13주차 1차시 정렬 1

【학습목표】

1. 정렬을 예를 들어 설명할 수 있다.
2. 선택, 버블, 퀵 정렬을 구분할 수 있다.

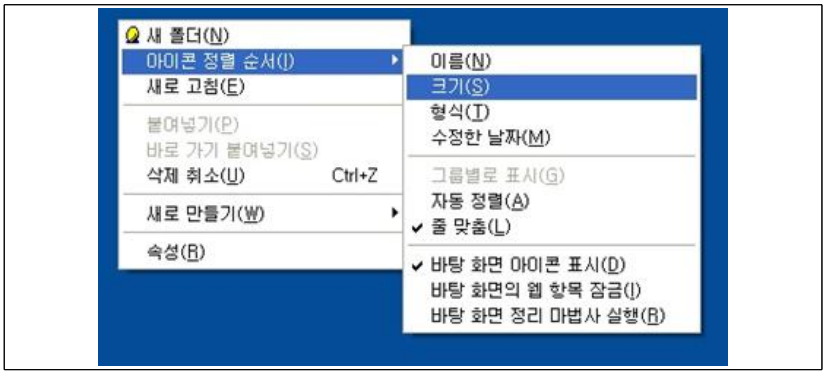
학습내용1 : 정렬과 정렬종류

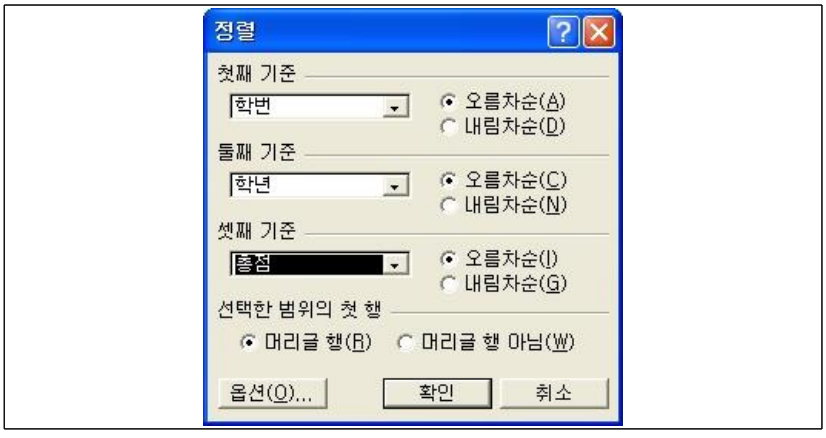
1. 정렬

* 2개 이상의 자료를 작은 것부터 큰 것 순서의 오름차순(ascending)이나 큰 것부터 작은 것 순서의 내림차순(descending)으로 재배열하는 것

* 키 : 자료를 정렬하는데 사용하는 기준이 되는 특정 값

* 정렬의 예





2. 정렬방법의 분류

* 실행 방법에 따른 분류
- 비교식 정렬
    - 비교하고자 하는 각 키 값들을 한 번에 두 개씩 비교하여 교환하는 방식
- 분산식 정렬
    - 키 값을 기준으로 하여 자료를 여려 개의 부분 집합으로 분해하고 각 부분집합을 정렬함으로써 전체를 정렬하는 방식

* 정렬 장소에 따른 분류
- 내부 정렬(internal sort)
    - 정렬할 자료를 메인 메모리에 올려서 정렬하는 방식
    - 정렬 속도가 빠르지만 정렬할 수 있는 자료의 양이 메인 메모리의 용량에 따라 제한됨

- 내부 정렬 방식
    - 교환 방식 : 키를 비교하고 교환하여 정렬하는 방식 (선택 정렬, 버블 정렬, 퀵 정렬)
    - 삽입 방식 : 키를 비교하고 삽입하여 정렬하는 방식(삽입 정렬, 쉘 정렬)
    - 병합 방식 : 키를 비교하고 병합하여 정렬하는 방식 (2-way병합, n-way병합)
    - 분배 방식 : 키를 구성하는 값을 여러 개의 부분집합에 분배하여 정렬하는 방식 (기수 정렬)
    - 선택 방식 : 이진 트리를 사용하여 정렬하는 방식 (히프 정렬, 트리 정렬)

- 외부 정렬(external sort)
    - 정렬할 자료를 보조 기억장치에서 정렬하는 방식
    - 대용량의 보조 기억 장치를 사용하기 때문에 내부 정렬보다 속도는 떨어지지만 내부 정렬로 처리할 수 없는 대용량의 자료에 대한 정렬 가능

- 외부 정렬 방식
    - 병합 방식 : 파일을 부분 파일로 분리하여 각각을 내부 정렬 방법으로 정렬하여 병합하는 정렬 방식 (2-way 병합, n-way 병합)

* 정렬 방식의 선택 기준

- 시스템 특성, 정렬할 자료의 양, 자료들의 정렬 전 상태, 정렬에 사용하는 기억 공간과 실행 시간 등의 조건에 따라 결정해야 함
- 정렬의 효율성을 비교하는 기준은 정렬에 필요한 비교 횟수와 이동 횟수

학습내용2 : 선택 정렬과 버블 정렬

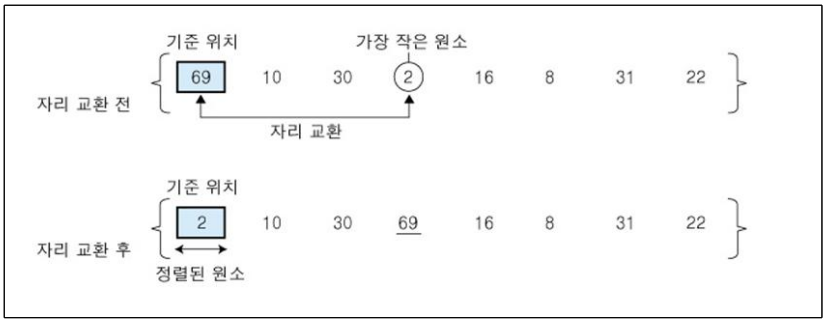
1. 선택 정렬

* 전체 원소들 중에서 기준 위치에 맞는 원소를 선택하여 자리를 교환하는 방식으로 정렬

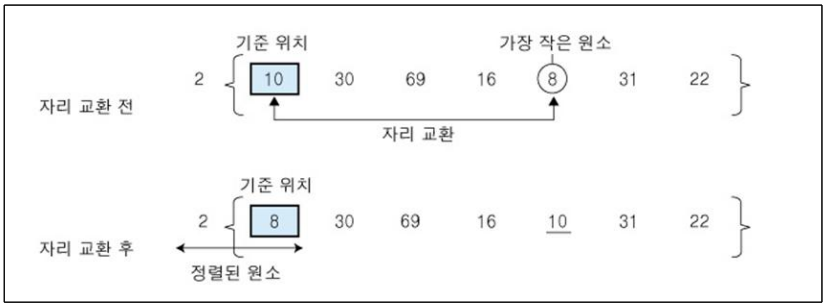
* 수행 방법
- 전체 원소 중에서 가장 작은 원소를 찾아 선택하여 첫 번째 원소와 자리를 교환한다.
- 그 다음 두 번째로 작은 원소를 찾아 선택하여 두 번째 원소와 자리를 교환한다.
- 그 다음에는 세 번째로 작은 원소를 찾아 선택하여 세 번째 원소와 자리를 교환한다.
- 이 과정을 반복하면서 정렬을 완성한다.

2. 선택 정렬 수행 과정

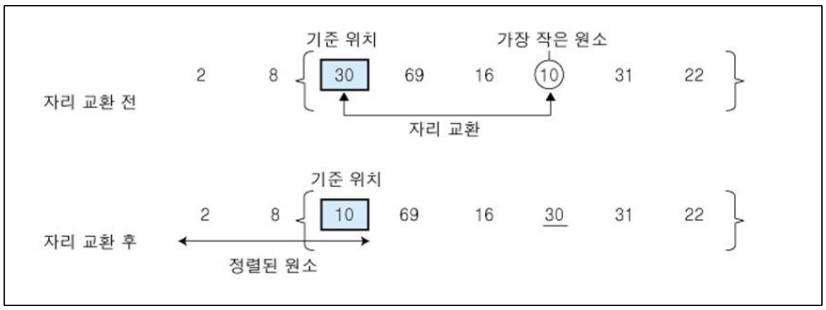
* 정렬되지 않는 {69, 10, 30, 2, 16, 8, 31, 22 }의 자료들을 선택 정렬 방법으로 정렬하는 과정
① 첫 번째 자리를 기준 위치로 정하고, 전체 원소 중에서 가장 작은 원소 2를 선택하여 기준 위치에 있는 원소 69와 자리 교환



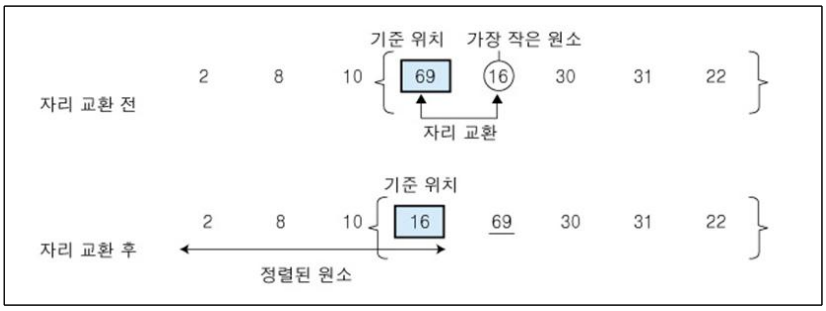
② 두 번째 자리를 기준 위치로 정하고 나머지 원소 중에서 가장 작은 원소 8을 선택하여 기준 위치에 있는 원소 10과 자리 교환



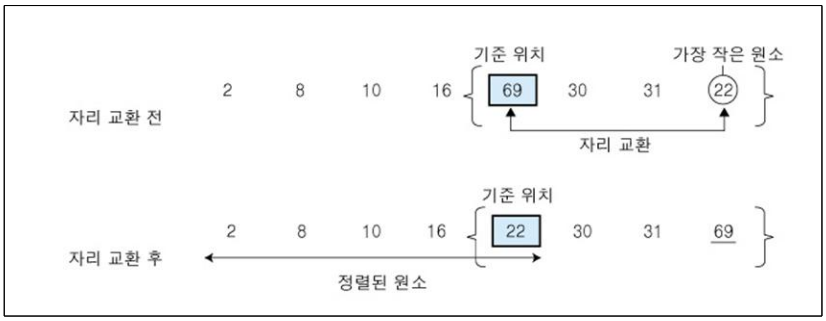
③ 세 번째 자리를 기준 위치로 정하고, 나머지 원소 중에서 가장 작은 원소 10을 선택하여 기준 위치에 있는 원소 30과 자리 교환



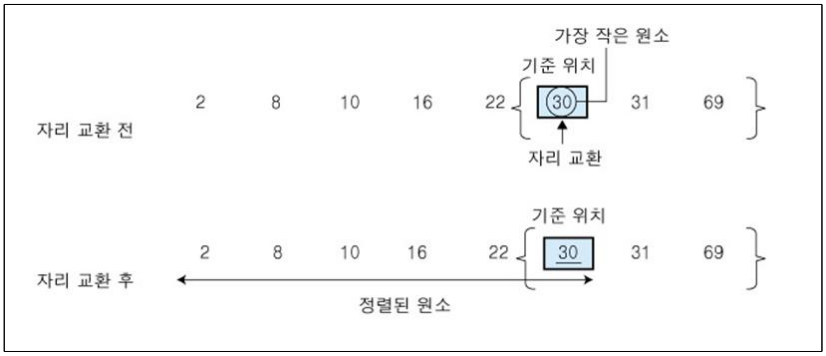
④ 네 번째 자리를 기준 위치로 정하고, 나머지 원소 중에서 가장 작은 원소 16을 선택하여 기준 위치에 있는 원소 69와 자리 교환



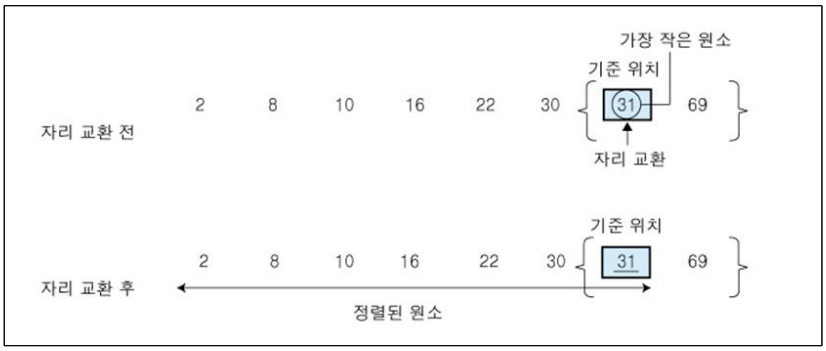
⑤ 다섯 번째 자리를 기준 위치로 정하고, 나머지 원소 중에서 가장 작은 원소 22를 선택하여 기준 위치에 있는 원소 69와 자리 교환



⑥ 여섯 번째 자리를 기준 위치로 정하고, 나머지 원소 중에서 가장 작은 원소 30을 선택하여 기준 위치에 있는 원소 30과 자리 교환 (제자리)



⑦ 일곱 번째 자리를 기준 위치로 정하고, 나머지 원소 중에서 가장 작은 원소 31을 선택하여 기준 위치에 있는 원소 31과 자리 교환. (제자리)



- 마지막에 남은 원소 69는 전체 원소 중에서 가장 큰 원소로서 이미 마지막 자리에 정렬된 상태이므로 실행을 종료하고 선택 정렬이 완성된다.

3. 선택 정렬 알고리즘

알고리즘 10-1 선택 정렬 알고리즘

```
selectionSort(a[],n)
    for (i←1; i<n; i←i+1) {
        a[i], …,a[n-1] 중에서 가장 작은 원소 a[k]를 선택하여,
        a[i]와 교환한다.
    }
end selectionSort()
```

* 메모리 사용공간
- n개의 원소에 대하여 n개의 메모리 사용

* 비교횟수
- 1단계 : 첫 번째 원소를 기준으로 n개의 원소 비교
- 2단계 : 두 번째 원소를 기준으로 마지막 원소까지 n-1개의 원소 비교
- 3단계 : 세 번째 원소를 기준으로 마지막 원소까지 n-2개의 원소 비교
- i 단계 : i 번째 원소를 기준으로 n-i개의 원소 비교

$$\text{전체 비교횟수} = \sum_{i=1}^{n-1} n-i = \frac{n(n-1)}{2}$$

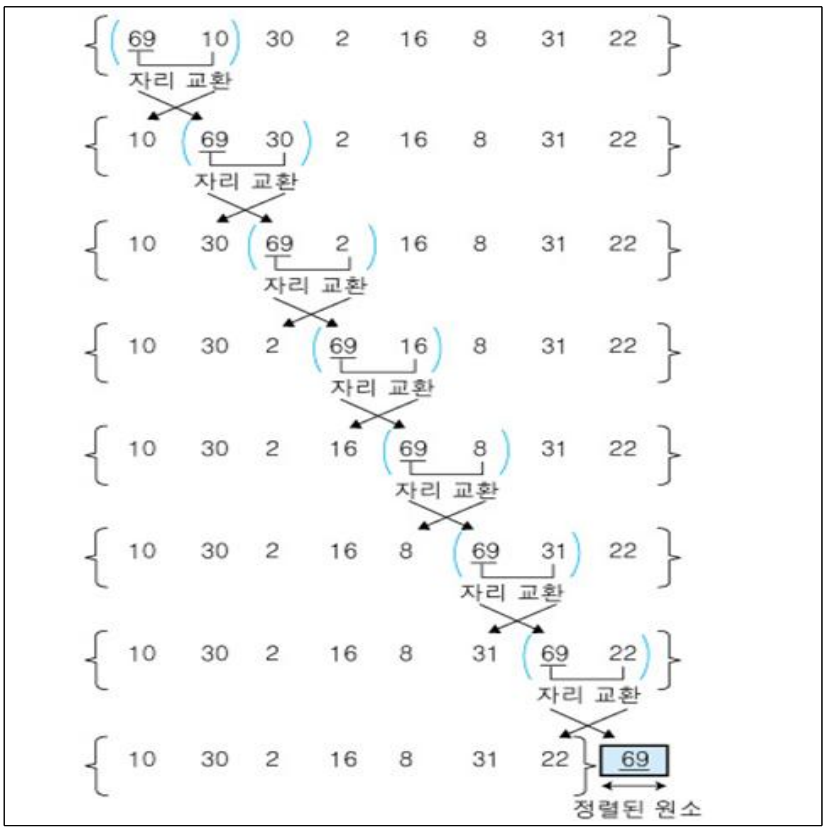
* 어떤 경우에서나 비교횟수가 같으므로 시간 복잡도는 $O(n^2)$이 된다.

4. 버블 정렬

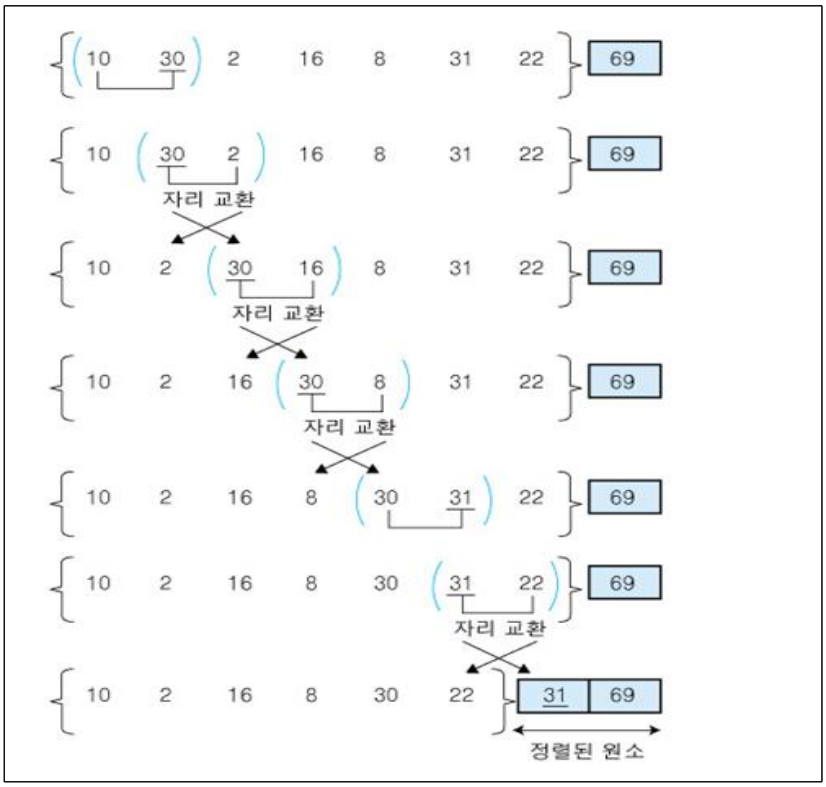
* 인접한 두 개의 원소를 비교하여 자리를 교환하는 방식
- 첫 번째 원소부터 마지막 원소까지 반복하여 한 단계가 끝나면 가장 큰 원소가 마지막 자리로 정렬
- 첫 번째 원소부터 인접한 원소끼리 계속 자리를 교환하면서 맨 마지막 자리로 이동하는 모습이 물 속에서 물 위로 올라오는 물방울 모양과 같다고 하여 버블(bubble) 정렬이라 함.

5. 버블 정렬 수행 과정

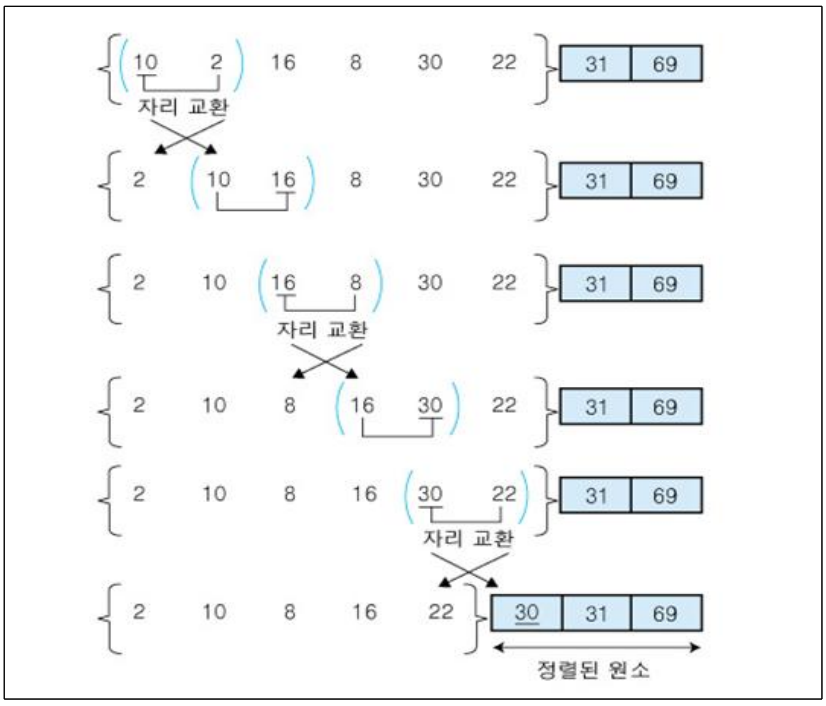
* 정렬되지 않은 {69, 10, 30, 2, 16, 8, 31, 22 }의 자료들을 버블 정렬 방법으로 정렬하는 과정
① 인접한 두 원소를 비교하여 자리를 교환하는 작업을 첫 번째 원소부터 마지막 원소까지 차례로 반복하여 가장 큰 원소 69를 마지막 자리로 정렬



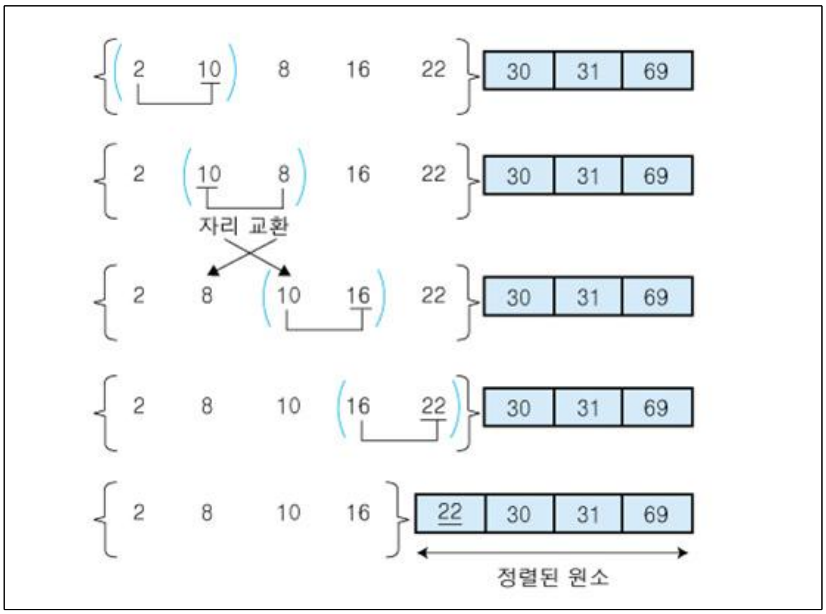
② 버블 정렬을 수행하여 나머지 원소 중에서 가장 큰 원소 31을 끝에서 두 번째 자리로 정렬



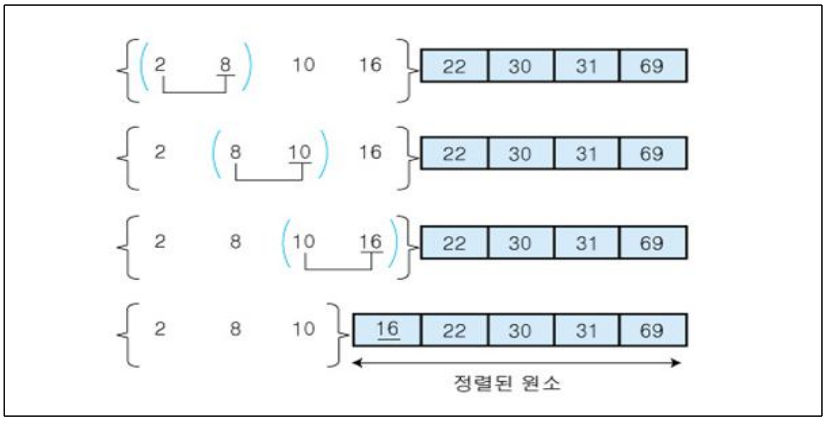
③ 버블 정렬을 수행하여 나머지 원소 중에서 가장 큰 원소 30을 끝에서 세 번째 자리로 정렬.



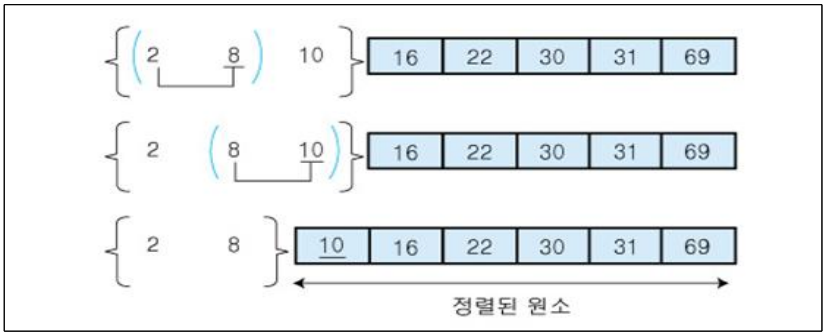
④ 버블 정렬을 수행하여 나머지 원소 중에서 가장 큰 원소 22를 끝에서 네 번째 자리로 정렬.



⑤ 버블 정렬을 수행하여 나머지 원소 중에서 가장 큰 원소 16을 끝에서 다섯 번째 자리로 정렬.

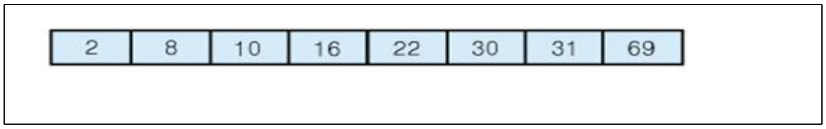


⑥ 버블 정렬을 수행하여 나머지 원소 중에서 가장 큰 원소 10을 끝에서 여섯 번째 자리로 정렬



⑦ 버블 정렬을 수행하여 나머지 원소 중에서 가장 큰 원소 8을 끝에서 일곱 번째 자리로 정렬

* 마지막에 남은 첫 번째 원소는 전체 원소 중에서 가장 작은 원소로 이미 정렬된 상태이므로 실행을 종료하고 버블 정렬이 완성된다



6. 버블 정렬 알고리즘

알고리즘 10-2 버블 정렬 알고리즘

```
bubbleSort(a[],n)
    for (i←n-1; i≥0; i←i-1) {
        for (j←0; j<i; j←j+1) {
            if (a[j]>a[j+1]) then {
                temp ← a[j];
                a[j] ← a[j+1];
                a[j+1] ← temp;
            }
        }
    }
end bubbleSort()
```

* 메모리 사용공간
- n개의 원소에 대하여 n개의 메모리 사용

* 연산 시간
- 최선의 경우 : 자료가 이미 정렬되어있는 경우
    - 비교횟수 : i번째 원소를 (n-1)번 비교하므로 n(n-1)/2번
    - 자리교환횟수 : 자리교환이 발생하지 않음

- 최악의 경우 : 자료가 역순으로 정렬되어 있는 경우
    - 비교횟수 : i번째 원소를 (n-1)번 비교하므로 n(n-1)/2번
    - 자리교환횟수 : i번째 원소를 (n-1)번 비교하므로 n(n-1)/2번

- 평균 시간 복잡도는 $O(n^2)$이 된다.

학습내용3 : 퀵 정렬

1. 개요

* 정렬할 전체 원소에 대해서 정렬을 수행하지 않고 기준 값을 중심으로 왼쪽 부분 집합과 오른쪽 부분 집합으로 분할하여 정렬하는 방법
- 왼쪽 부분 집합에는 기준 값보다 작은 원소들을 이동시키고, 오른쪽 부분 집합에는 기준 값보다 큰 원소들을 이동시킨다.
- 기준 값 : 피봇(pivot)
    - 일반적으로 전체 원소 중에서 가운데에 위치한 원소를 선택

* 퀵 정렬은 다음의 두 가지 기본 작업을 반복 수행하여 완성한다.
- 분할(divide) : 정렬할 자료들을 기준값을 중심으로 2개의 부분 집합으로 분할하기
- 정복(conquer) : 부분 집합의 원소들 중에서 기준값보다 작은 원소들은 왼쪽 부분 집합으로, 기준값보다 큰 원소들은 오른쪽 부분집합으로 정렬하기
    - 부분 집합의 크기가 1 이하로 충분히 작지 않으면 순환호출을 이용하여 다시 분할

* 퀵 정렬 수행 방법
- 부분 집합으로 분할하기 위해서 L과 R을 사용
① 왼쪽 끝에서 오른쪽으로 움직이면서 크기를 비교하여 피봇보다 크거나 같은 원소를 찾아 L로 표시
② 오른쪽 끝에서 왼쪽으로 움직이면서 피봇 보다 작은 원소를 찾아 R로 표시
③ L이 가리키는 원소와 R이 가리키는 원소를 서로 교환한다

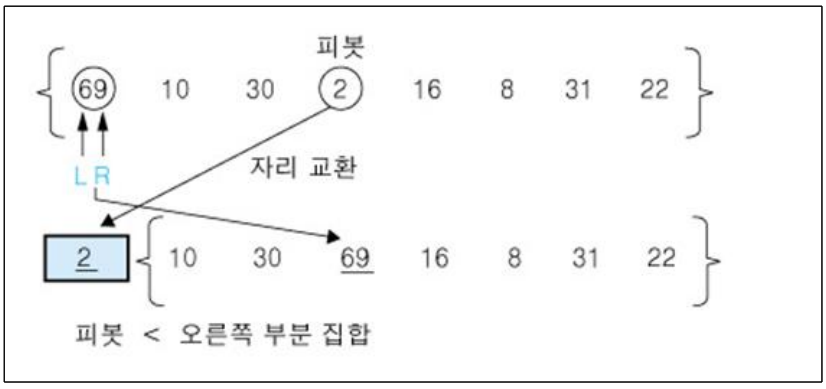
- L와 R이 만나게 되면 피봇과 R의 원소를 서로 교환하고, 교환한 위치를 피봇의 위치로 확정한다.
- 피봇의 확정된 위치를 기준으로 만들어진 새로운 왼쪽 부분 집합과 오른쪽 부분 집합에 대해서 퀵 정렬을 순환적으로 반복 수행하는데 모든 부분 집합의 크기가 1 이하가 되면 퀵 정렬을 종료한다.

2. 퀵 정렬 수행 과정

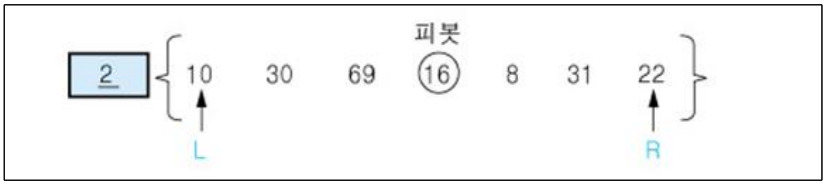
* 정렬되지 않은 {69, 10, 30, 2, 16, 8, 31, 22 }의 자료들을 퀵 정렬 방법으로 정렬하는 과정
- 원소의 개수가 8개이므로 네 번째 자리에 있는 원소 2를 첫 번째 피봇으로 선택하고 퀵 정렬 시작

① 원소 2를 피봇으로 선택하고 퀵 정렬 시작

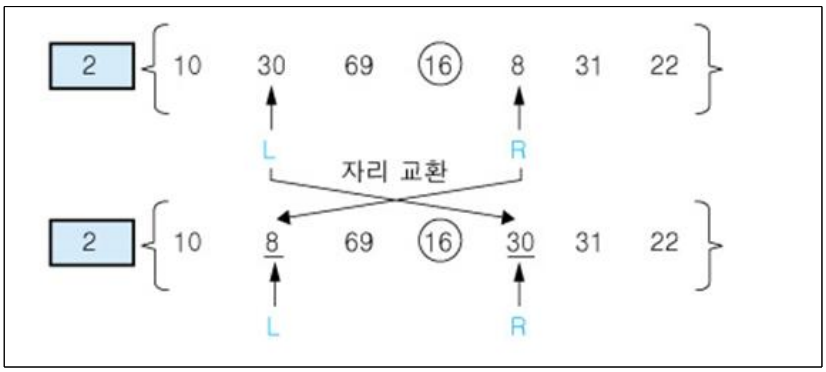
- L이 오른쪽으로 이동하면서 피봇 보다 크거나 같은 원소를 찾고, R은 왼쪽으로 이동하면서 피봇 보다 작은 원소를 찾는다
- L은 원소 69를 찾았지만, R은 피봇 보다 작은 원소를 찾지 못한 채로 원소 69에서 L과 만나게 된다.
- L과 R이 만났으므로, 원소 69를 피봇과 교환하여 피봇 원소 2의 위치를 확정한다.



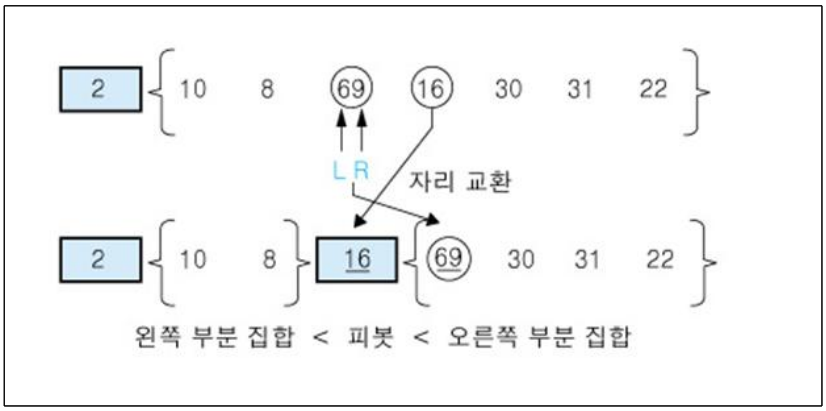
② 피봇 2의 왼쪽 부분 집합은 공집합이므로 퀵 정렬을 수행하지 않고, 오른쪽 부분 집합에 대해서 퀵 정렬 수행, 오른쪽 부분 집합의 원소가 7개 이므로 가운데 있는 원소 16을 피봇으로 선택.



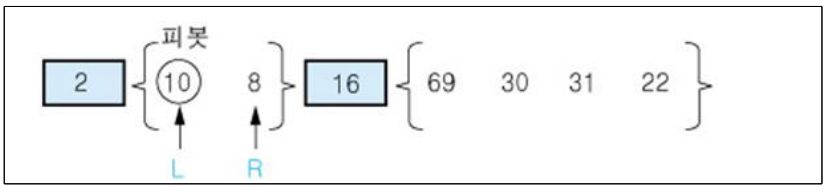
- L이 찾은 30과 R이 찾은 8을 서로 교환한다



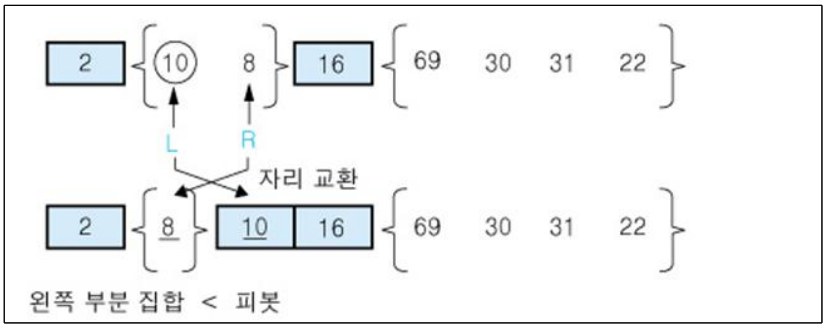
- 현재 위치에서 L과 R의 작업을 반복한다
- L은 원소 69를 찾았지만, R은 피봇 보다 작은 원소를 찾지 못한 채로 원소 69에서 L과 만나게 된다. L과 R이 만났으므로, 원소 69를 피봇과 교환하여 피봇 원소 16의 위치를 확정한다.



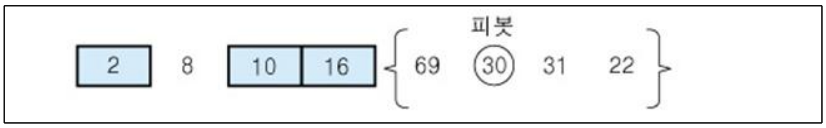
③ 피봇 16의 왼쪽 부분 집합에서 원소 10을 피봇으로 선택하여 퀵 정렬 수행



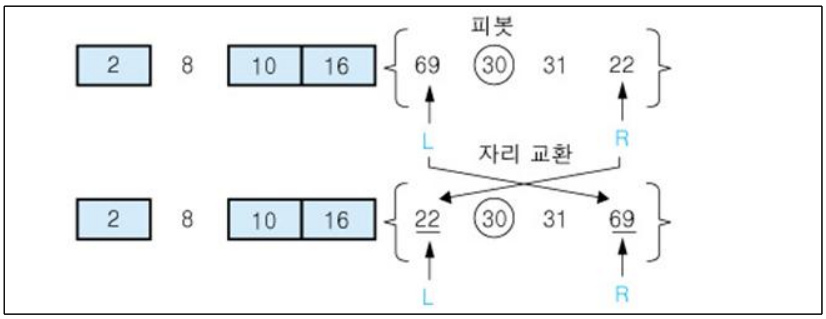
- L의 원소 10과 R의 원소 8을 교환하는데, L의 원소가 피봇이므로 피봇원소에 대한 자리교환이 발생한 것이므로 교환한 자리를 피봇 원소 10의 위치로 확정한다



④ 피봇 10의 확정된 위치에서의 왼쪽 부분 집합은 원소가 한 개이므로 퀵 정렬을 수행하지 않고, 오른쪽 부분 집합은 공집합이므로 역시 퀵 정렬을 수행하지 않는다.
이제 1단계의 피봇이었던 원소 16에 대한 오른쪽 부분 집합에 대해 퀵 정렬을 수행한다. 오른쪽 부분 집합의 원소가 4개이므로 두 번째 원소 30을 피봇으로 선택한다.

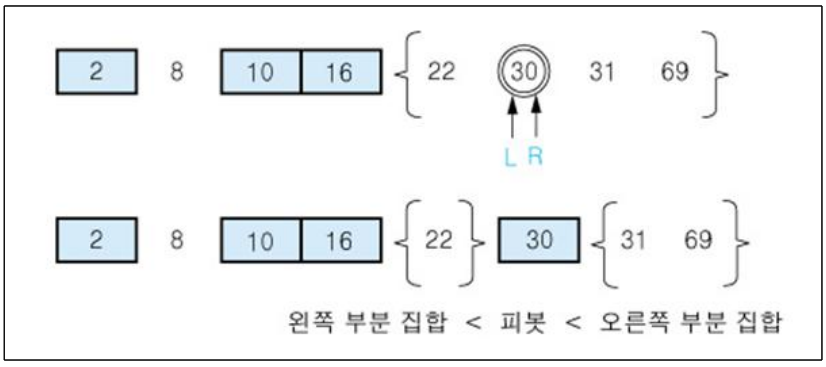


- L이 찾은 69와 R이 찾은 22를 서로 교환한다

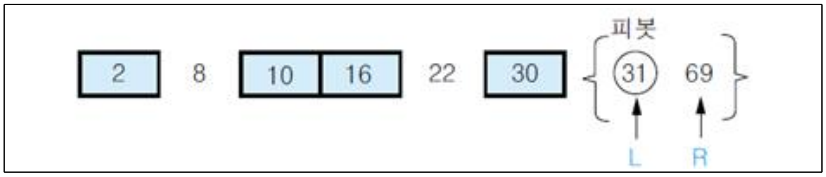


- 현재 위치에서 L과 R의 작업을 반복한다.
- L은 오른쪽으로 이동하면서 피봇 보다 크거나 같은 원소인 30을 찾고, R은 왼쪽으로 이동하면서 피봇 보다 작은 원소를 찾다가 못 찾고 원소 30에서 L과 만난다.

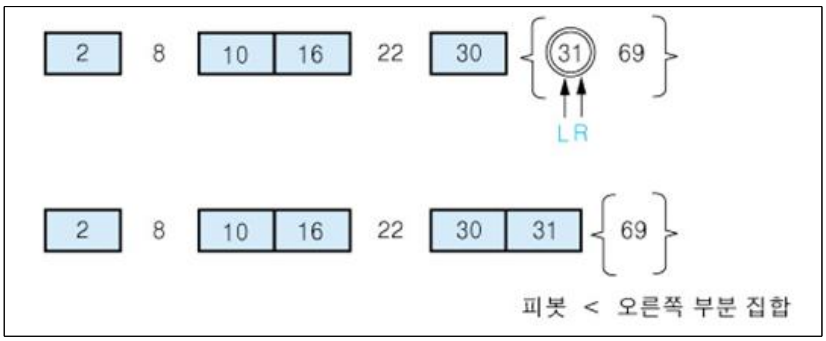
L과 R이 만났으므로 피봇과 교환한다. 이 경우는 R의 원소가 피봇이므로 피봇에 대한 자리교환이 발생한 것이므로 교환한 자리를 피봇의 자리로 확정한다



⑤ 피봇 30의 확정된 위치에서의 왼쪽 부분 집합의 원소가 한 개 이므로 퀵 정렬을 수행하지 않고, 오른쪽 부분 집합에 대해서 퀵 정렬 수행. 오른쪽 부분 집합의 원소 2개 중에서 원소 31을 피봇으로 선택한다.



- L은 오른쪽으로 이동하면서 원소 31을 찾고, R은 왼쪽으로 이동하면서 피봇 보다 작은 원소를 찾다가 못 찾은 채로 원소 31에서 L과 만난다. L과 R이 만났으므로 피봇과 교환하는데 R의 원소가 피봇이므로 결국 제자리가 확정된다



- 피봇 31의 오른쪽 부분 집합의 원소가 한 개이므로 퀵 정렬을 더 이상 수행하지 않고 전체 퀵 정렬이 모두 완성된다.

3. 퀵 정렬 알고리즘

알고리즘 10-3 퀵 정렬 알고리즘

```
quickSort(a[], begin, end)
    if (m < n) then {
        p ← partition(a, begin, end);
        quickSort(a[], begin, p-1);
        quickSort(a[], p+1, end);
    }
end quickSort()
```

알고리즘 10-4 파티션 분할 알고리즘

```
partiton(a[], begin, end)
    pivot ← (begin + end)/2;
    L ← begin;
    R ← end;
    while(L<R) do {
        while(a[L]<a[pivot] and L<R) do L++;
        while(a[R]≥a[pivot] and L<R) do R--;
        if(L<R) then { // L의 원소와 R의 원소 교환
            temp ← a[L];
            a[L] ← a[R];
            a[R] ← temp;
        }
    }
    temp ← a[pivot]; // R의 원소와 피봇 원소 교환
    a[pivot] ← a[R];
    a[R] ← temp;
    return R;
end partition()
```

* 메모리 사용공간
- n개의 원소에 대한 n개의 메모리 사용

* 연산시간
- 최선의 경우 : 피봇에 의해서 원소들이 왼쪽 부분 집합과 오른쪽 부분 집합으로 정확히 n/2개씩 이등분이 되는 경우가 반복되어 수행 단계 수가 최소가 되는 경우
- 최악의 경우 : 피봇에 의해 원소들을 분할하였을 때 1개와 n-1개로 한쪽으로 치우쳐 분할되는 경우가 반복되어 수행단계 수가 최대가 되는 경우
- 평균 시간 복잡도 : : $O(n \log_2 n)$
    - 같은 시간 복잡도를 가지는 다른 정렬 방법에 비해서 자리 교환 횟수를 줄임으로써 더 빨리 실행되어 실행 시간 성능이 좋은 정렬 방법임

【학습정리】

1. 정렬이란 순서없이 배열되어 있는 자료들을 작은 것부터 큰 것 순서의 오름차순이나 큰 것부터 작은 것 순서의 내림차순으로 재배열하는 것이다.

2. 정렬 방법은 수행되는 위치에 따라 컴퓨터 메모리 내부에서 수행하는 내부 정렬과 메모리의 외부인 보조 기억 장치에서 수행하는 외부 정렬로 분류할 수 있고, 수행 방버에 따라 비교식 정렬과 분산식 정렬로 구분할 수 있다.