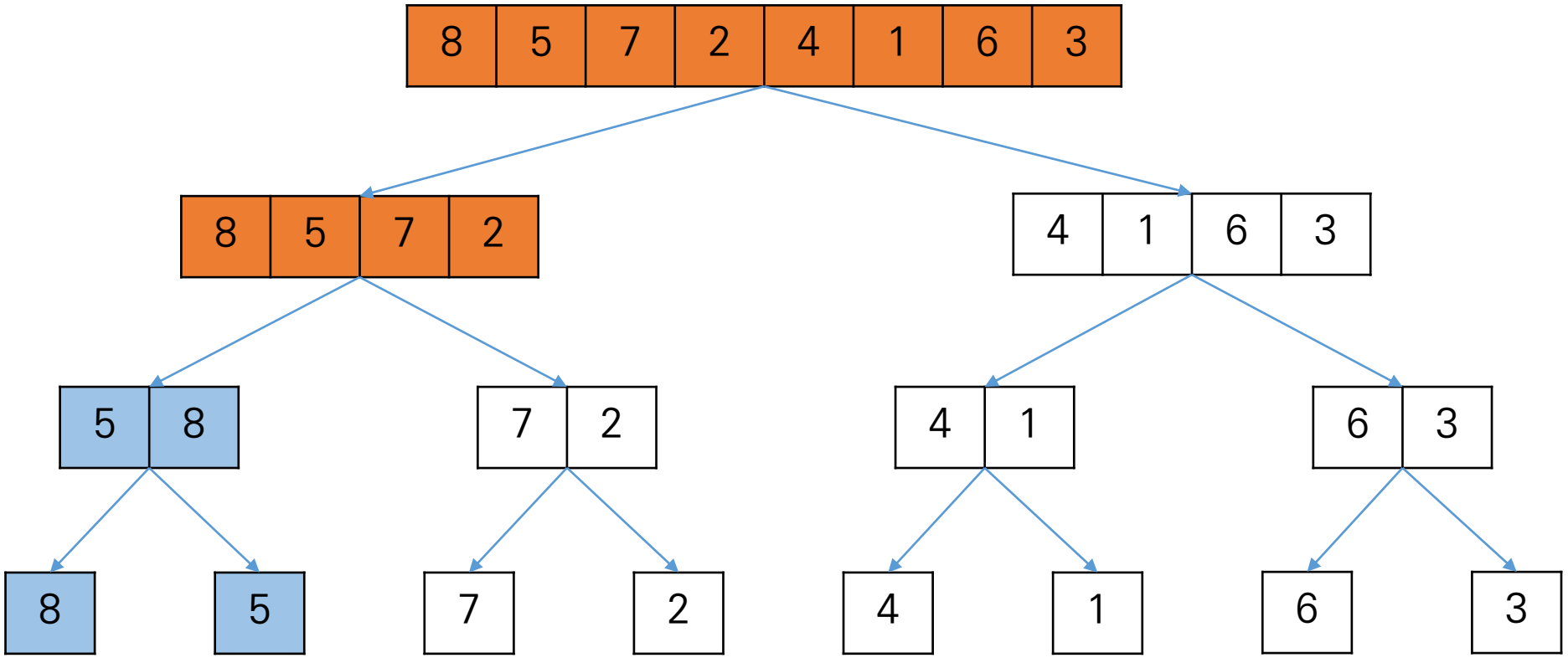
Merge Sort

❖ 정렬 과정을 트리로 표현



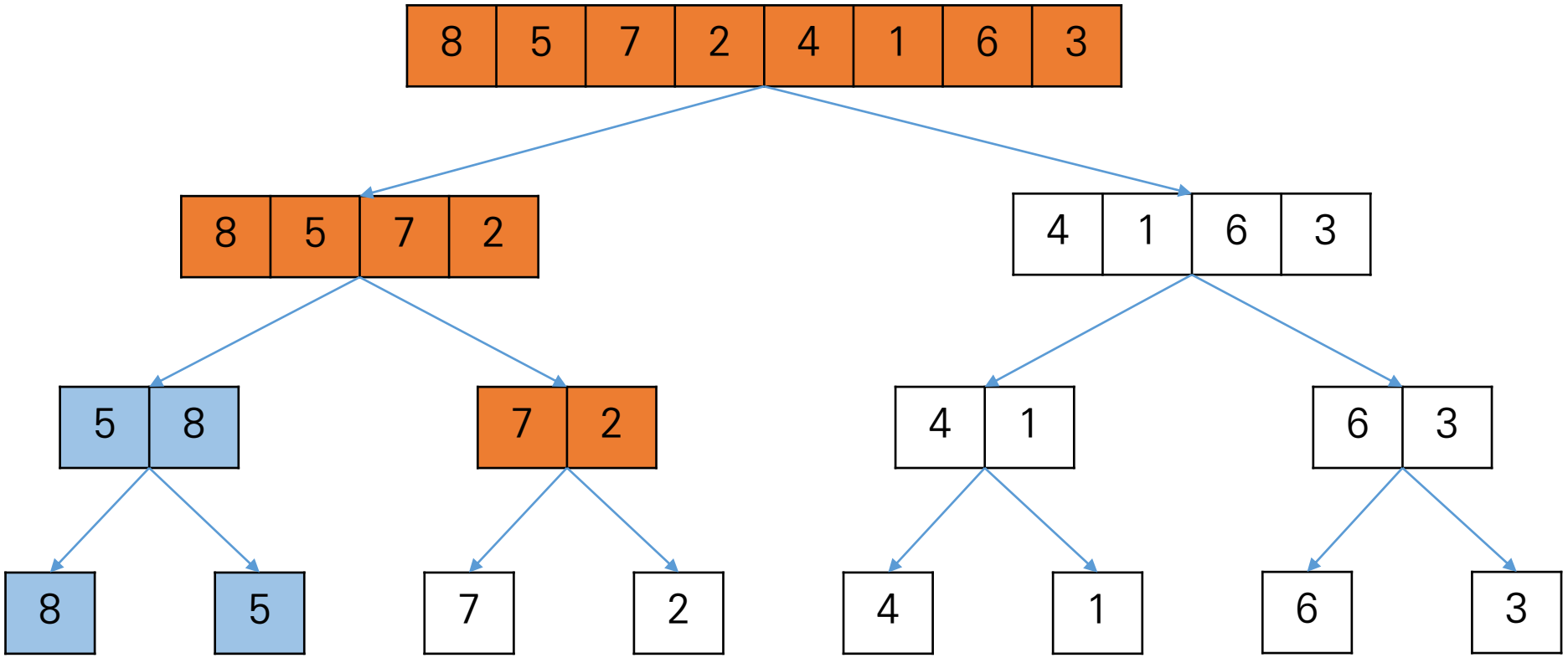
Merge Sort

❖ 정렬 과정을 트리로 표현



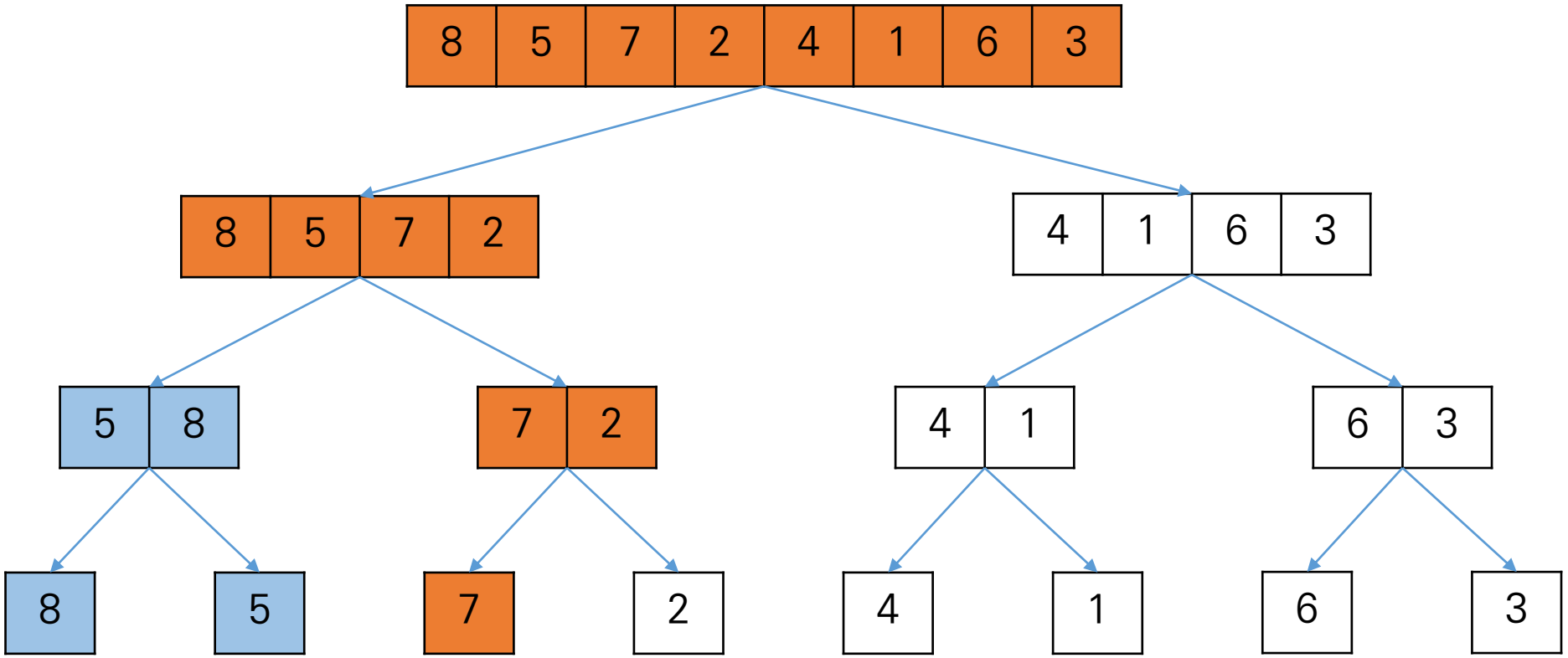
Merge Sort

❖ 정렬 과정을 트리로 표현



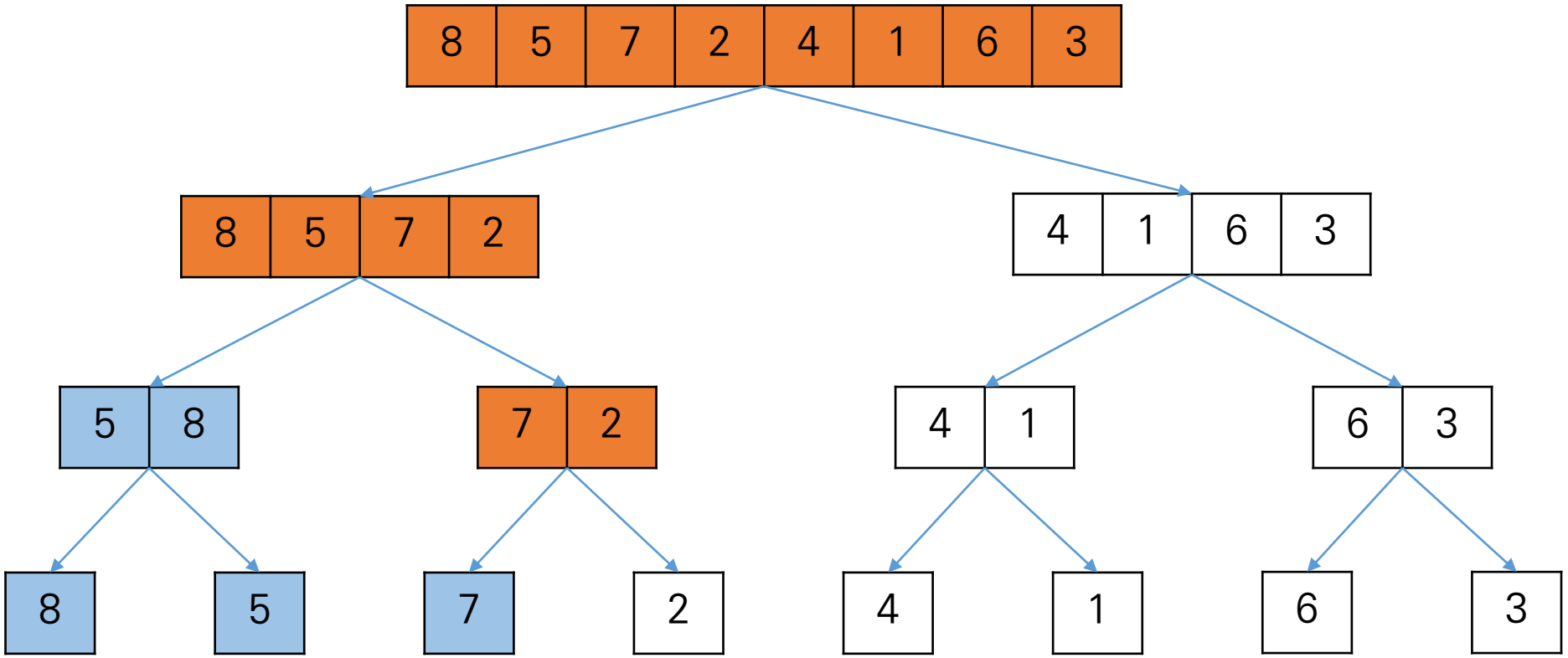
Merge Sort

❖ 정렬 과정을 트리로 표현



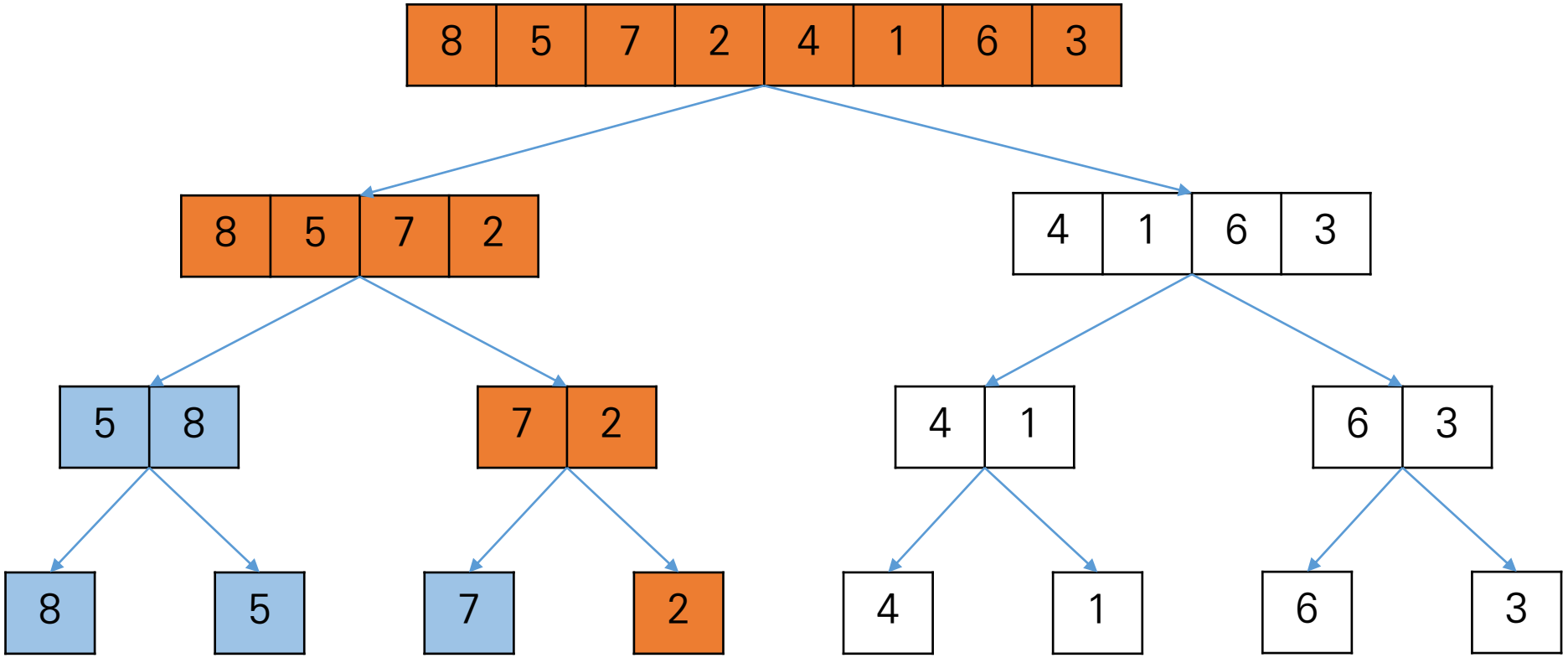
Merge Sort

❖ 정렬 과정을 트리로 표현



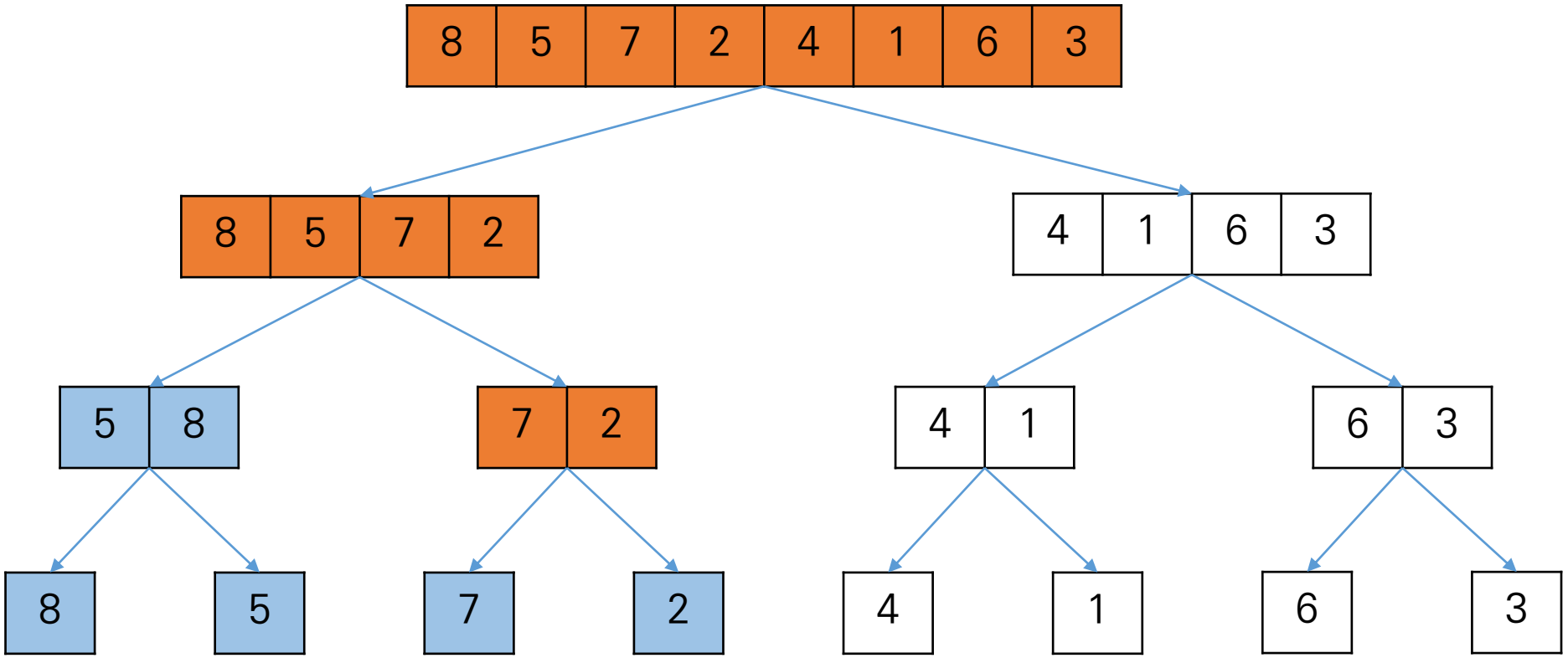
Merge Sort

❖ 정렬 과정을 트리로 표현



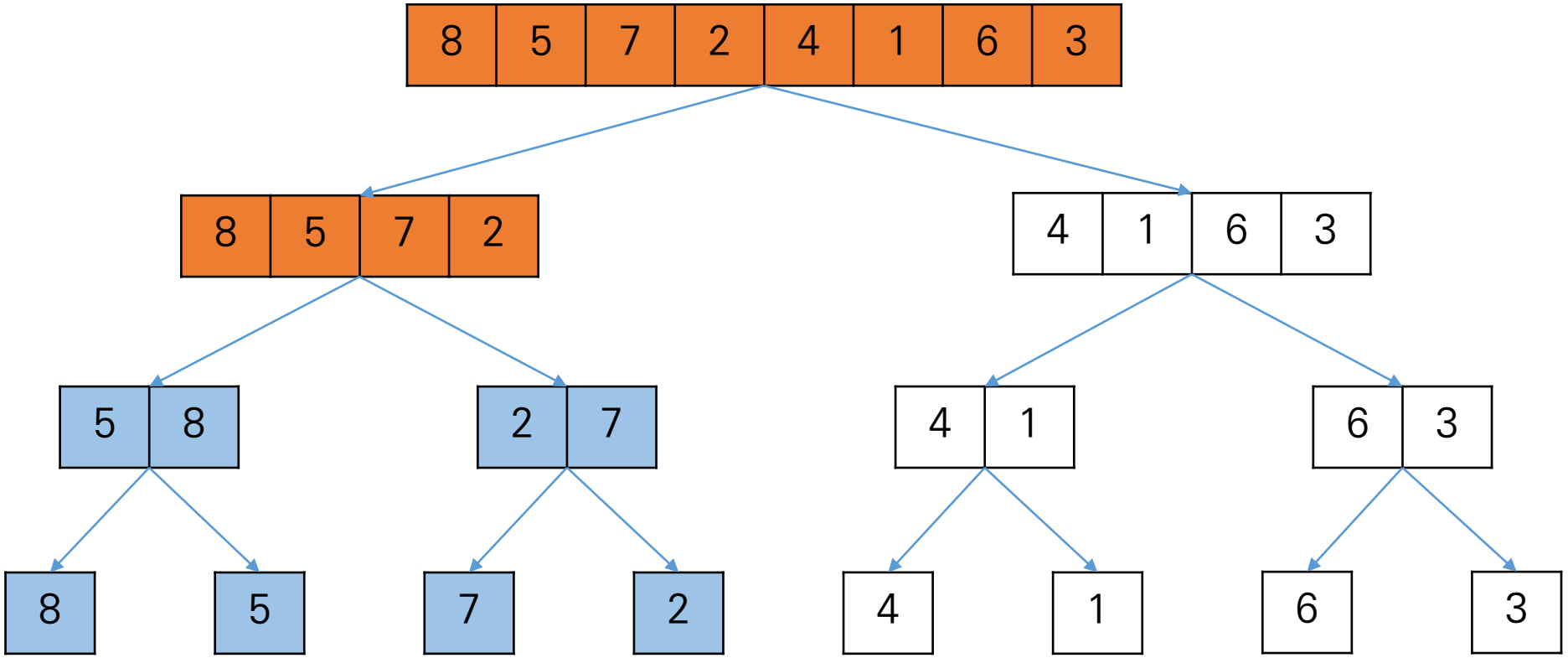
Merge Sort

❖ 정렬 과정을 트리로 표현



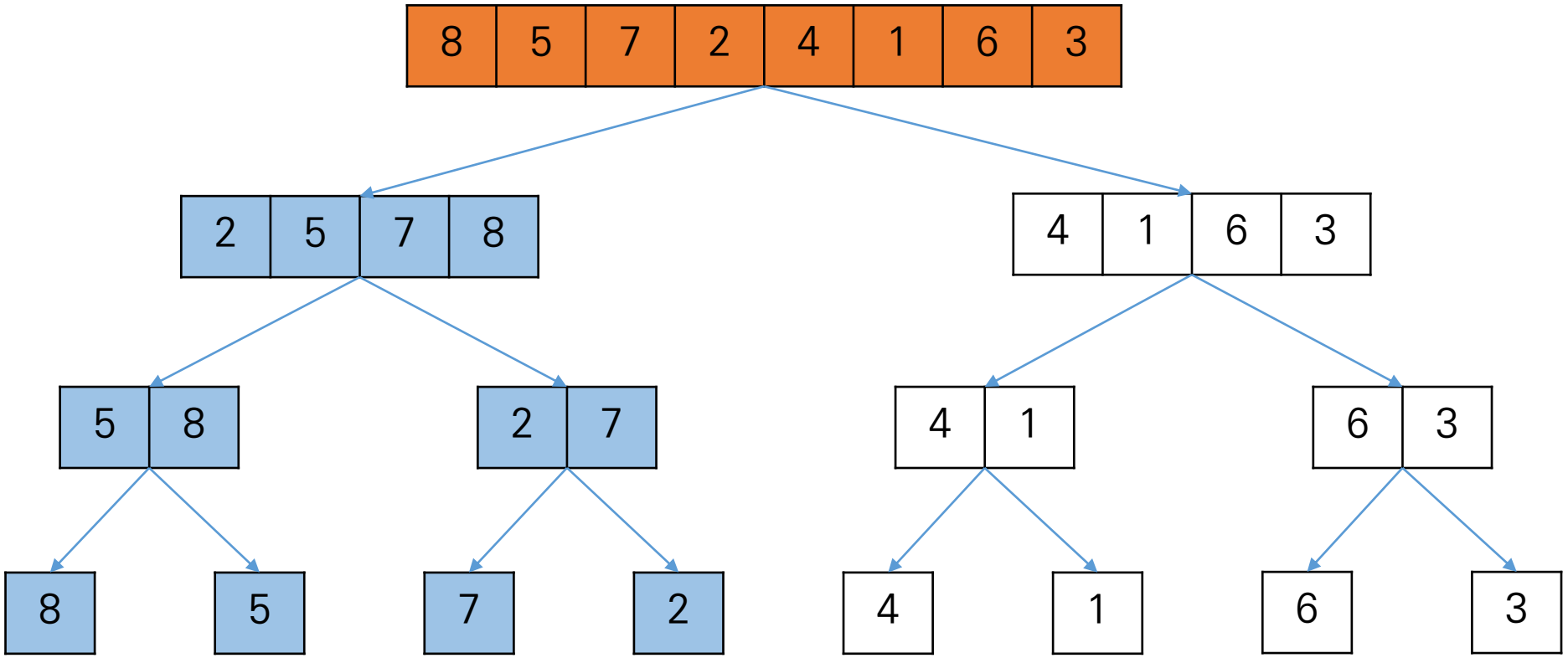
Merge Sort

❖ 정렬 과정을 트리로 표현



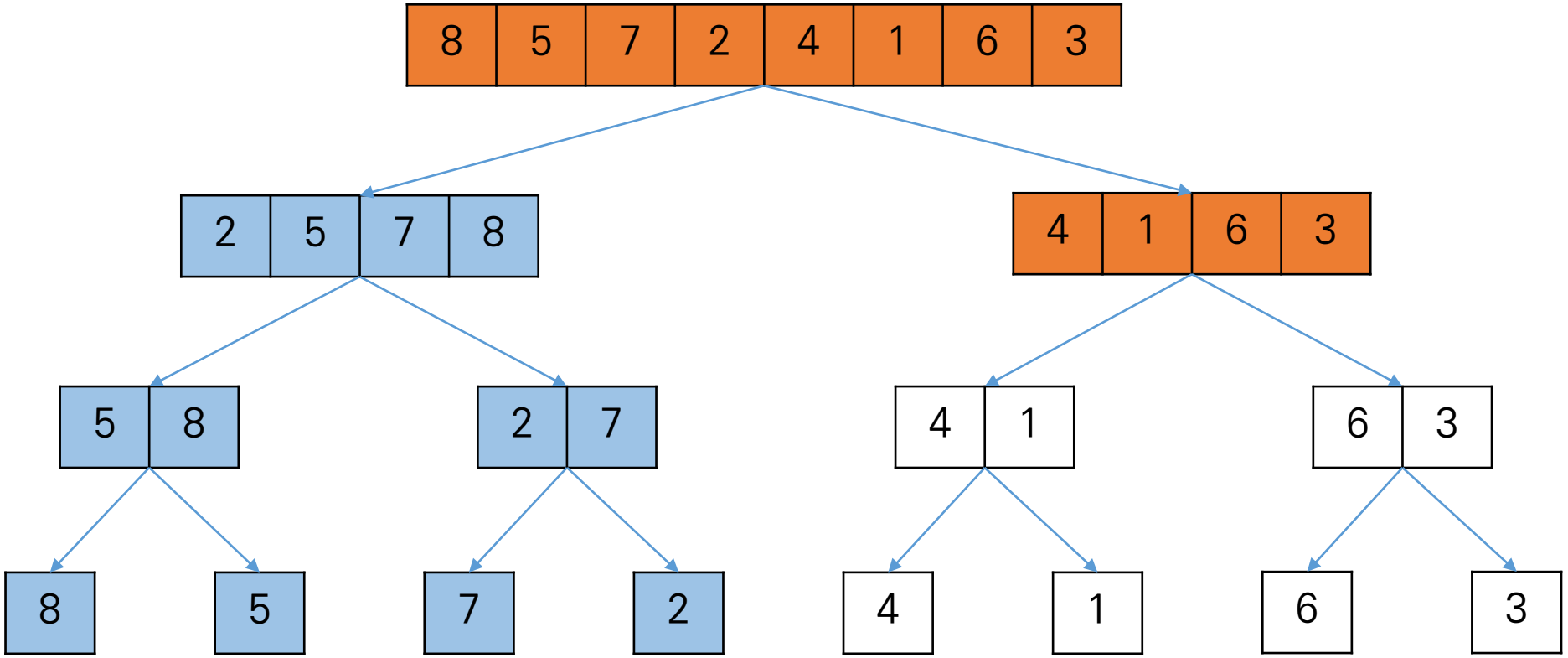
Merge Sort

❖ 정렬 과정을 트리로 표현



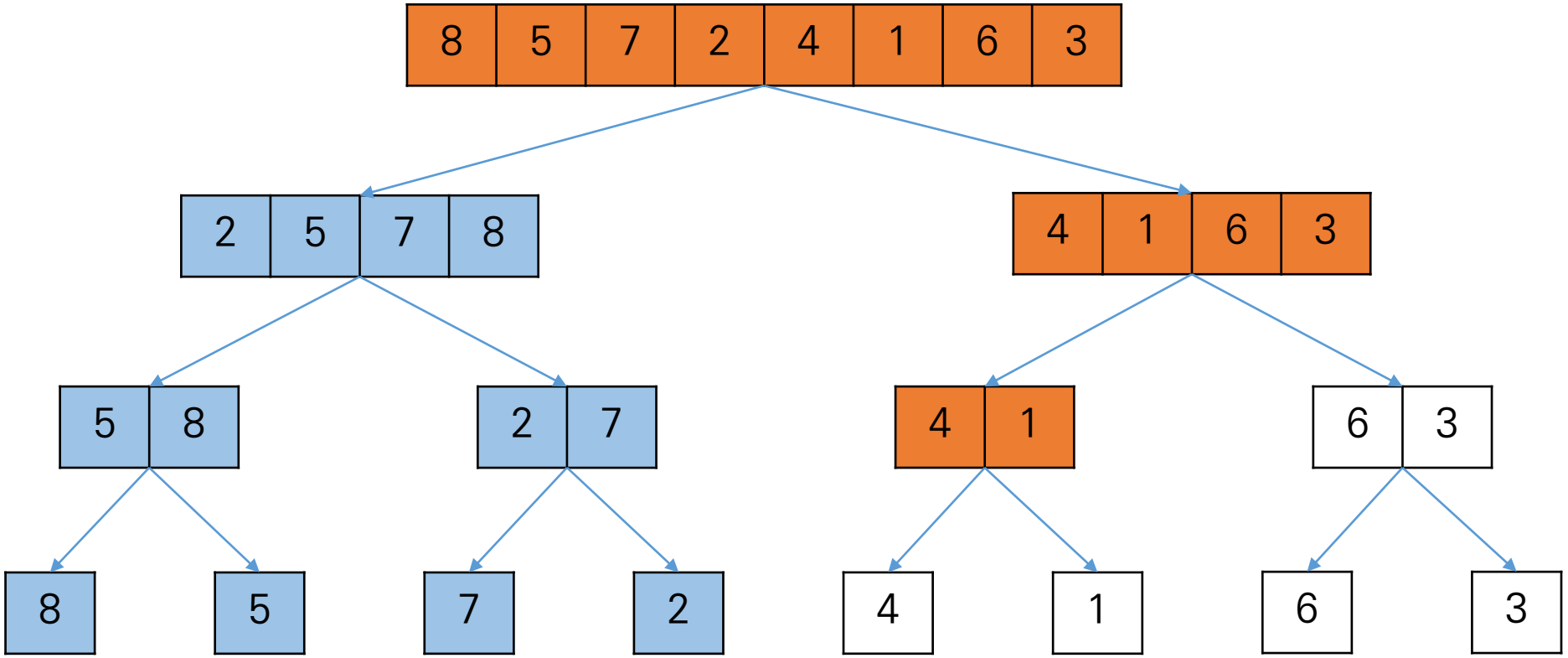
Merge Sort

❖ 정렬 과정을 트리로 표현



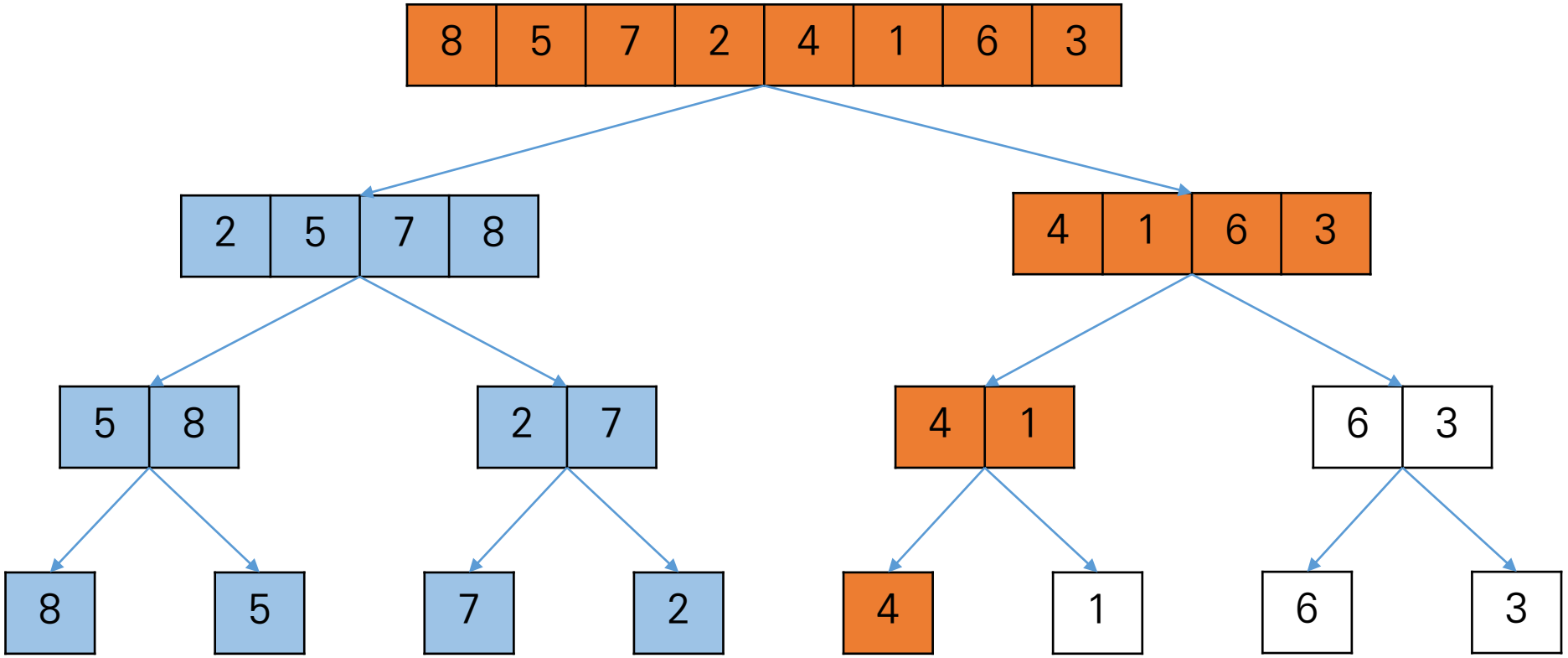
Merge Sort

❖ 정렬 과정을 트리로 표현



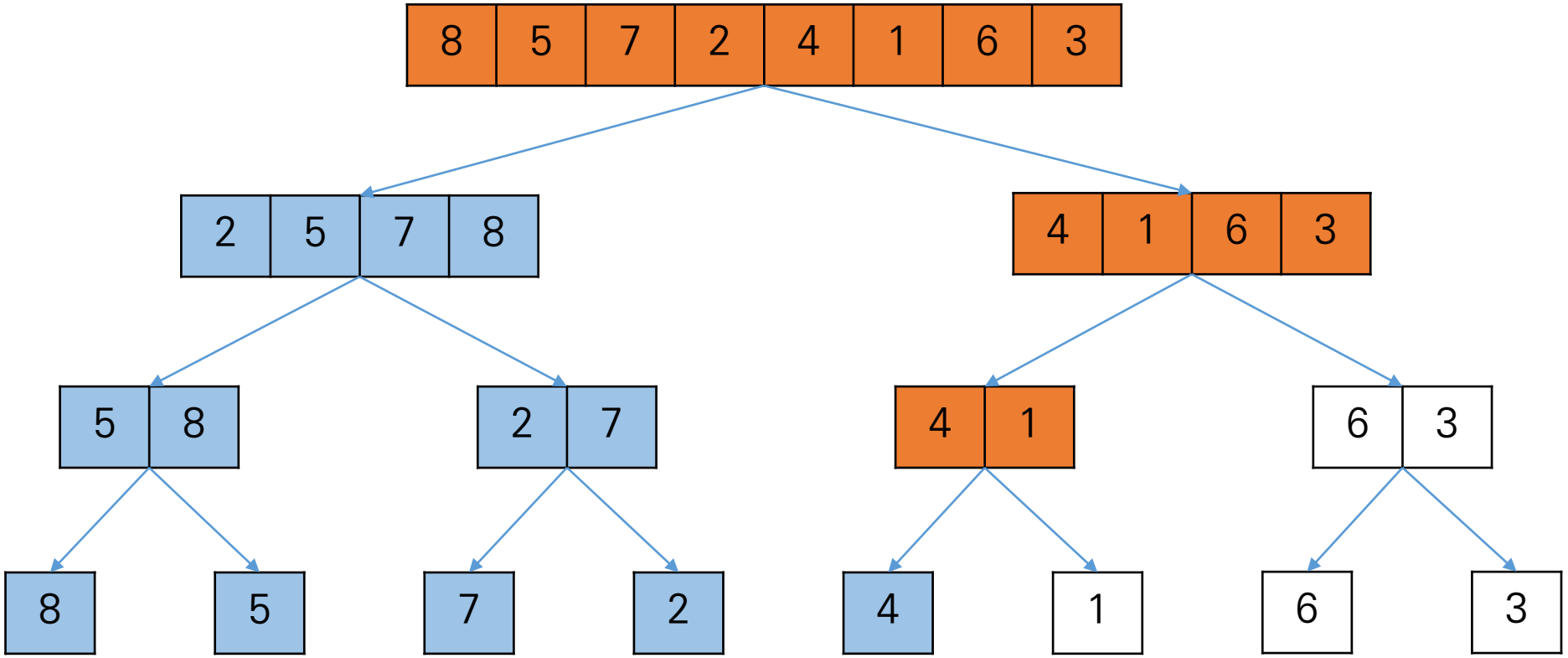
Merge Sort

❖ 정렬 과정을 트리로 표현



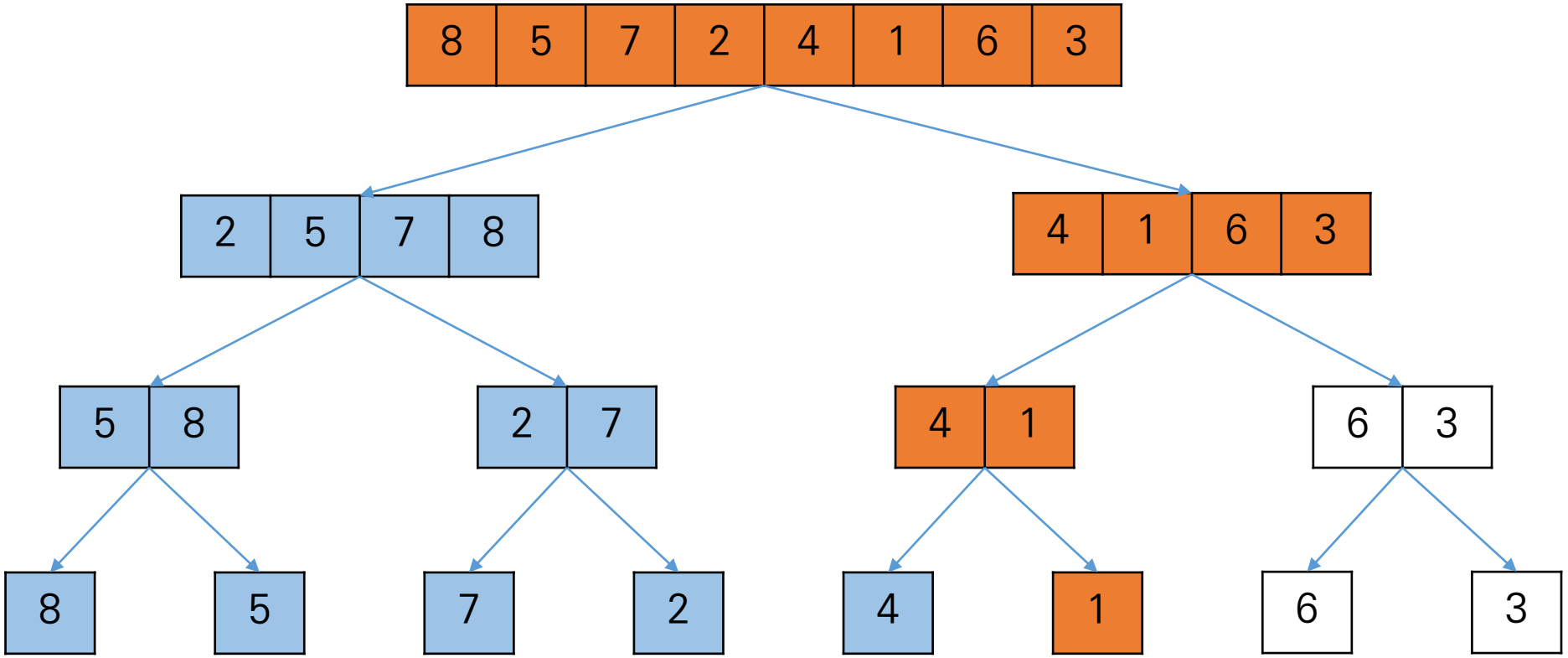
Merge Sort

❖ 정렬 과정을 트리로 표현



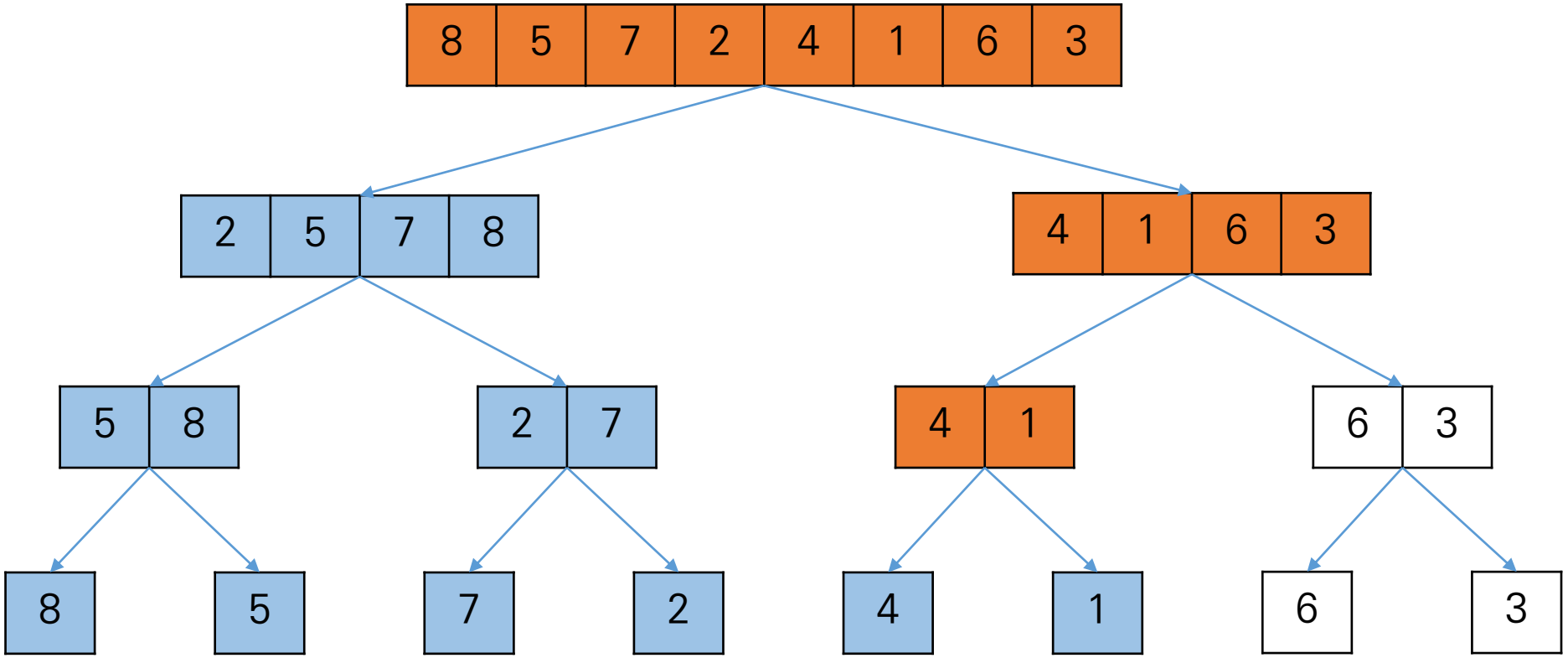
Merge Sort

❖ 정렬 과정을 트리로 표현



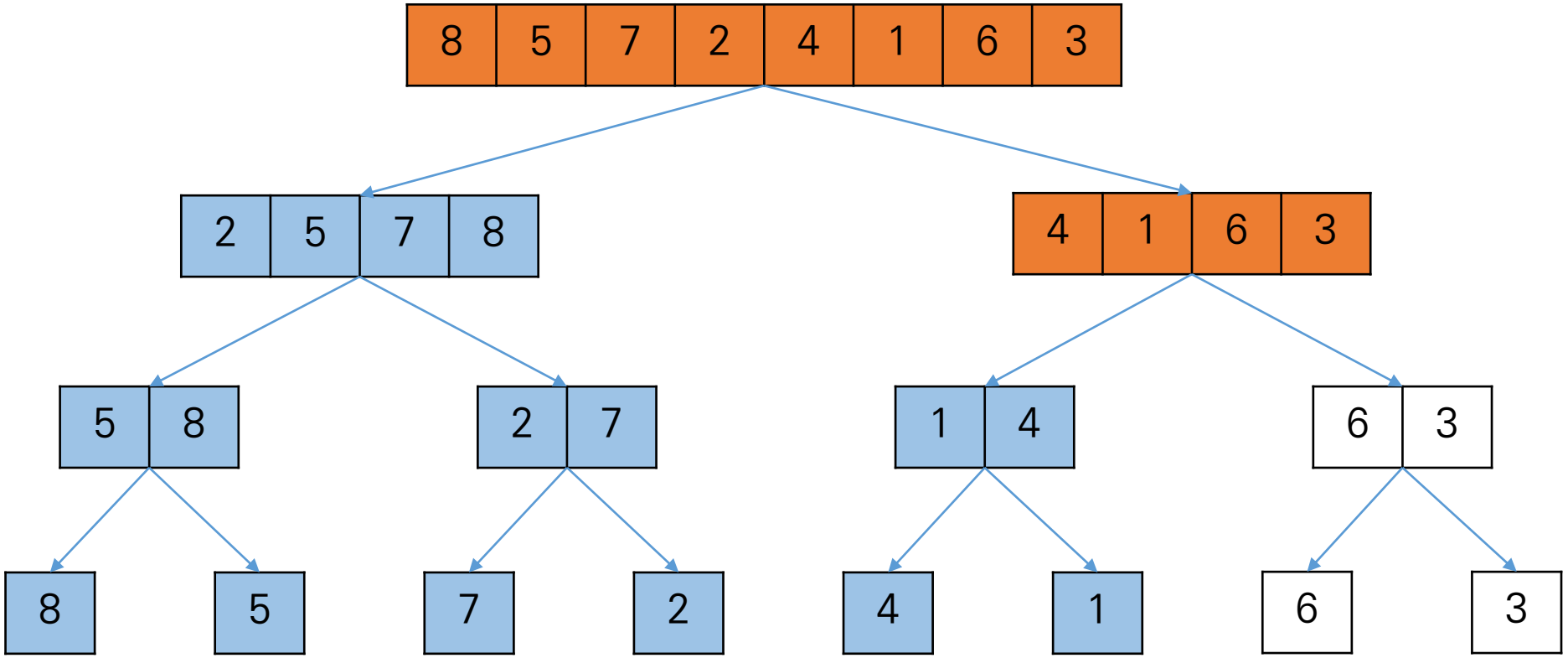
Merge Sort

❖ 정렬 과정을 트리로 표현



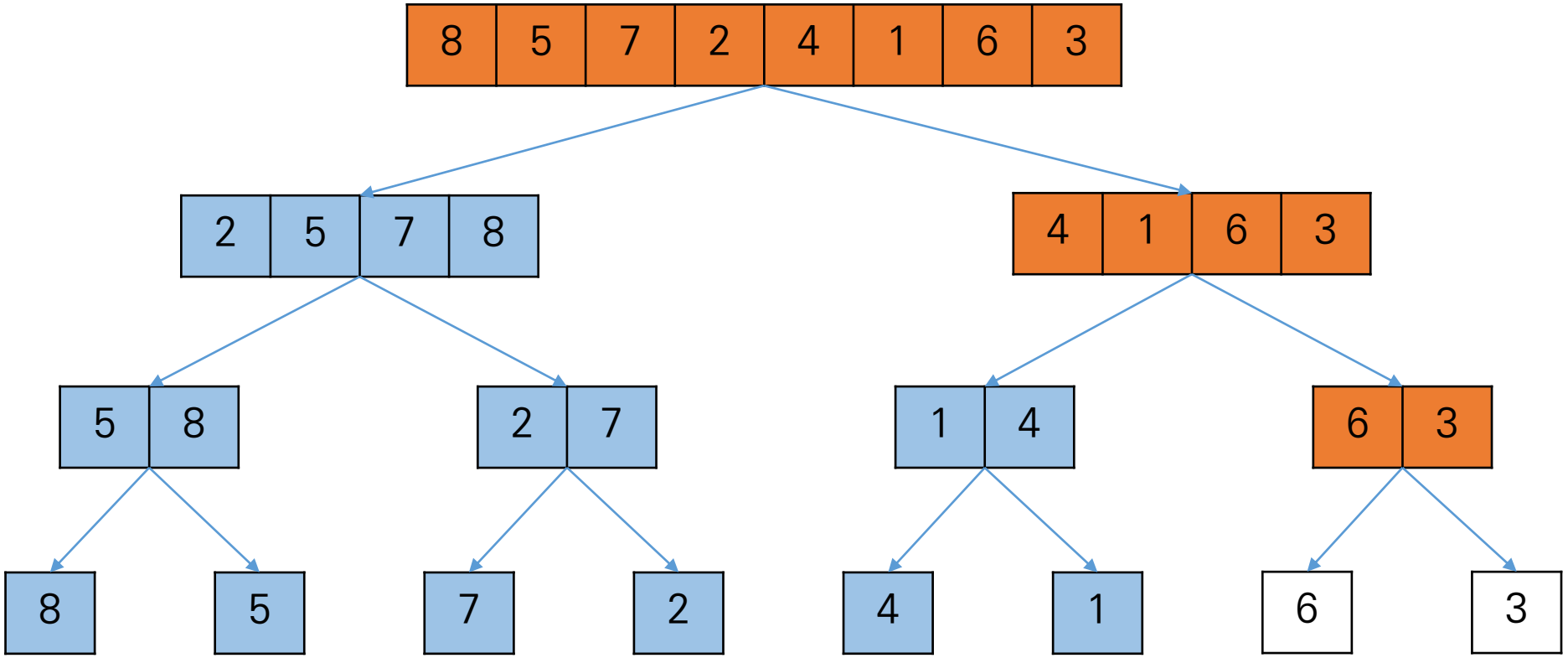
Merge Sort

❖ 정렬 과정을 트리로 표현



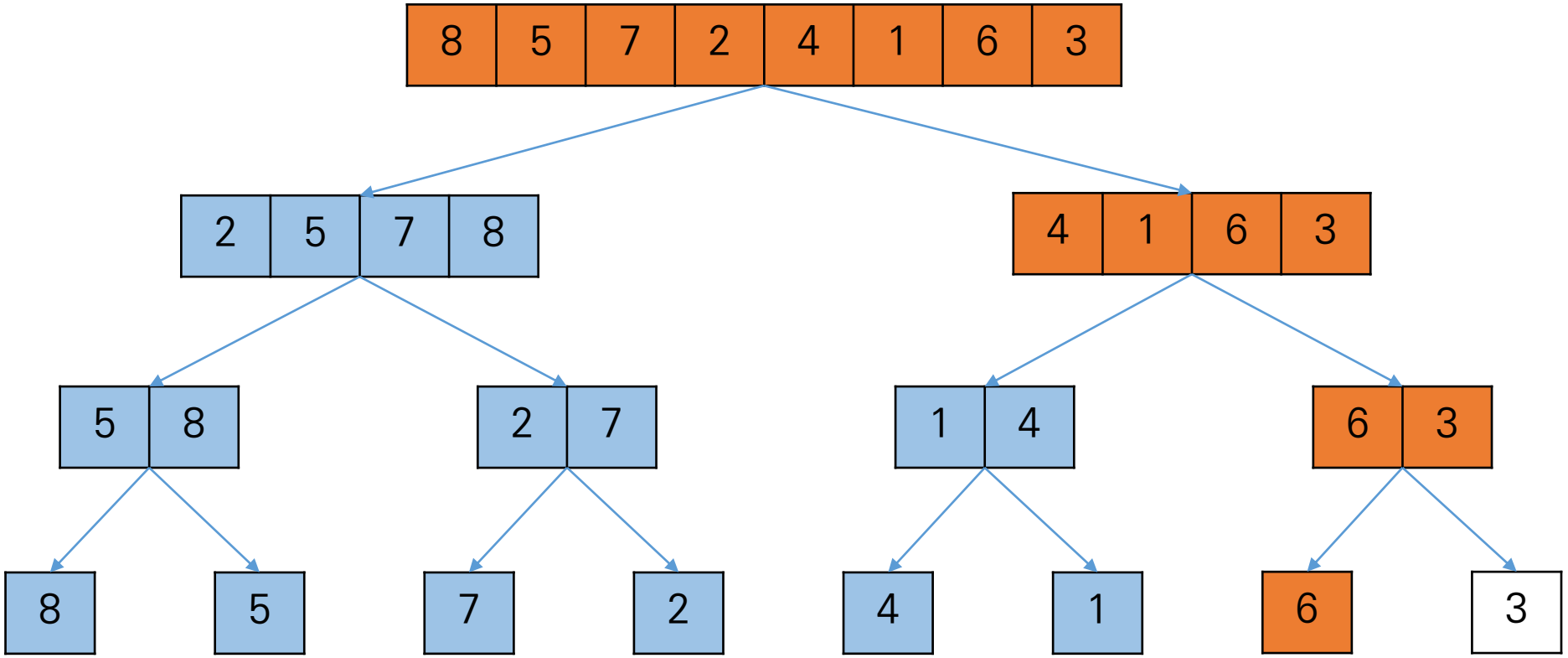
Merge Sort

❖ 정렬 과정을 트리로 표현



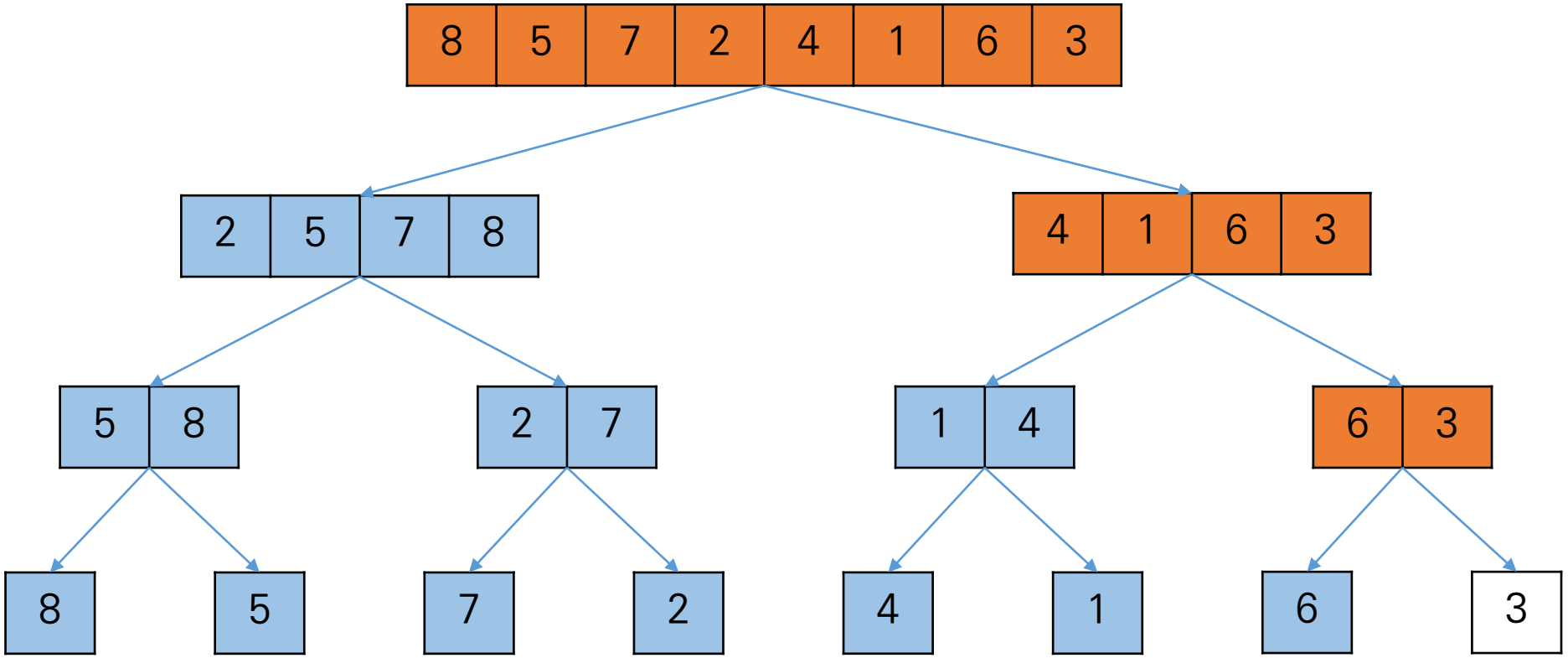
Merge Sort

❖ 정렬 과정을 트리로 표현



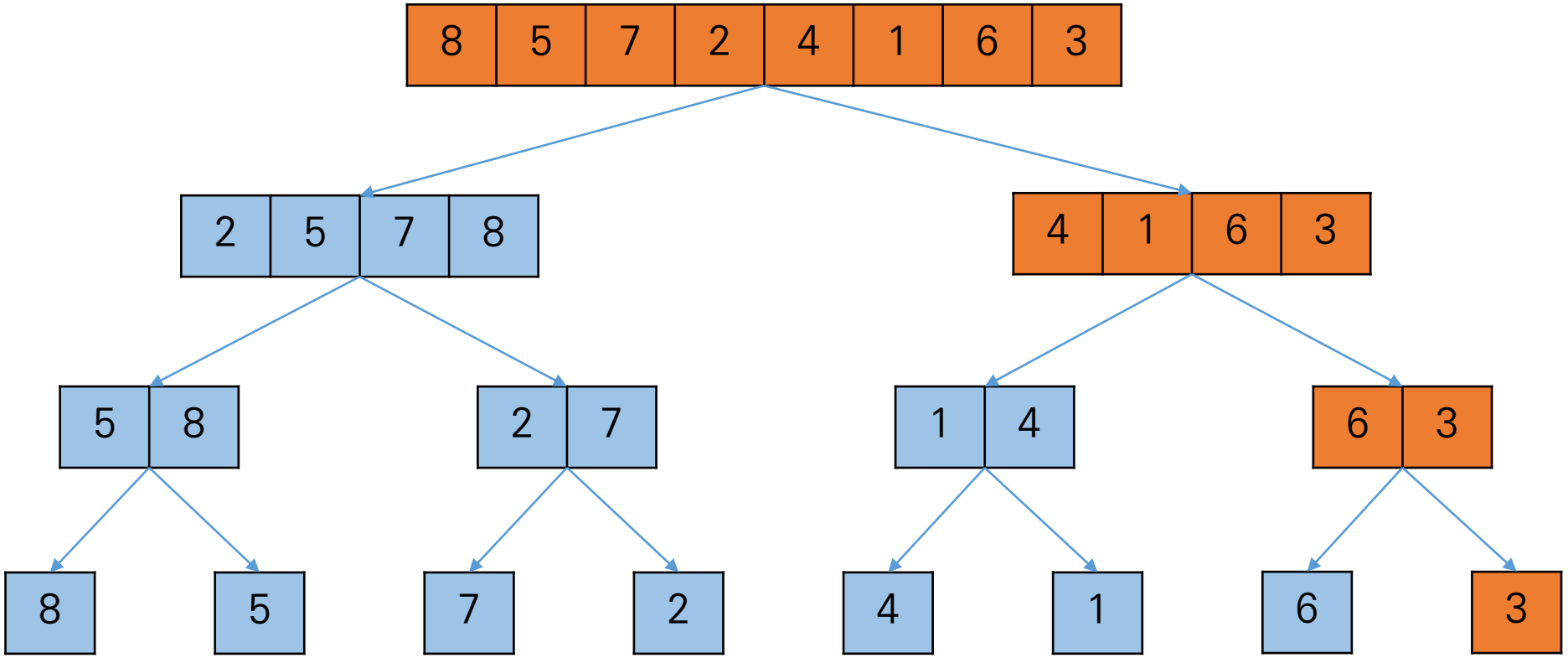
Merge Sort

❖ 정렬 과정을 트리로 표현



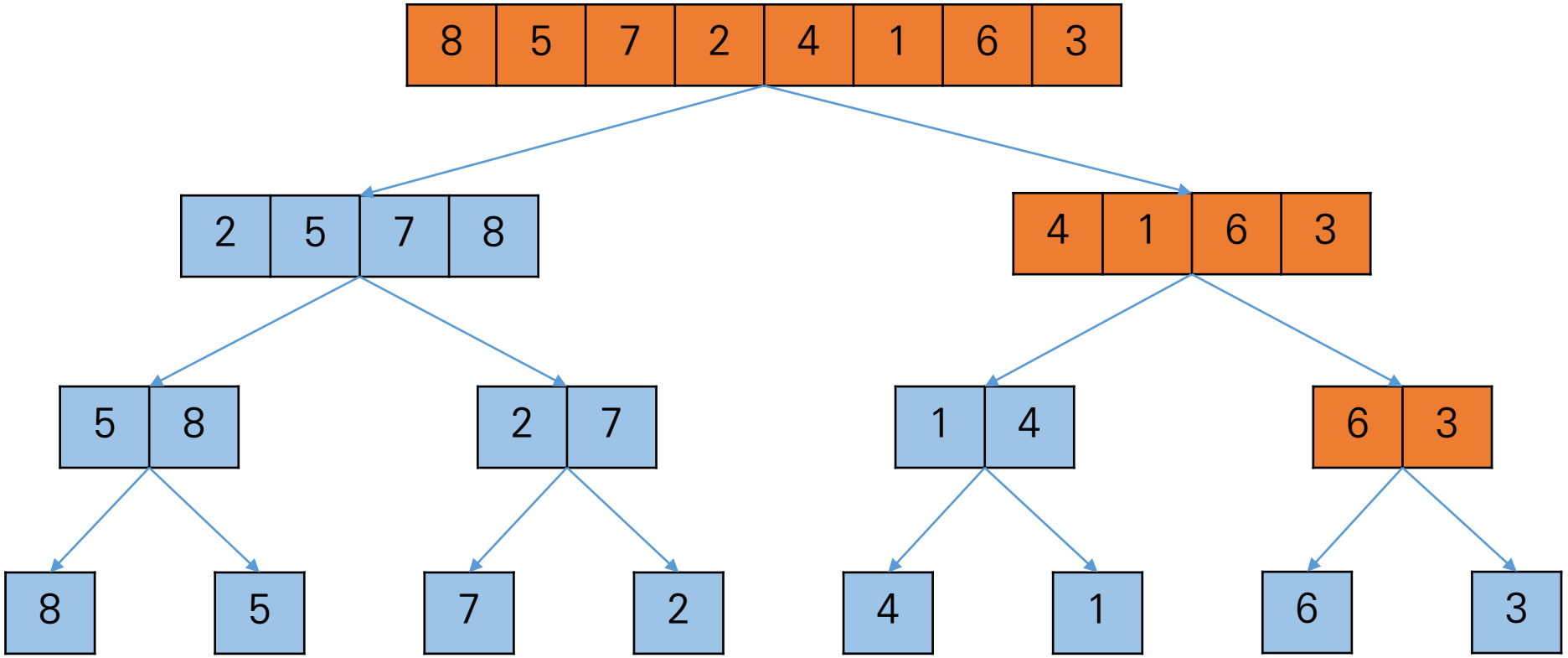
Merge Sort

❖ 정렬 과정을 트리로 표현



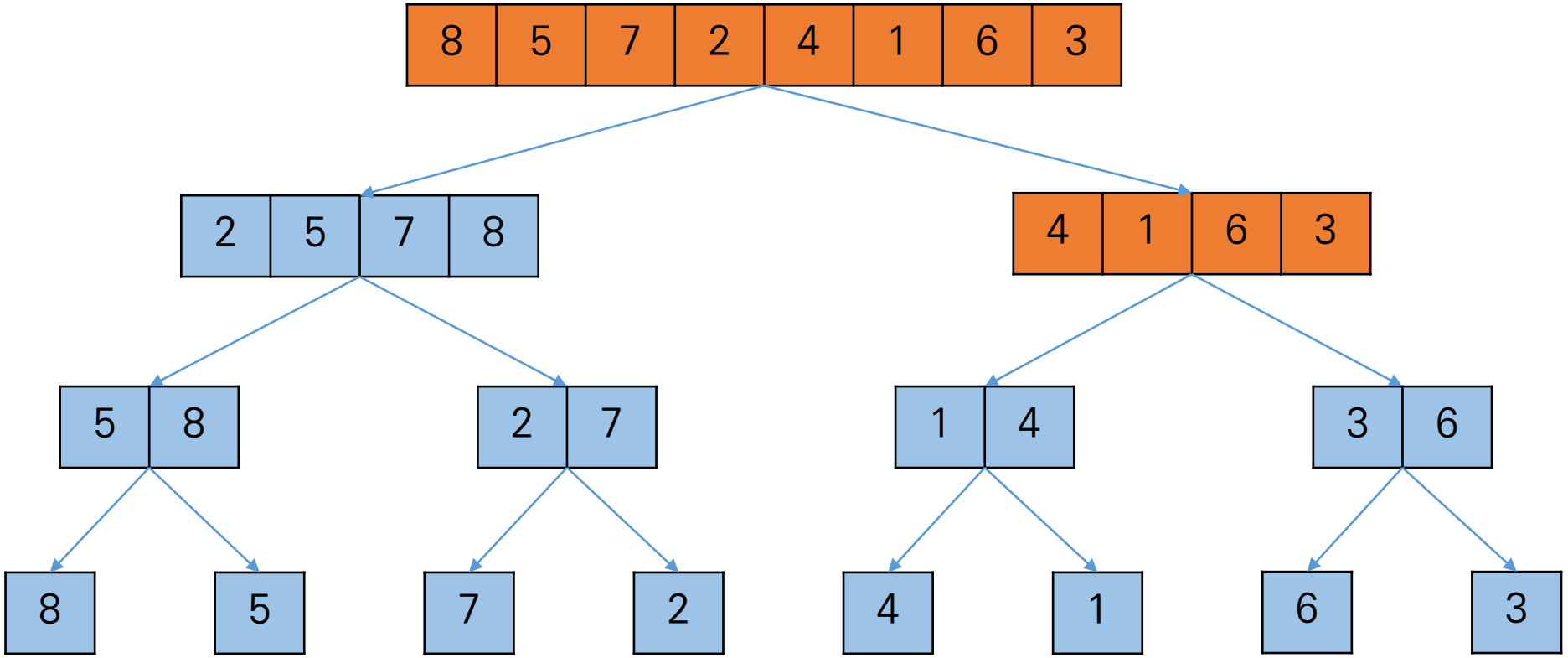
Merge Sort

❖ 정렬 과정을 트리로 표현



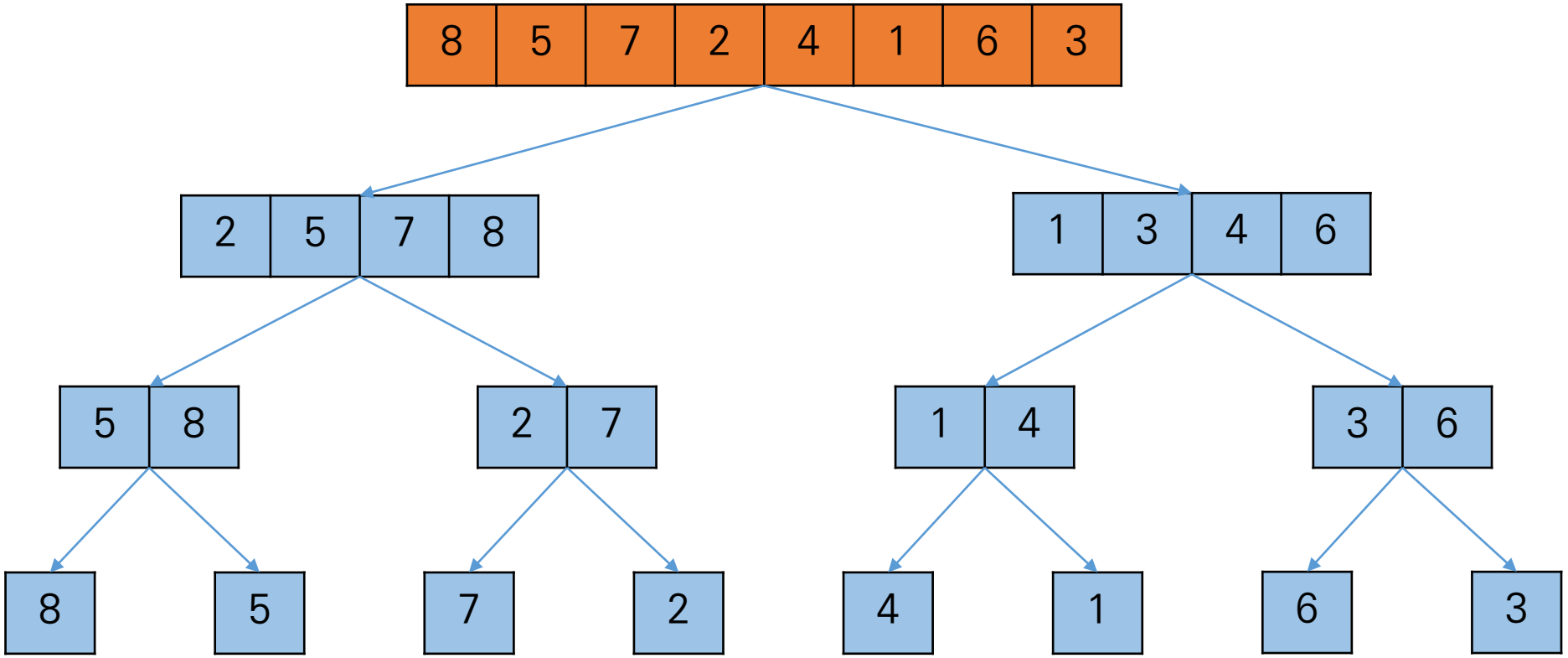
Merge Sort

❖ 정렬 과정을 트리로 표현



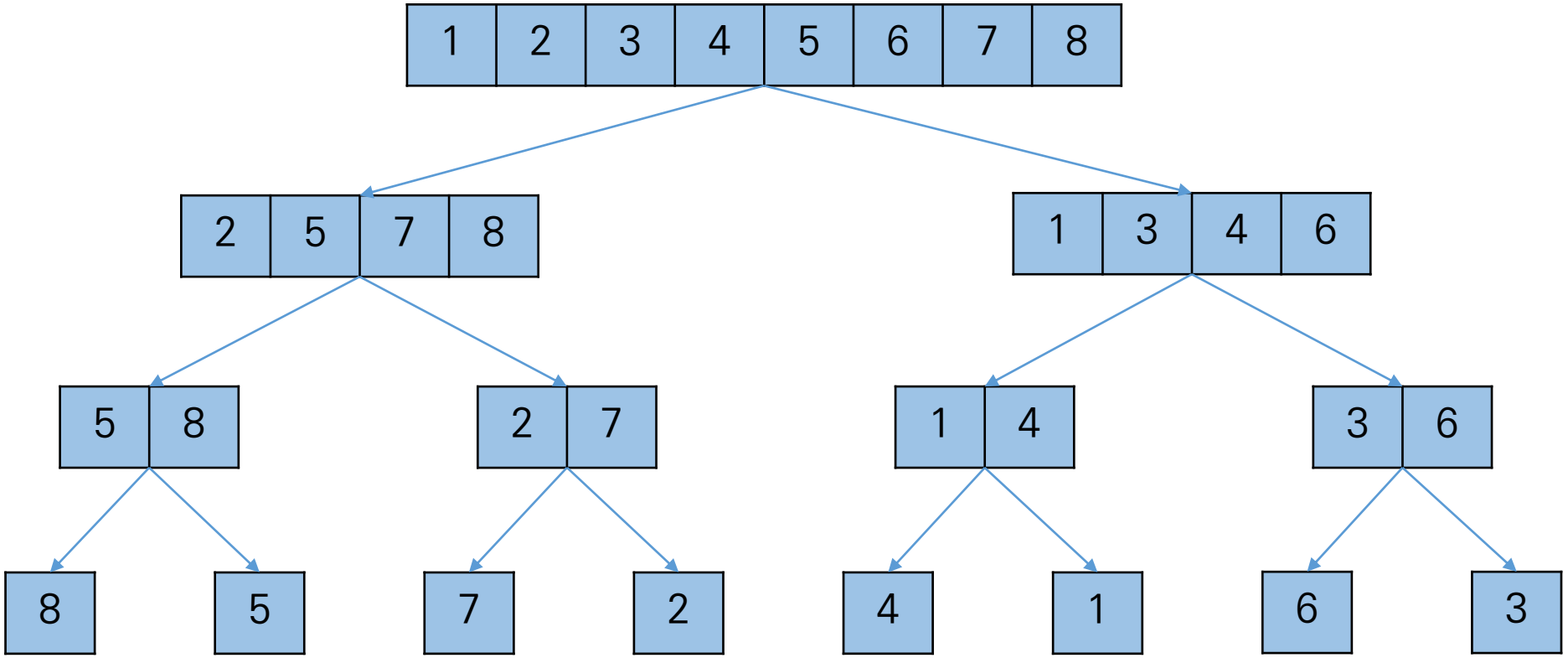
Merge Sort

❖ 정렬 과정을 트리로 표현



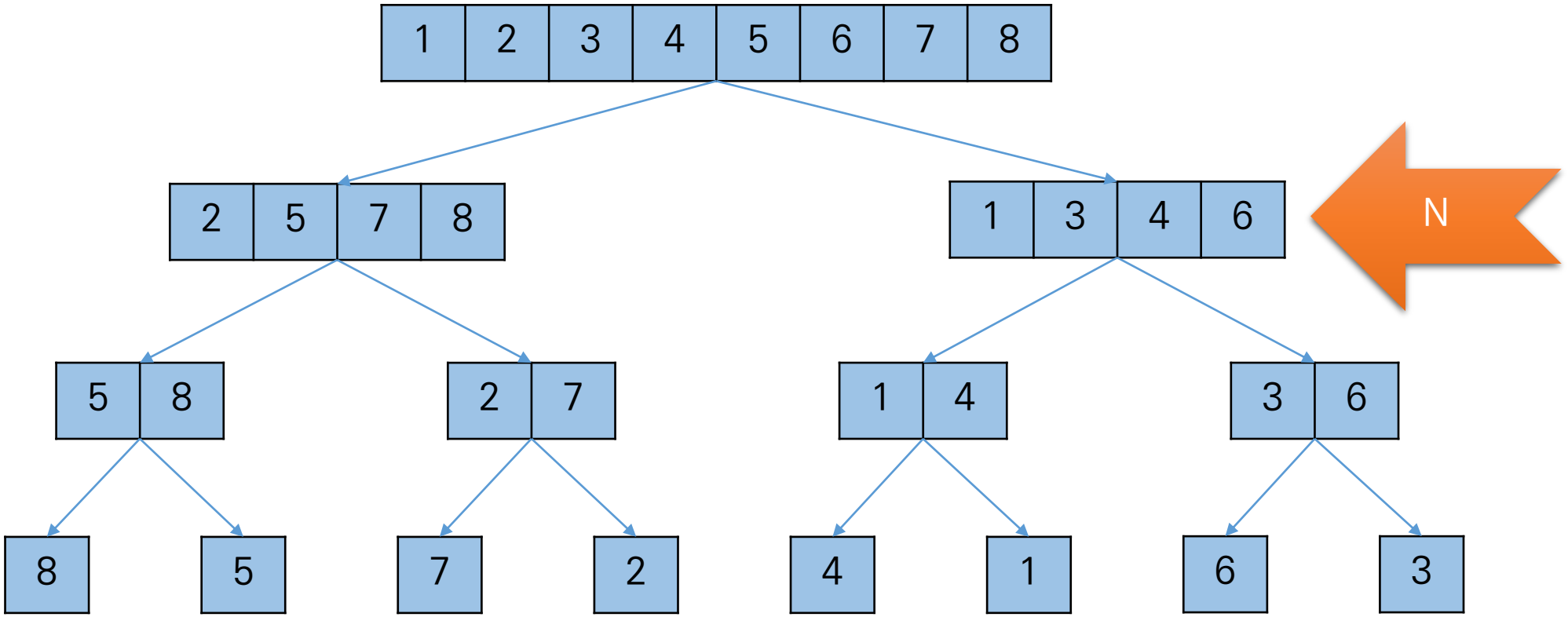
Merge Sort

❖ 정렬 과정을 트리로 표현



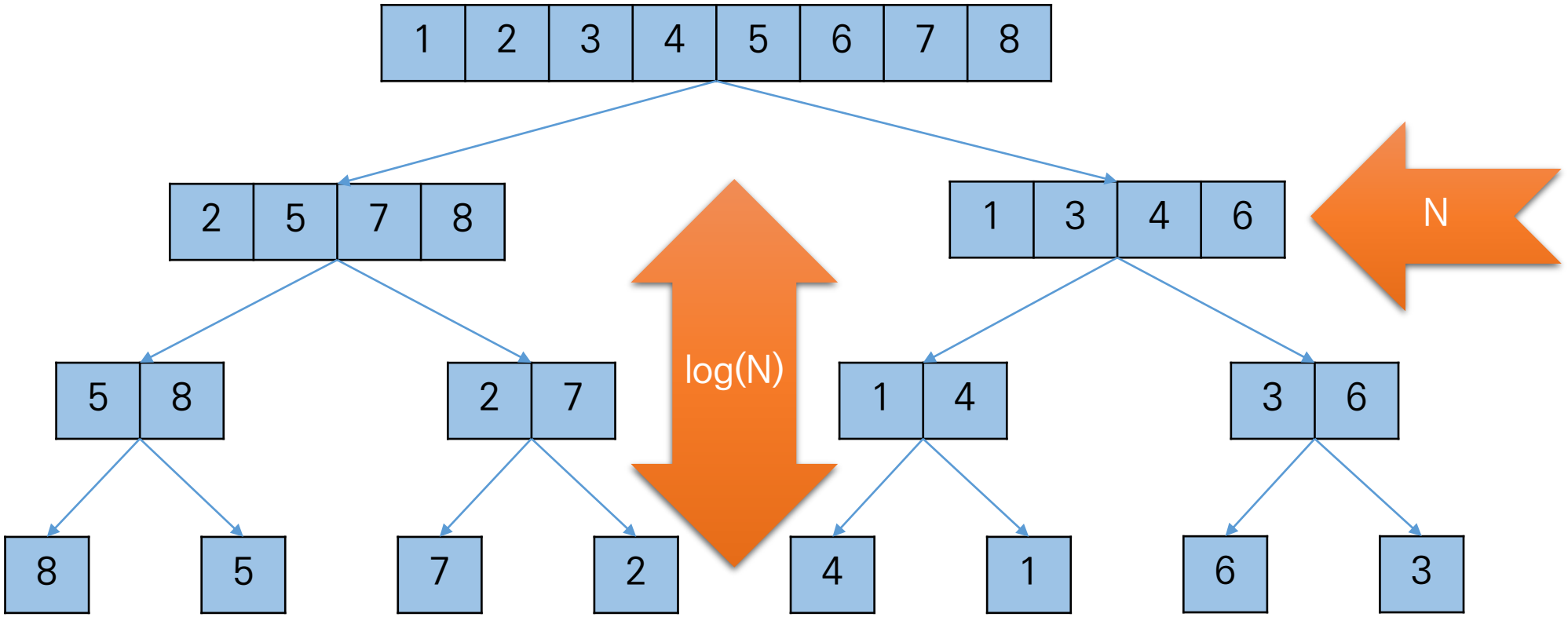
Merge Sort

❖ 시간복잡도(Time Complexity)



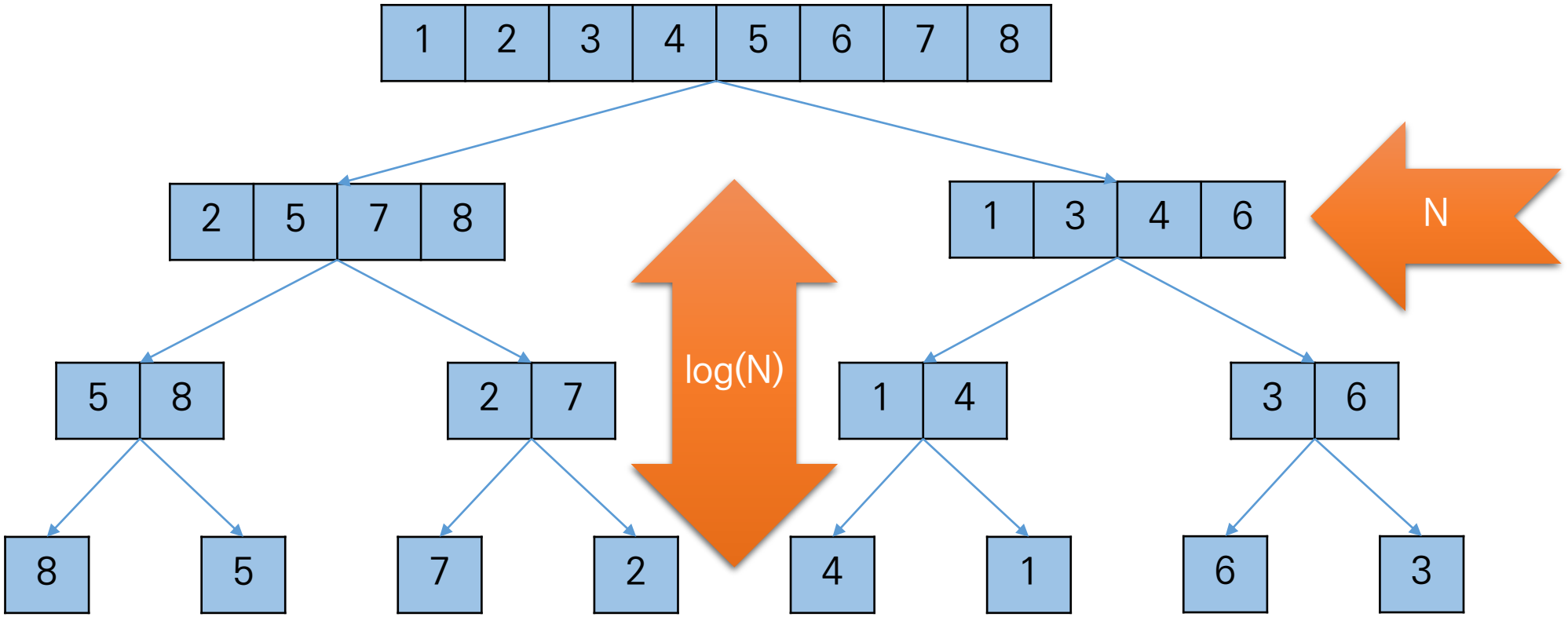
Merge Sort

❖ 시간복잡도(Time Complexity)



Merge Sort

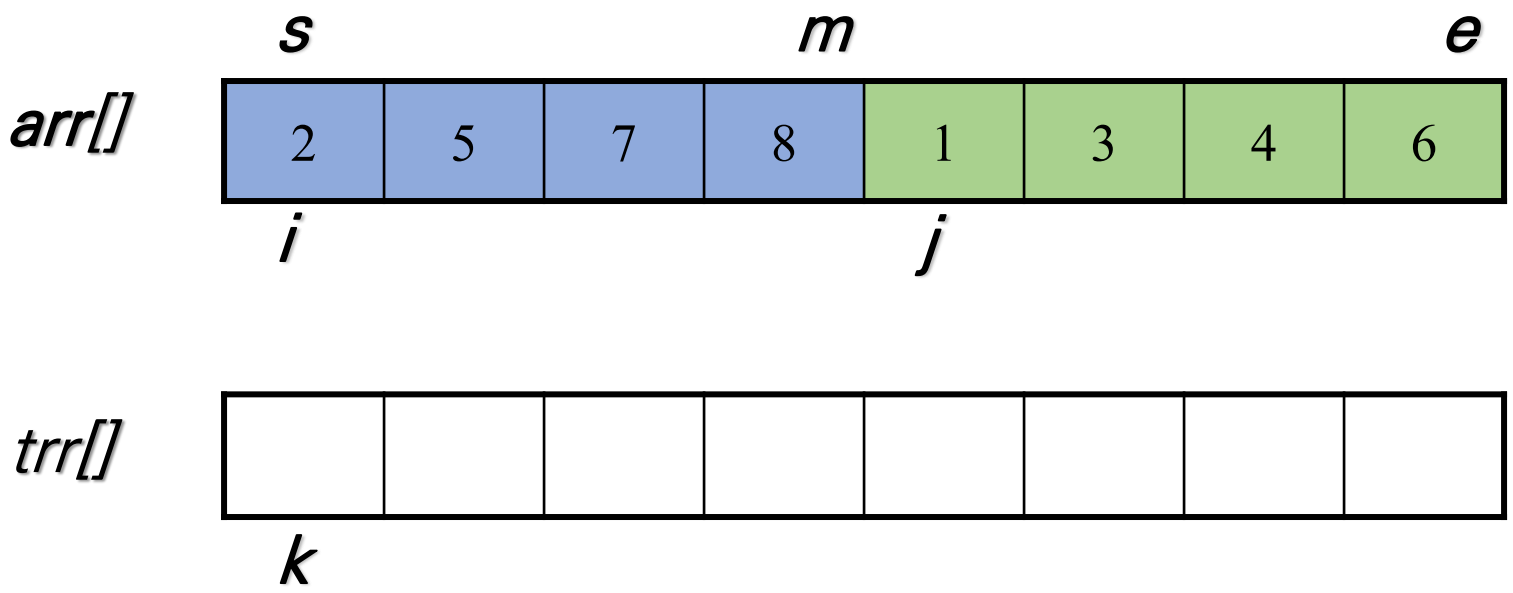
❖ 시간복잡도(Time Complexity) : O(N * log(N))



Merge Sort

❖ merge(comdbine) 과정 : tow pointer 를 이용한다.

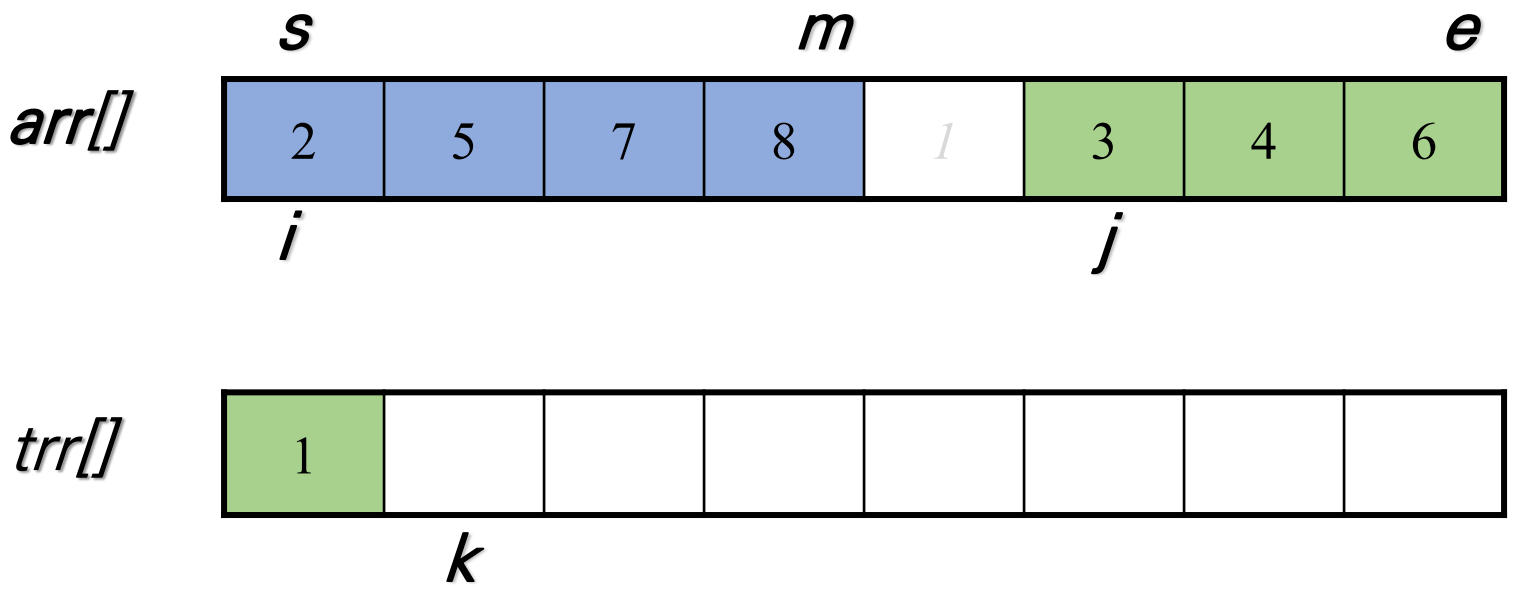
- 앞쪽 절반과 뒤쪽 절반이 각각 정렬된 상태이다.



Merge Sort

❖ merge(comdbine) 과정

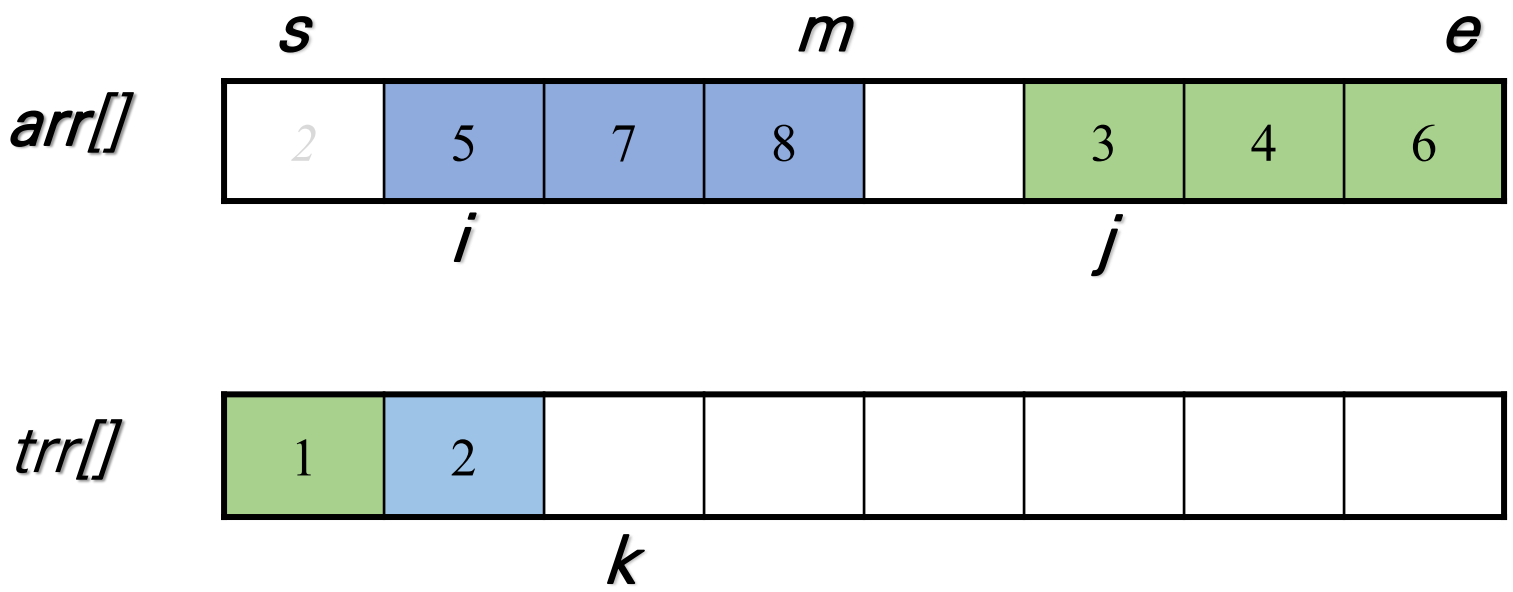
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 1을 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

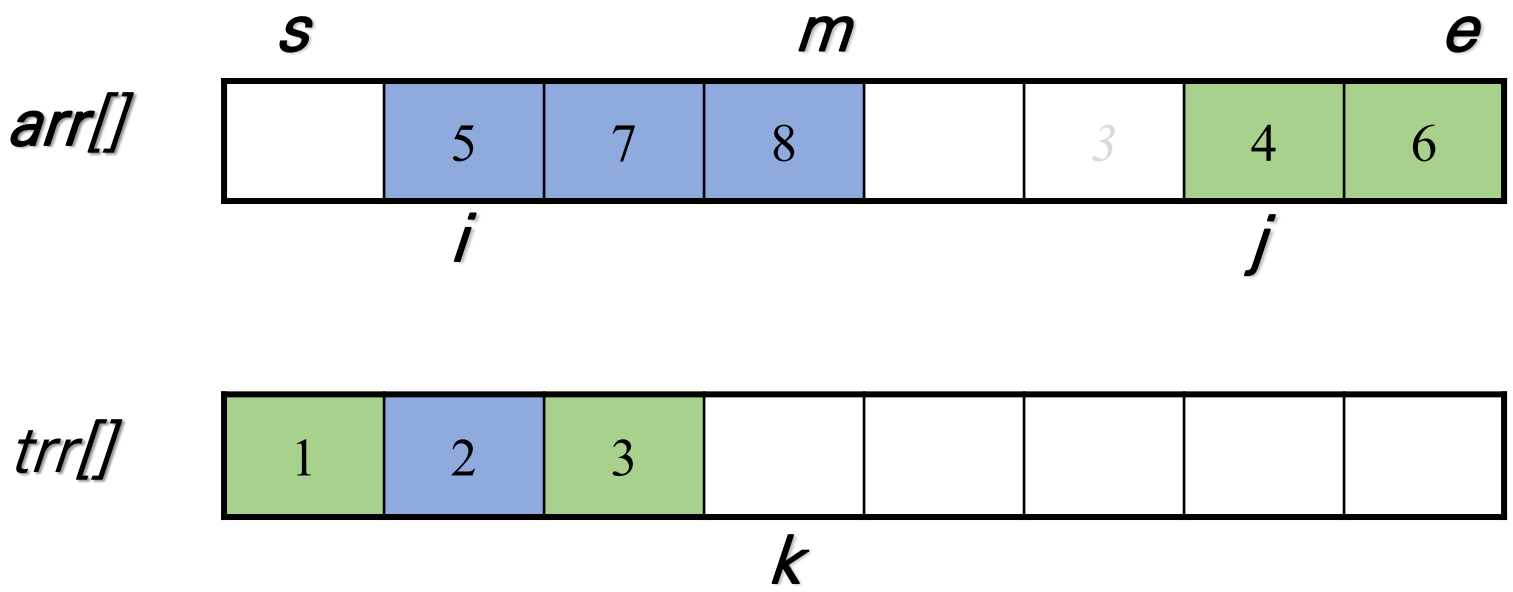
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 2를 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

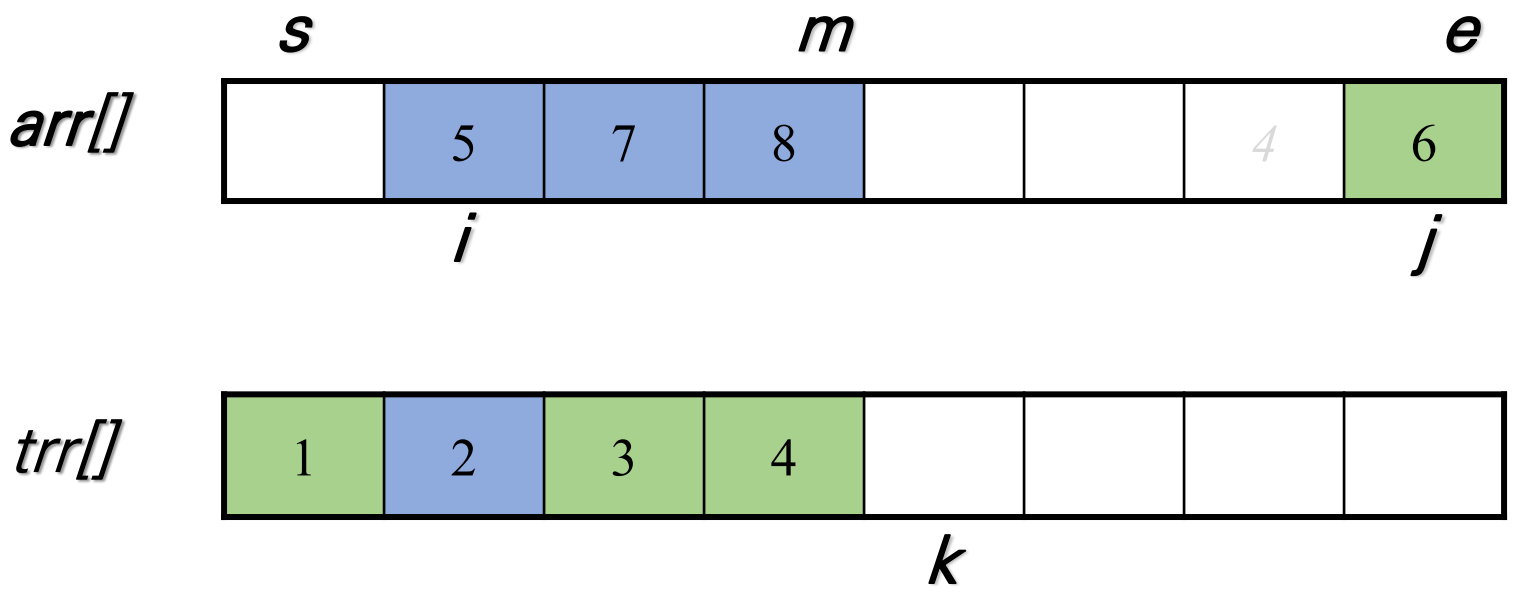
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 3을 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

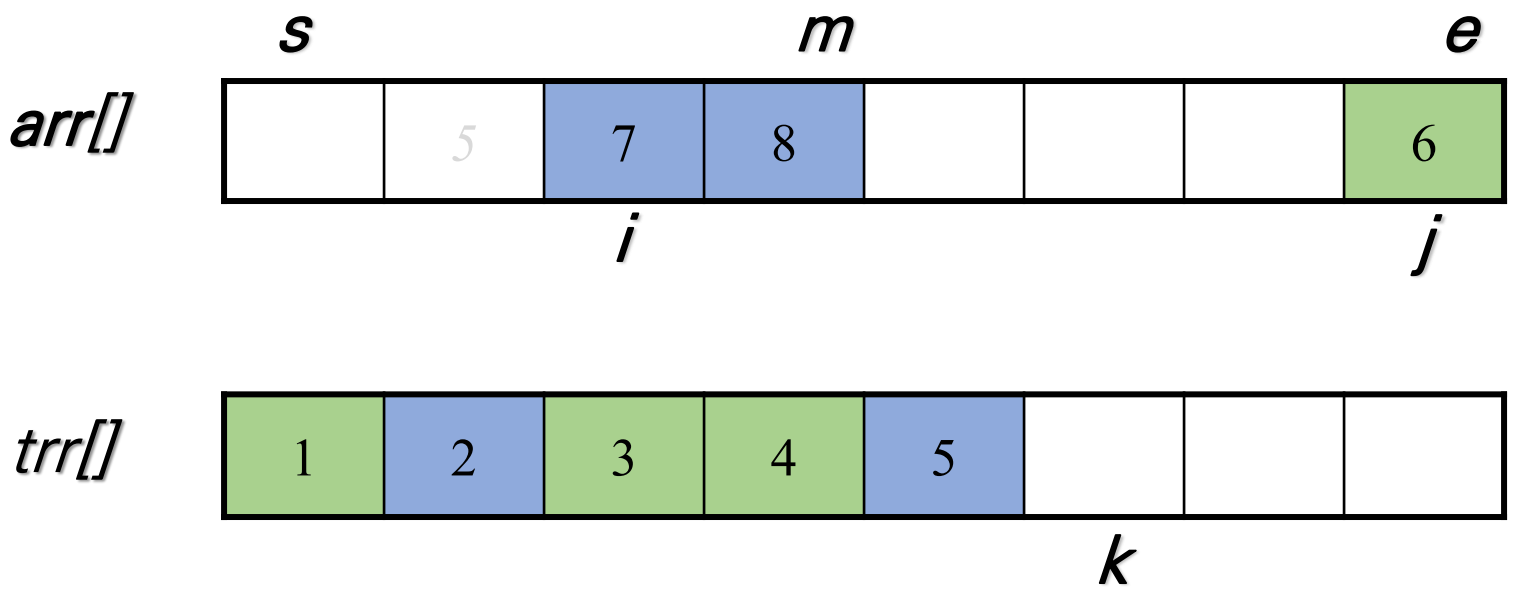
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 4를 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

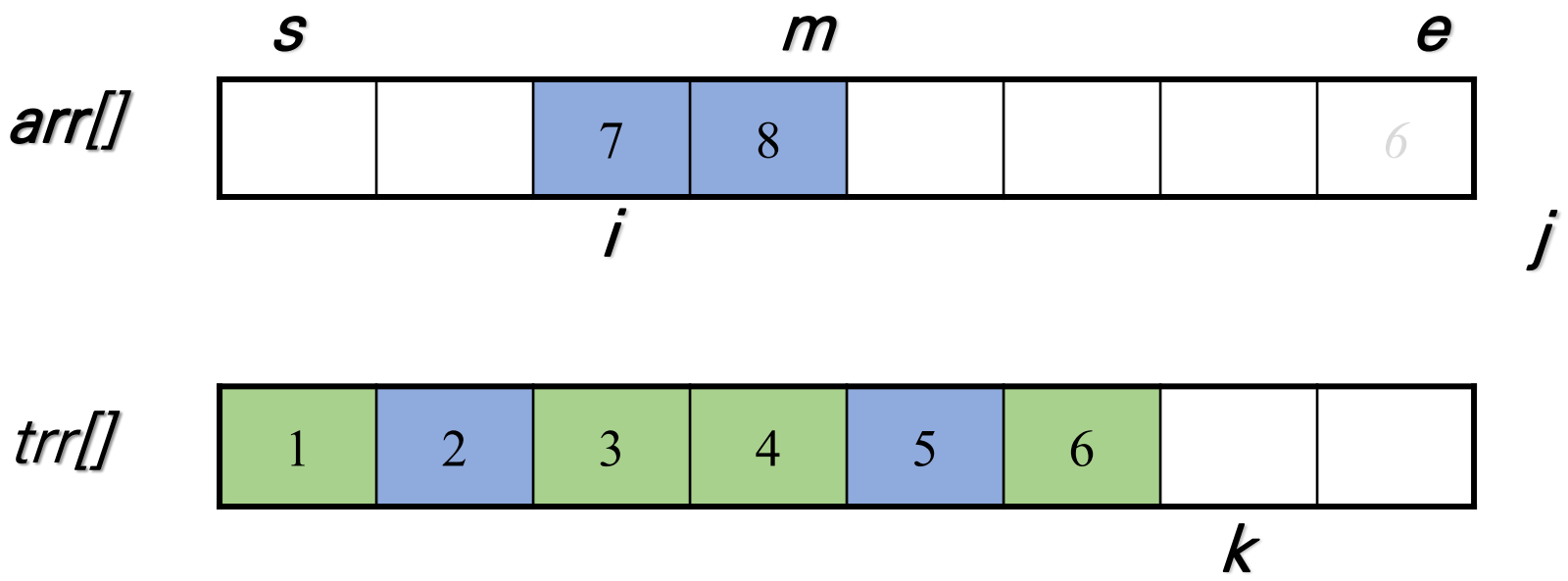
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 5를 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

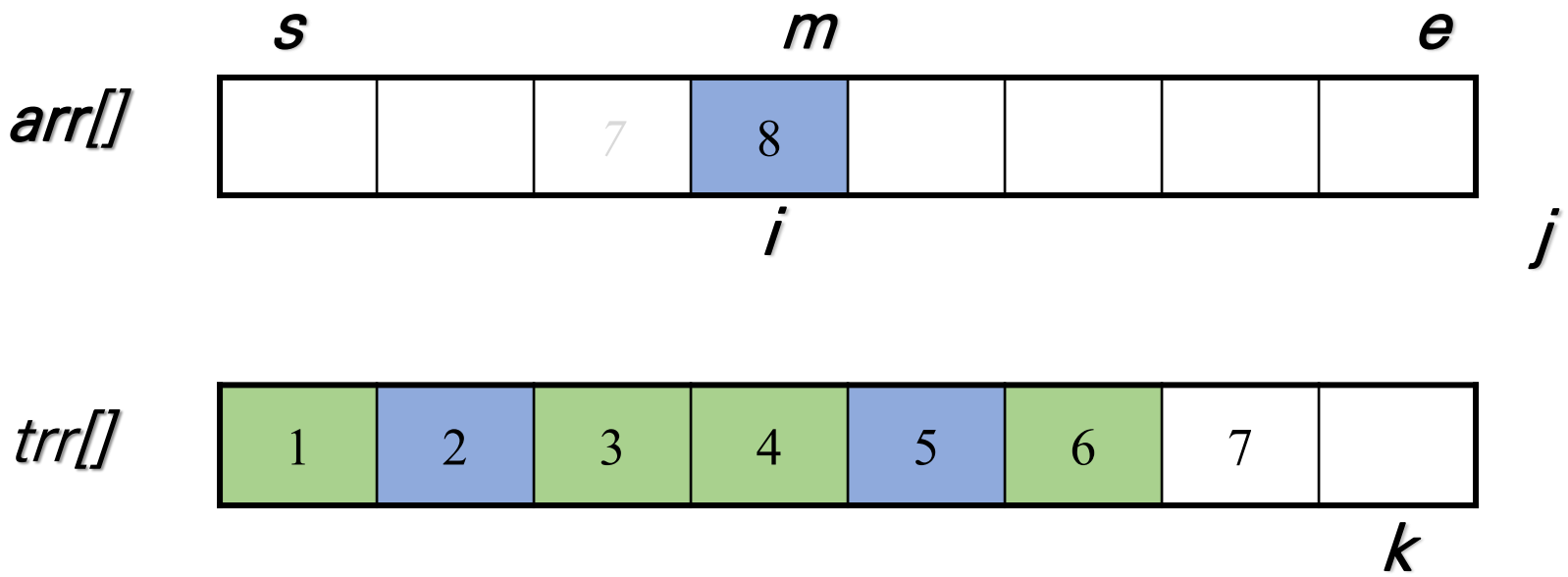
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 6을 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

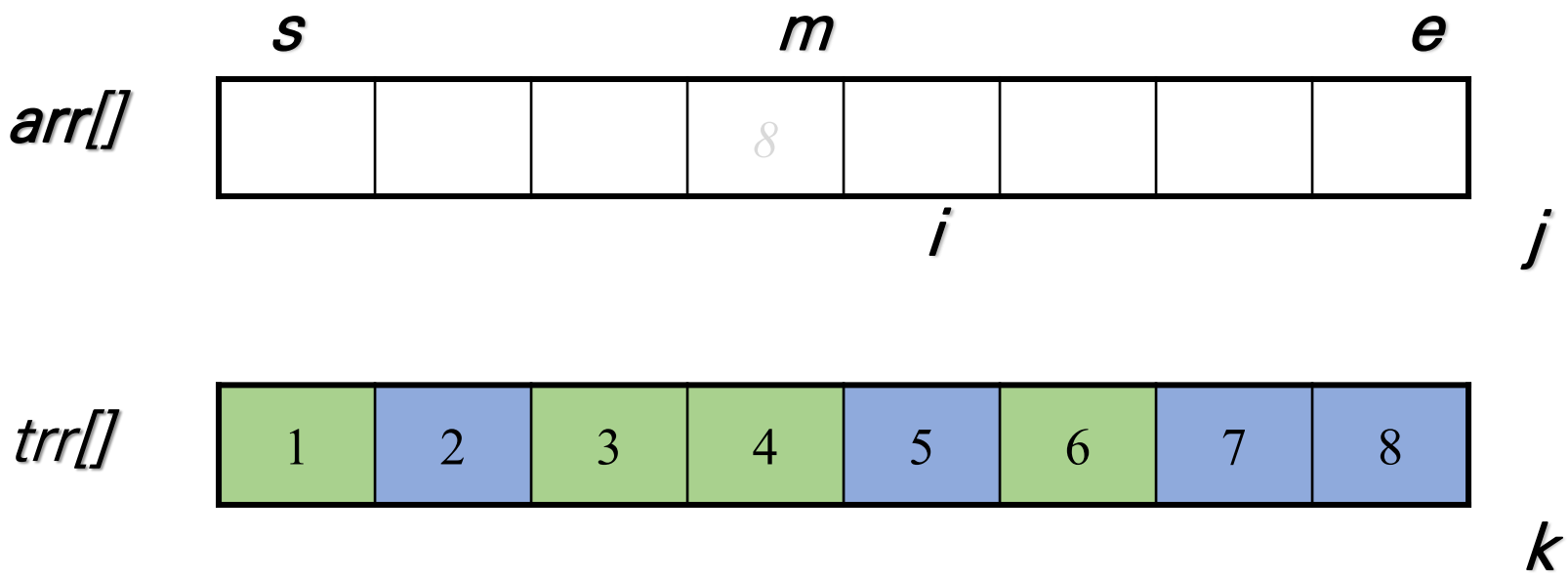
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 7을 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정

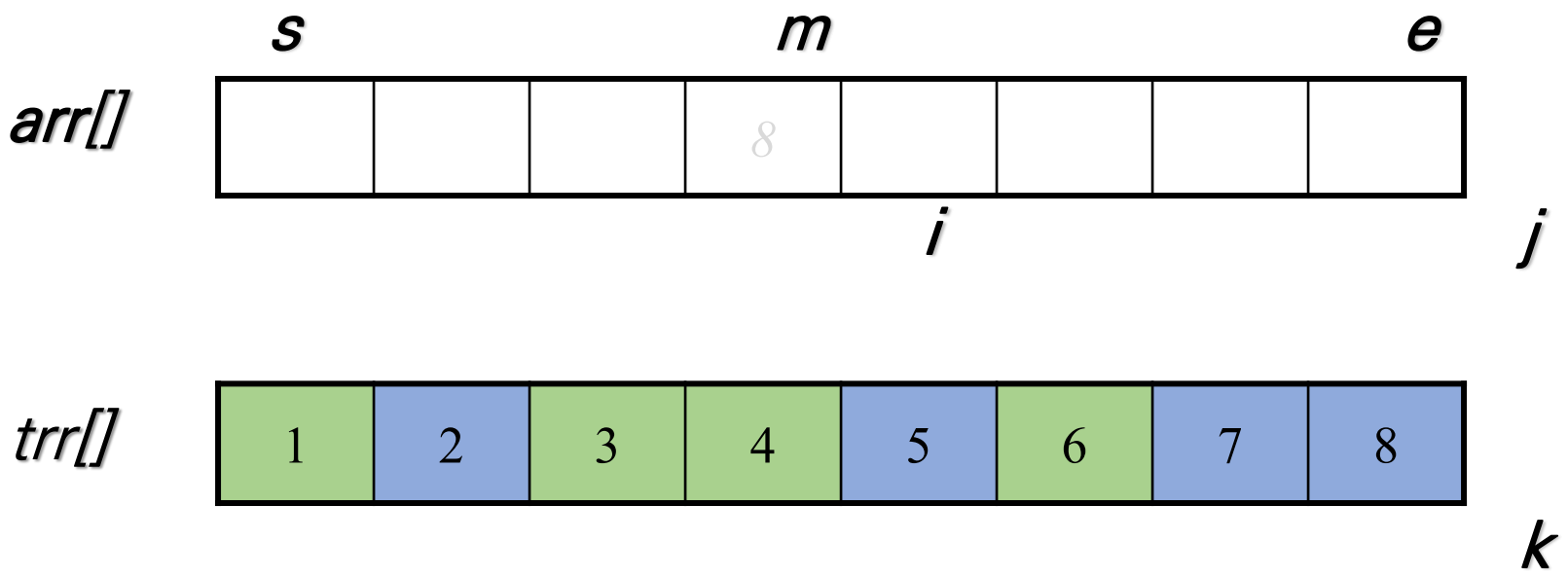
- i 번째 값과 j번째 값을 비교하여 더 작은 값이 8을 trr[k]에 넣는다.



Merge Sort

❖ merge(comdbine) 과정 시간복잡도 : O(N)

- i 번째 값과 j번째 값을 비교하여 더 작은 값이 8을 trr[k]에 넣는다.



합병정렬(merge sort) – code sample 1

```
int trr[SIZE];
void mergeSort(int *arr, int s, int e)
{
    if (s >= e) return; // ---------------------- 1. base condition

    int m = (s + e) / 2;        // ---------------------- 2. divide &
    mergeSort(arr, s, m);       // ----------------------    conquer
    mergeSort(arr, m+1, e);

    int i = s, j = m+1, k = s; // ---------------------- 3. merge
    while (i<=m && j<=e) {
        if (arr[j] < arr[i]) trr[k++] = arr[j++]; // to be stable
        else trr[k++] = arr[i++];
    }

    while (i<=m) trr[k++] = arr[i++];
    while (j<=e) trr[k++] = arr[j++];

    for (i=s;i<=e;i++) arr[i] = trr[i]; // ---------------------- 4. copy
}
```

합병정렬(merge sort) – code sample 2

```
int trr[SIZE];
void mergeSort(int *arr, int s, int e)
{
    if (s >= e) return;         // ---------------------- 1. base condition

    int m = (s + e) / 2;        // ---------------------- 2. divide &
    mergeSort(arr, s, m);       // ----------------------    conquer
    mergeSort(arr, m+1, e);

    int i = s, j = m+1, k = s; // ---------------------- 3. merge
    for (k=s;k<=e;k++) {
        if (j > e) trr[k] = arr[i++];
        else if (i > m) trr[k] = arr[j++];
        else if (arr[j] < arr[i]) trr[k++] = arr[j++]; // to be stable
        else trr[k++] = arr[i++];
    }

    for (i=s;i<=e;i++)          // ---------------------- 4. copy
        arr[i] = trr[i];
}
```

감사합니다.^^