Merge Sort (병합 정렬)

Algorithm, Sort

```
A[p ... r]을 정렬한다.
{
    if(p < r) then {
        q ← (p+r)/2        // 1.중간 지점 계산
        mergeSort(A, p, q);    // 2-1. 전반부 정렬
        mergeSort(A, q+1, r)    // 2-2. 후반부 정렬
        merge(A, p, q, r);    // 3. 병합
    }
}

merge(A[], p, q, r)
{
  정렬된 부분배열 2-1, 2-2를 병합 후
  재정렬, 하나의 배열로 만든다.
}
```

Java 구현

```
import java.util.ArrayList;

public class MergeSort {

    public static void main(String[] args) {
        int[] arr = {100, 99, 97, 1, 3, 0, -15, 30, 36/6, 7, 2};

        mergeSort(arr, 0, arr.length-1);

        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i] + " ");
    }

    private static void mergeSort(int[] arr, int startIndex, int endIndex) {
        if (startIndex < endIndex) {
            int middleIndex = (startIndex + endIndex) / 2;

            mergeSort(arr, startIndex, middleIndex);
            mergeSort(arr, middleIndex + 1, endIndex);

            merge(arr, startIndex, middleIndex, endIndex);
        } else
            return;
    }
```

```
/**
  1. 두 배열을 합한 크기만큼의 비어있는 임시 배열 준비
  2. 첫번째 요소끼리 비교하여 작은 요소를 "1"에 추가. 정렬된 원소 삭제.
  3. 양쪽 배열이 빌 때까지 "2" 반복.
*/
    private static void merge(int[] arr, int startIndex, int middleIndex,
int endIndex) {
        int L_startIndex = startIndex;
        int R_startIndex = middleIndex + 1;
        int tempList_index = 0;

        ArrayList<Integer> tempList = new ArrayList<Integer>(); // 동적 배열

        while (L_startIndex <= middleIndex && R_startIndex <= endIndex) {
            if (arr[L_startIndex] < arr[R_startIndex])
                tempList.add(arr[L_startIndex++]);
            else
                tempList.add(arr[R_startIndex++]);
        }

        while (L_startIndex <= middleIndex)
            tempList.add(arr[L_startIndex++]);

        while (R_startIndex <= endIndex)
            tempList.add(arr[R_startIndex++]);

        while (startIndex <= endIndex)
            arr[startIndex++] = tempList.get(tempList_index++);

        tempList.removeAll(tempList); // 원소 전부 제거
    }
}
```

C 구현

```
#include <stdio.h>
#include <stdlib.h>

void mergeSort(int arr[], int start_index, int end_index);
void merge(int arr[], int start_index, int middle_Index, int end_index);

void main() {
    int arr[] = {100, 99, 97, 1, 3, 0, -15, 30, 36/6, 7, 2};
    int length = sizeof(arr)/sizeof(int);
    int i=0;

    mergeSort(arr, 0, length-1);
```

```
    for (i=0 ; i<length ; i++)
        printf("%d ", arr[i]);

    printf("\n");
}

void mergeSort(int arr[], int start_index, int end_index)
{
    if(start_index < end_index) {
        int middle_index = (start_index+end_index)/2;        // 1.중간지점 계
산

        mergeSort(arr, start_index, middle_index);            // 2-1. 전반부
정렬
        mergeSort(arr, middle_index+1, end_index);            // 2-2. 후반부
정렬

        merge(arr, start_index, middle_index, end_index);    // 3. 병합
    }
    else
        return; // start_index >= end_index, 원소가 딱 하나 남았을 때 탈출!
}

/**
1. 두 배열을 합한 크기만큼의 비어있는 임시 배열 준비
2. 첫번째 요소끼리 비교하여 작은 요소를 "1"에 추가. 정렬된 원소 삭제.
3. 양쪽 배열이 빌 때까지 "2" 반복.
**/
void merge(int arr[], int start_index, int middle_index, int end_index)
{
    int LStart_index = start_index;            //전반부 시작 인덱스
    int RStart_index = middle_index+1;    //후반부 시작 인덱스

    int* tempArr = (int*) malloc(sizeof(int) * (end_index - start_index+1));
// 1. 동적 배열 준비
    int tempArr_index = 0;

    while(LStart_index <= middle_index && RStart_index <= end_index) {
        if(arr[LStart_index] < arr[RStart_index])
            tempArr[tempArr_index++] = arr[LStart_index++];
        else
            tempArr[tempArr_index++] = arr[RStart_index++];
    }

    while (LStart_index <= middle_index)
        tempArr[tempArr_index++] = arr[LStart_index++];

    while (RStart_index <= end_index)
        tempArr[tempArr_index++] = arr[RStart_index++];
```

```
  // 2-1. "원배열 = 임시 배열" 복사.
    tempArr_index = 0;
    while (start_index <= end_index)
        arr[start_index++] = tempArr[tempArr_index++];

    free(tempArr); // 2-2. 원소 삭제
}
```