dongguk UNIVERSITY

자료 구조와 실습

실습 #11 : Sorting

Eun Man Choi
emchoi@dgu.ac.kr

목 차

- Sorting
  - Sorting Algorithms
- 실습 문제
  - Insertion sort, quick sort, merge sort구현

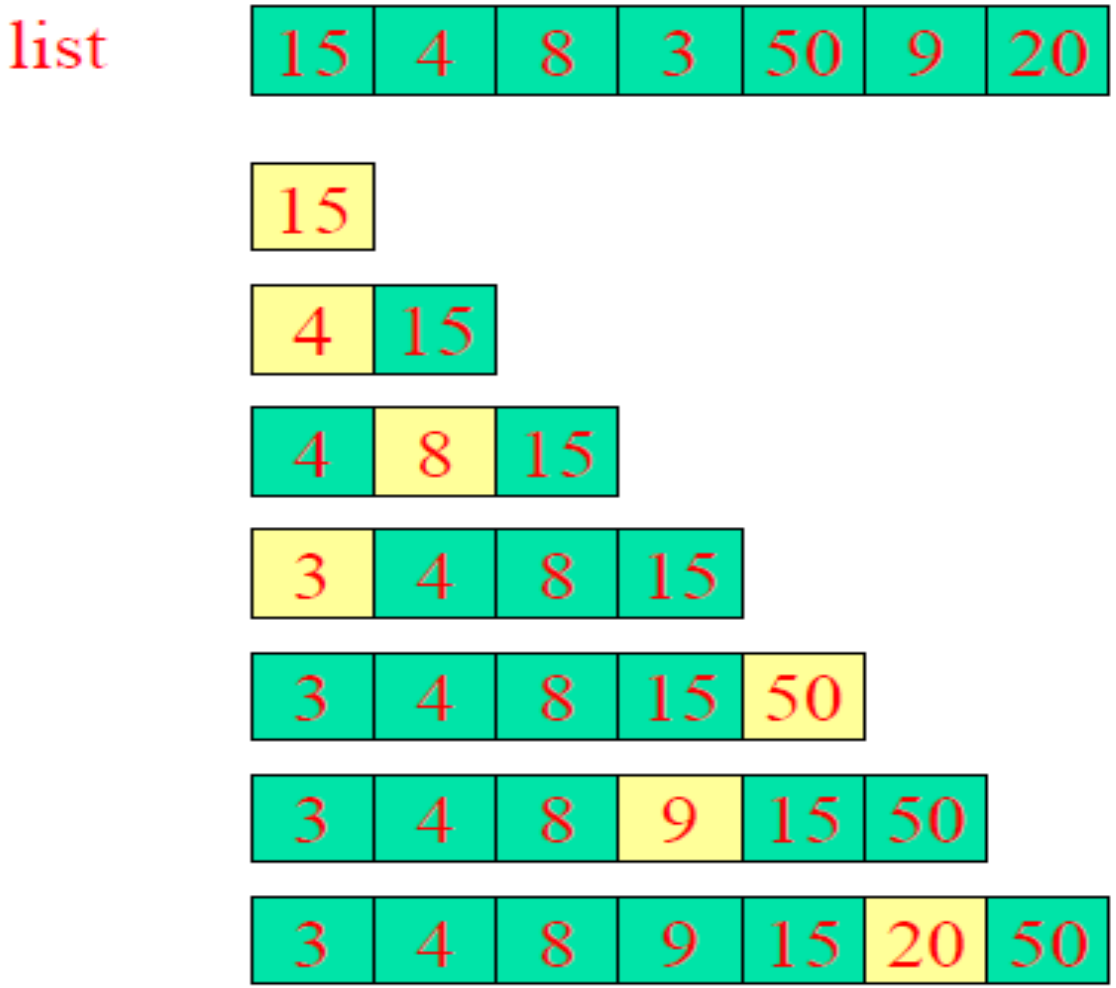
Sorting

- 정렬
  - 자료를 특정 키 값에 따라 오름차순(내림차순)으로 정리

- 정렬의 필요성
  - 대부분의 기관에서 자료 정렬에 20%-50% 계산 시간을 소비

- 정렬방법
  - 내부정렬(internal sorting) : 메인 메모리 내에서 정렬
    - Bubble sort, Insertion sort, Quick sort, Shell sort, Heap sort, Radix sort…
  - 외부정렬(external sorting) : 자료의 양이 방대하여 보조기억 장치를 활용하여 정렬

Insertion Sort



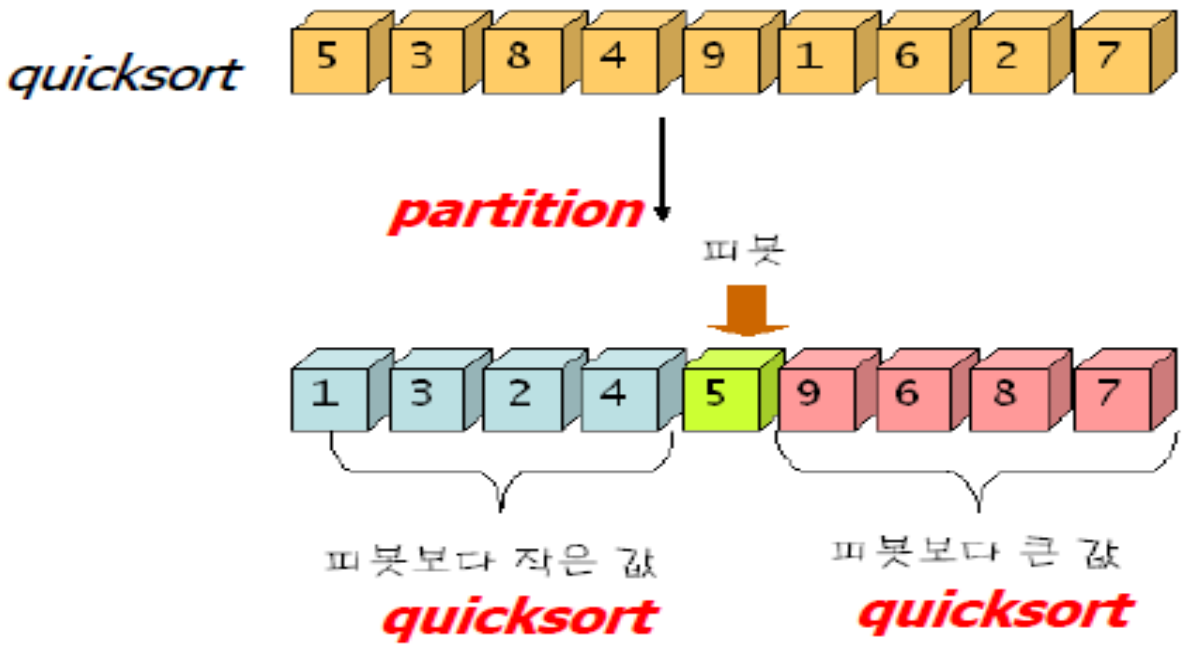
Insertion Sort

- Time Complexity
  - worst case : 1 + 2 + … + (n-1) = O(n^2)
  - best case : O(n)

- 장점
  - 알고리즘이 간단함
  - 안정성이 있음
  - 거의 정렬된 리스트에서는 매우 효율적임

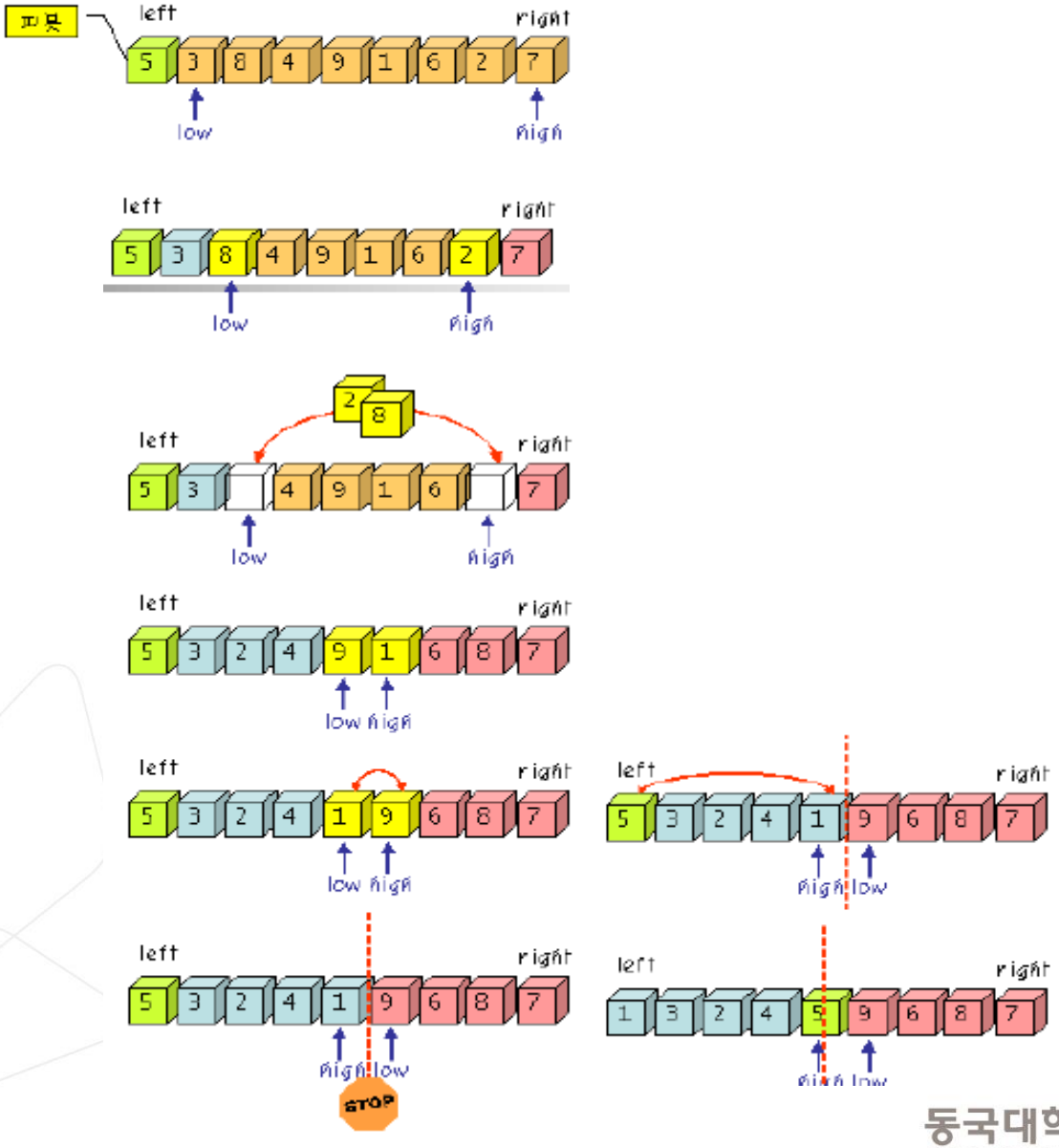
Quick Sort

- Divide – and – conquer(분할정복)
  - 피봇 키를 사용하여 리스트를 분할
  - Recursive하게 두 번의 quick sort 호출



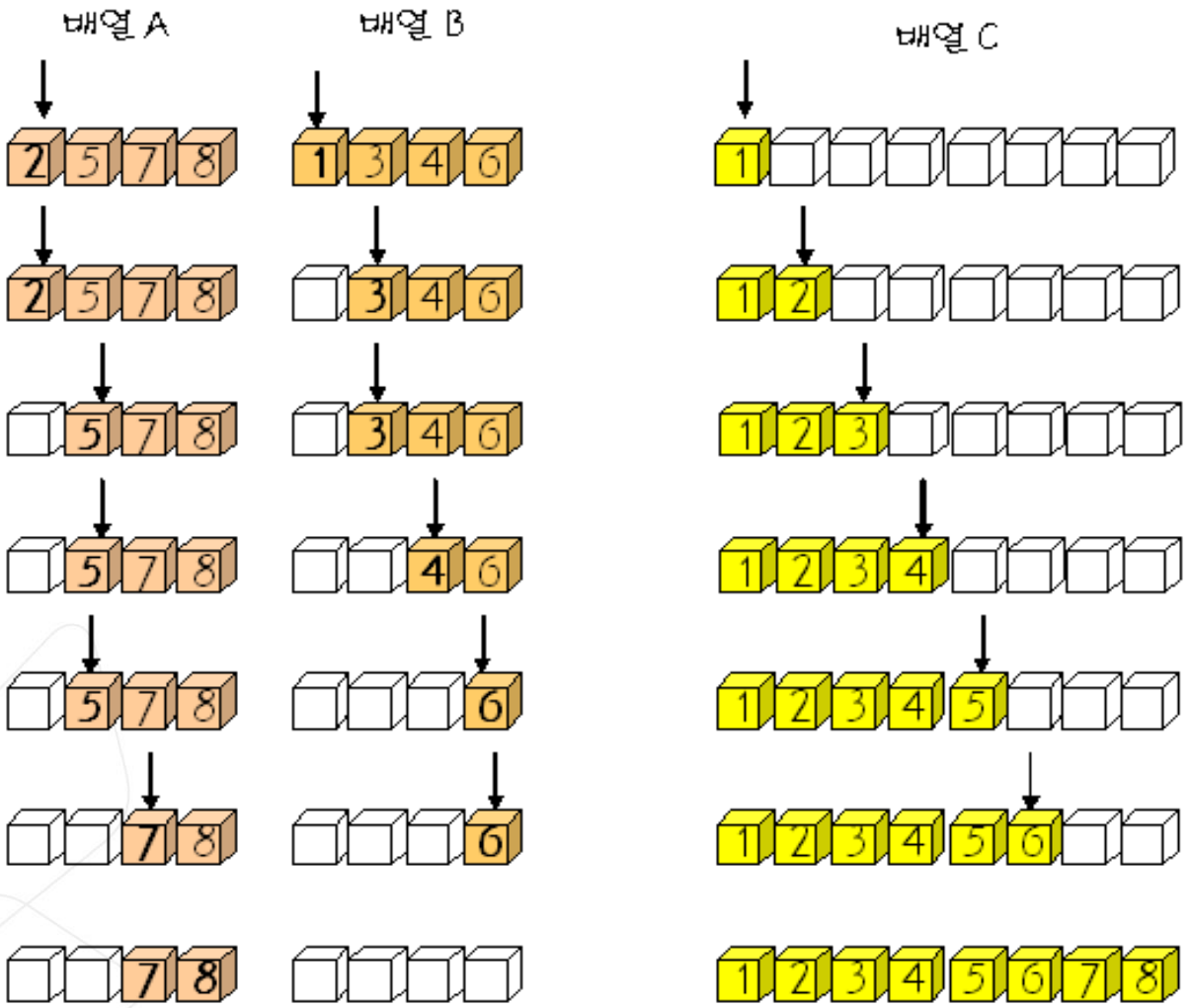
Quick Sort

- 분할(partition)
  - *O*(n)



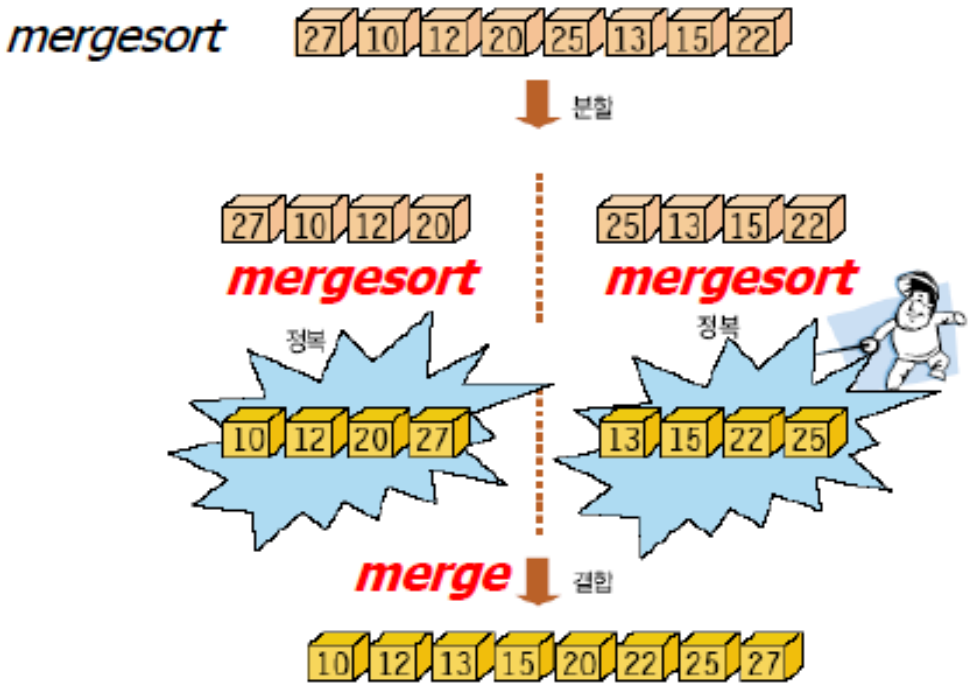
Merge Sort

- 합병(merge)
  - *O*(n)



Merge Sort

- Divide – and – conquer(분할정복)
  - 리스트를 분할하여 두번의 merge sort 호출
  - 정렬된 리스트 두개를 합병(merge)



실습 13-1. Sorting

- Sorting함수의 구현
  - Insertion Sort
  - Quick Sort
  - Merge Sort

실습 13-1. 자료구조 및 함수

- void insertion_sort(int list[], int n);
  - insertion sort. 중간 단계와 키 비교 수 출력
- void quick_sort(int list[], int left, int right);
  - quick sort. 중간 단계와 키 비교 수 출력
- void merge_sort(int list[], int left, int right);
  - merge sort. 중간 단계와 키 비교 수 출력
- void merge(int list[], int left, int mid, int right);
  - 정렬되어 있는 list[left…mid]와 list[mid+1…right]를 합병

실습 13-1. 자료구조 및 함수

- void copy_list(int original[], int list[], int n);
  - original을 list에 복사
- void print_list(int list[], int left, int right);
  - list를 left에서 right까지만 출력

실습 13-1. 실행 예

```
----- insertion sort -----
 25   5  37   1  61  11  59  15  48  19
  5  25  37   1  61  11  59  15  48  19
  5  25  37   1  61  11  59  15  48  19
  1   5  25  37  61  11  59  15  48  19
  1   5  25  37  61  11  59  15  48  19
  1   5  11  25  37  61  59  15  48  19
  1   5  11  25  37  59  61  15  48  19
  1   5  11  15  25  37  59  61  48  19
  1   5  11  15  25  37  48  59  61  19
  1   5  11  15  19  25  37  48  59  61

  1   5  11  15  19  25  37  48  59  61

Total number of comparison = 19
```

실습 13-1. 실행 예

```
----- quick sort -----
 25   5  37   1  61  11  59  15  48  19
 11   5  19   1  15  25  59  61  48  37
  1   5  11  19  15
  1   5
          15  19
                      48  37  59  61
                      37  48

  1   5  11  15  19  25  37  48  59  61

Total number of comparison = 11
```

실습 13-1. 실행 예

```
----- merge sort -----
 25   5  37   1  61  11  59  15  48  19
  5  25
  5  25  37
             1  61
  1   5  25  37  61
                 11  59
                 11  15  59
                             19  48
                 11  15  19  48  59
  1   5  11  15  19  25  37  48  59  61

  1   5  11  15  19  25  37  48  59  61

Total number of comparison = 25
```

실습 13-1. sorting.h

```
#define MAX_SIZE 10
#define boolean unsigned char
#define true 1
#define false 0


//정렬할 테스트 데이터
int original[] = {25, 5, 37, 1, 61, 11, 59, 13, 48, 19};
//키값 비교횟수 카운트를 위한 변수
int num_compare;
```

실습 13-1. sorting.h

```
//insertion sort
void inertion_sort(int list[], int n);
//quick sort
void quick_sort(int list[], int left, int right);
//merge sort
void merge_sosrt(int list[], int left, int right);
//merge
void merge(int list[], int left, int mid, int right);

//list를 복사
void copy_list(int original[], int list[], int n);
//list를 left에서 right까지만 출력
void print_list(int list[], int left, int right);
```

실습 13-1. copy_list, print_list 함수

```
void copy_list(int original[], int list[], int n)
{
        int i;
        for(i=0;i<n;i++)
                list[i] = original[i];
}
void print_list(int list[], int left, int right)
{
        int i;
        for(i=0;i<left;i++)
                printf("");
        for(i=0;i<=right;i++)
                printf("%4d", list[i]);
        printf("₩n");
}
```