합병정렬

하나의 리스트를 두 개의 균등한 크기로 분할하고, 분할된 리스트를 정렬한 뒤 합치는 것

분할정복(divide and conquer) 기법에 바탕을 두고 있다.

↳ 순환호출을 이용하여 구현

1. 분할 : 입력 배열을 같은 크기의 두 부분배열로 분할
2. 정복 : 부분 배열 정렬 → 부분배열의 크기가 충분히 작지 않으면 순환호출
3. 결합 : 정렬된 부분배열을 합침

알고리즘

합병정렬 (mergeSort)

```
mergeSort (list, left, right):

if left < right
    mid = (left + right) / 2
    mergesort (list, left, mid);
    mergesort (list, mid+1, right);
    merge (list, left, mid, right);
```

↳ 실제 합병이 되는 단계

합병 (merge)

```
merge (list, left, mid, right):

i ← left                    // i는 왼쪽 부분 배열의 index
j ← mid+1                   // j는 오른쪽 부분 배열의 index
K ← left                    // K는 전체 배열의 index
sorted 배열 생성
while i<= mid && j<=right do
        if (list[i] < list[j])
            then
                sorted[k] = list[i];
                k++
                i++
            else
                sorted[k] = list[j];
                k++
                j++
정렬한 배열을 sorted에 복사
sorted를 list에 복사
```

구현

```
int sorted[MAX];

void merge(int list[], int left, int mid, int right){
    int i, j, k;
    i = left, j = mid+1, k = left;

    while(i <= mid && j <= right) {
        if(list[i] < list[j]) {
            sorted[k] = list[i];
            k++;
            i++;
        } else {
            sorted[k] = list[j];
            k++;
            j++;
        }
    }
    if(i > mid){        // 왼쪽 부분배열이 먼저 끝
        for(int l=j; l<=right; l++) {
            sorted[k] = list[l];
            k++;
        }
    }else {             // 오른쪽이 먼저 끝
        for(int l=i; l<=mid; l++) {
            sorted[k] = list[l];
            k++;
        }
    }
    for (int l= left; l<=right; l++)
        list[l] = sorted[l];
}
```

```
void merge_sort (int list[], left, right) {
        int mid;
        if (left < right) {
                mid = (left + right)/2
                merge_sort (list, left, mid);
                merge_sort (list, mid+1, right);
                merge (list, left, mid, right);
        }
}
```

시간복잡도

Let, list의 요소 개수 n은 2의 거듭제곱

$n = 2^k$ then $k = \log_2 n$

순환은 k번 진행된다. ex) $8 \to 4 \to 2 \to 1$ (3번 재귀) ⇒ 깊이가 3

↳ k번 만큼 합병이 일어난다. ∴ 재귀의 시간복잡도는 $O(k) = O(\log_2 n)$

각 합병단계에서는 최대 n번 비교연산이 필요하다.

∴ 합병정렬의 시간복잡도 : $O(n \log_2 n)$

☆ 입력되는 데이터와 무관하게 정렬 시간이 같다.

⇒ 최악, 최선, 평균이 같다.

단점

1. 임시 배열 필요
2. n이 큰 경우 이동 횟수가 많아져서 오래걸림

↳ 합병 정렬을 연결 리스트로 구성하면 해결가능