병합 정렬 (Merge Sort)

> 단순하지 않은 정렬 쪽에 가장 단순한 정렬

특징

- 합병의 대상이 되는 두 배열의 영역이 각각 **이미 정렬**이 되어 있기 때문에,
  두 배열을
  순차적으로 비교하며 정렬 할 수 있다.
- 합볍정렬은 **순차적인 비교로 정렬**하므로,

  linkedList를 정렬할 때 사용하면 효율적이다.
  - 퀵 소트는 순차 접근이 아닌 pivot을 **임의 접근**으로 접근하기 때문에 비효율적
    - linkedList는 접근 연산에서 비효율적임.
    - 임의 접근 시, 오버헤드 발생

진행 순서

1. 왼쪽, 오른쪽, 중앙 index 설정
2. 배열의 왼쪽 ~ 중앙, 중앙 + 1 ~ 오른쪽 구간을 나눔
3. 왼쪽 == 오른쪽 될때까지 위 과정 반복
4. 왼쪽 == 오른쪽이면 정렬 후 재귀를 빠져 나옴.
5. 위 과정 반복

코드

```
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right-1)/2;

        // 분할
        mergeSort(arr, left, mid);
        mergeSort(arr, mid+1, right);

        // 정복
```

```
        merge(arr, left, mid, right)
    }
}
```

```
void merge(int arr[], int left, int mid, int right) {
    int i = 0;
    int j = 0;
    int k = left;
    while ( i < n1 && j < n2){
        if (L[i] <= R[j]){
            arr[k] = L[i++];
        }
        else {
            arr[k] = R[j++];
        }
        k++;
    }

    while(i < leftArr.length()){
        arr[k++] = leftArr[i++];
    }
    while(j < rightArr.length()){
        arr[k++] = rightArr[j++];
    }
}
```

퀵 소트와 차이점

- 퀵 소트
  - 피봇을 통해 정렬(partition()) 후 영역을 쪼갬(quickSort())
- 병합 정렬
  - 영역을 쪼갤수 있을 만큼 쪼갬(mergeSort()) 후 정렬(merge())

시간 복잡도

최선 평균 최악

$O(NlogN)$ $O(NlogN)$ $O(NlogN)$

장점

- 안정 정렬
- 언제든 시간복잡도 O(N * log N) 보장

단점

- 정렬을 위한 추가적인 배열이 필요함
  - 메모리 공간이 추가적으로 필요함
  - 정렬에 사용되는 배열은 전역
    - 함수내에서 정렬 배열을 사용하면 매번 배열을 선언해야해서 메모리 낭비가 큼