분할 정복

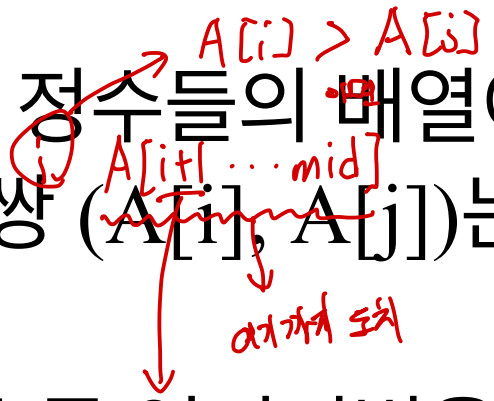
이충기

명지대학교
컴퓨터공학과

```
// 정렬된 부분 배열 A[low . . mid] 와 A[mid+1 . . high]를 합병한다.
MERGE(A[ ], low, mid, high) {
1   크기가 (high + 1)인 배열 B를 만든다
2   h = low;  i = low;  j = mid +1
3   while (i ≤ mid &&  j ≤ high) {
4       if (A[i] ≤ A[j] { B[h] = A[i]; i = i + 1  }
5       else { B[h] = A[j]; j = j + 1 }
6       h = h+1
    }
7   if (i > mid)
8       for (k = j; k ≤ high; k++) {  B[h] = A[k];  h = h + 1 }
9   else
10      for (k = i; k ≤ mid; k++) {  B[h] = A[k]; h = h + 1  }
11  for (k = low; k ≤ high; k++) A[k] = B[k];
}
```

치들의 개수 구하기

의 서로 다른 정수들의 배열이
A[j]이면 한 쌍 (A[i], A[j])는



1. 배열 A 내에 있는 도치들의 수를 억지기법을 사용하여 구하는 알고리즘을 작성하라.
2. 배열 A 내에 있는 도치들의 수를 구하는 분할 정복 알고리즘을 작성하라. 알고리즘의 최악의 시간복잡도는 O(N log N)이어야 한다. 배열 내에 있는 정수들의 순서를 바꿀 수 있다. (힌트: 합병 정렬을 이용하라)

① 반문제

```java
public static int find(int[] list){
    int count = 0; // 1. 도치의 개수를 저장할 변수를 선언하고 초기화한다. (count)
    int length = list.length; // 2. 입력받은 배열의 크기를 저장한다.
    for(int i = 0; i < length - 1; i++){ // 3. 정수 변수 i를 선언하고 i가 0에서 배열 크기 - 2 까지 증가하도록 하는 반복문 속에서
        for(int j = i + 1; j < length; j++){ // 정수 변수 j를 선언하여 j가 i + 1에서 배열 크기 - 1 까지 증가하도록 하는 반복문을 만든다.
            if(list[i] > list[j]){ // 4. 이때 반복문을 돌면서, list[i]가 list[j]보다 크다면 도치의 개수를 1 증가시킨다.
                count ++;
            }
        }
    }
    return count; // 5. 반복문이 모두 끝나고 나면 count에는 도치의 개수가 저장되게 된다.
}
```

1. 도치의 개수를 저장할 변수를 선언하고 초기화한다. (count)
2. 입력받은 배열의 크기를 저장한다.
3. 정수 변수 i를 선언하고 i가 0에서 배열 크기 - 2 까지 증가하도록 하는 반복문 속에서
정수 변수 j를 선언하여 j가 i + 1에서 배열 크기 - 1 까지 증가하도록 하는 반복문을 만든다.
4. 이때 반복문을 돌면서, list[i]가 list[j]보다 크다면 도치의 개수를 1 증가시킨다.
5. 반복문이 모두 끝나고 나면 count에는 도치의 개수가 저장되게 된다.

② 문제 2번

```java
public static int mergeSort(int[] list, int low, int high){ // list는 입력받은 배열, low는 시작 지점, high는 끝 지점
    int mid = 0;
    int left_count = 0;
    int right_count = 0;
    // 1. 분할 지점의 인덱스를 저장할 정수 변수와, 왼쪽 부분과 오른쪽 부분에 대한 도치를 저장하는 정수 변수를 선언한다.

    if(low < high){ // 2. 입력받은 low가 high보다 작다는 것이 확인되면
        mid = (low + high) / 2; // 3. 중간 지점을 (low + high) / 2로 정하고
        left_count = mergeSort(list, low, mid); // low부터 mid까지에 대한 합병 정렬을 한 뒤 도치의 개수를 (왼쪽 부분에 대한) 저장한다.
        right_count = mergeSort(list, low: mid+1, high); // mid+1부터 high까지에 대한 합병 정렬을 한 뒤 도치의 개수를 (오른쪽 부분에 대한) 저장한다.
        return left_count + right_count + merge(list, low, mid, high); // 4. 상술했던 두 부분을 합병 정렬하고 계산되는 도치의 개수를 저장한 뒤, 각 두 부분에 대한 도치의 개수를 더한다.
    }
    return 0; // 그외에는 0을 리턴
}
```

```java
public static int merge(int[] list, int low, int mid, int high){ // list는 입력받은 배열, low는 시작 지점, mid는 분할 지점, high는 끝 지점
    int count = 0;
    int[] B = new int[high + 1];
    int h = low;
    int i = low;
    int j = mid + 1;
    // 1. 도치의 개수를 저장할 정수변수, high + 1크기의 정수 배열 B를 선언하고
    // low를 저장하는 정수 변수 2개(각각 왼쪽 부분을 돌기 위한 변수 i와 생성된 배열을 돌기 위한 h)와
    // mid + 1을 저장하는 정수 변수(오른쪽 부분을 돌기 위한 변수 j)를 선언한다.

    while(i <= mid && j <= high){ // 2. i가 mid보다 작거나 같고 j가 high보다 작거나 같은 동안 다음과 같이 수행한다 :
        if(list[i] <= list[j]){ // 2-1. list[i]가 list[j]보다 작거나 같은 경우 도치는 발생하지 않고,
            B[h] = list[i]; // B[h]에 list[i]를 저장한다.
            i = i + 1; // i에 i + 1을 저장한다.
        }
        else { //2-2. 그 외의 경우에는
            B[h] = list[j]; // B[h]에 list[j]를 저장한다.
            j = j + 1; // j에 j + 1을 저장한다.
            count += mid + 1 - i; // 도치의 개수에 mid + 1 - i를 더한다.
        }
        h = h + 1; // h에 h + 1을 저장한다.
    }

    if(i > mid){ // 3. 반복문이 모두 종료된 후,
        for(int k = j; k <= high; k++){ // 3-1. i > mid라면 j부터 high까지 증가하는 정수변수 k를 선언하고
            B[h] = list[k]; // B[h]에 list[k]를 저장한다.
            h = h + 1; // h에 h + 1을 저장한다.
        }
    }
    else{
        for(int k = i; k <= mid; k++){ // 3-1. 그 외의 경우에는 i부터 mid까지 증가하는 정수변수 k를 선언하고
            B[h] = list[k]; // B[h]에 list[k]를 저장한다.
            h = h + 1; // h에 h + 1을 저장한다.
        }
    }
    for(int k = low; k <= high; k++){ // 4. low부터 high까지 증가하는 정수변수 k에 대한 반복문을 만들어 B에 있는 요소들을 low 부터 high까지 원래 list에 넣는다.
        list[k] = B[k];
    }

    return count; // 5. 그러면 합병 정렬을 하면서 count에 도치의 개수가 저장된다.
}
}
```

// list는 입력받은 배열, low는 시작 지점, high는 끝 지점

1. 분할 지점의 인덱스를 저장할 정수 변수와, 왼쪽 부분과 오른쪽 부분에 대한 도치를 저장하는 정수 변수를 선언한다.

2. 입력받은 low가 high보다 작다는 것이 확인되면

2-1. 중간 지점을 (low + high) / 2로 정하고 low부터 mid까지에 대한 합병 정렬을 한 뒤 도치의 개수를 (왼쪽 부분에 대한) 저장한다.
또, mid+1부터 high까지에 대한 합병 정렬을 한 뒤 도치의 개수를 (오른쪽 부분에 대한) 저장한다. 상술했던 두 부분을 합병 정렬하고 계산되는 도치의 개수를 저장한 뒤, 각 두 부분에 대한 도치의 개수를 더한다.

2-2. 입력받은 low가 high보다 작지 않다면 0 리턴

// list는 입력받은 배열, low는 시작 지점, mid는 분할 지점, high는 끝 지점

1. 도치의 개수를 저장할 정수변수, high + 1크기의 정수 배열 B를 선언하고 low를 저장하는 정수 변수 2개(각각 왼쪽 부분을 돌기 위한 변수 i와 생성된 배열을 돌기 위한 h)와 mid + 1을 저장하는 정수 변수(오른쪽 부분을 돌기 위한 변수 j)를 선언한다.

2. i가 mid보다 작거나 같고 j가 high보다 작거나 같은 동안 다음과 같이 수행한다 :

2-1. list[i]가 list[j]보다 작거나 같은 경우 도치는 발생하지 않고, B[h]에 list[i]를 저장한다. 또, i에 i + 1을 저장한다.

2-2. 그 외의 경우에는 B[h]에 list[j]를 저장한다. j에 j + 1을 저장한다. 또, 도치의 개수에 mid + 1 - i를 더한다.

2-3. 위 조건문이 끝나고 나면 h에 h+1을 저장한다.

3. 반복문이 모두 종료된 후,

3-1. i > mid라면 j부터 high까지 증가하는 정수변수 k를 선언하고 B[h]에 list[k]를 저장한다. 또, h에 h + 1을 저장한다.

3-2. 그 외의 경우에는 i부터 mid까지 증가하는 정수변수 k를 선언하고 B[h]에 list[k]를 저장한다. h에 h + 1을 저장한다.

4. low부터 high까지 증가하는 정수변수 k에 대한 반복문을 만들어 B에 있는 요소들을 low 부터 high까지 원래 list에 넣는다.

5. 그러면 합병 정렬을 하면서 count에 도치의 개수가 저장된다.