1 선택정렬 Selection Sort

선택정렬 알고리즘의 개념 요약

- 제자리 정렬(in-place sorting) 알고리즘의 한 종류
  - 입력 배열(정렬되지 않은 값들) 이외에 다른 추가 메모리를 요구하지 않는 정렬

- 해당 순서에 원소를 넣을 위치는 이미 정해져 있고, 어떤 원소를 넣을지 선택하는 알고리즘
  - 첫번째 순서에는 첫번째 위치에 가장 최소값을 입력
  - 두번째 순서에는 두번째 위치에 남은 값 중에서 최소값을 입력
  - ...

- 과정 설명
  1. 주어진 배열 중에서 최소값을 찾는다
  2. 그 값을 맨 앞에 위치한 값과 교체한다
  3. 맨 처음 위치를 뺀 나머지 리스트를 같은 방법으로 교체
  4. 하나의 원소가 남을 때까지 위의 1 ~ 3과정을 반복한다

1 선택정렬 Selection Sort

1 선택정렬 Selection Sort

선택정렬 알고리즘의 특징

- 장점
  - 자료 이동 횟수가 미리 결정된다.
- 단점
  - 안정성을 만족하지 않는다
  - 즉, 값이 같은 데이터가 있을 경우에 상대적인 위치가 변경될 수 있다.

선택정렬의 시간복잡도

- 시간복잡도를 계산한다면
  - 비교 횟수
    - 정렬 횟수 : n-1
    - 최소값 찾기 : n-1, n-2, …., 2, 1번
  - 교환횟수
    - 정렬 실행 횟수와 동일 즉, 상수 시간 작업
    - 한번 교환하기 위해 3번의 이동(SWAP 작업)이 필요하므로 3(n-1)
  - $T(n) = (n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2} = O(n^2)$

2 삽입정렬 Insertion Sort

삽입정렬 알고리즘의 개념 요약

- 손안의 카드를 정렬하는 방법과 유사하다
  - 새로운 카드를 기존의 정렬된 카드 사이의 올바른 자리를 찾아 삽입한다.
  - 새로 삽입될 카드의 수만큼 반복하게 되면 전체 카드가 정렬된다.

- 자료 배열의 모든 요소를 **앞에서부터 차례대로 이미 정렬된 배열 부분과 비교**하여, 자신의 위치를 찾아 삽입함으로써 정렬을 완성하는 알고리즘

- 매 순서마다 해당 원소를 삽입할 수 있는 위치를 찾아 해당 위치에 넣는다.

삽입정렬 알고리즘의 구체적인 개념

- 삽입 정렬은 두번째 자료부터 시작하여 그 앞(왼쪽)의 자료들과 비교하여 삽입할 위치를 지정한 후 자료를 뒤로 옮기고 지정한 자리에 자료를 삽입하여 정렬하는 알고리즘이다.

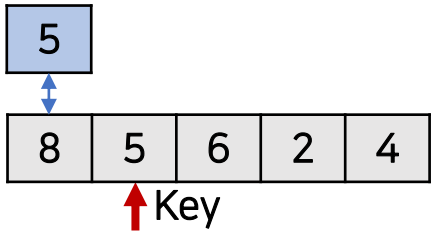
- 즉, 두번째 자료는 첫번째 자료, 세번째 자료는 두번째 자료, 네번째 자료는 세번째와 비교한 후 자료가 삽입될 위치를 찾는다. 자료가 삽입될 위치를 찾았다면 그 위치에 자료를 삽입하기 위해 자료를 한칸씩 뒤로 이동시킨다.

- **첫 Key**값은 두번째 자료부터 시작한다.

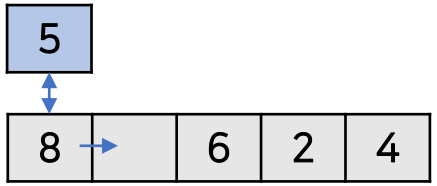
2 삽입정렬 Insertion Sort

초기상태

①



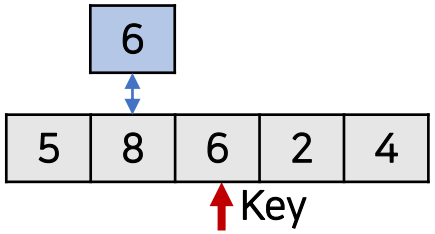
Key: 5 (두번째 값)
첫번째 값 8과 비교



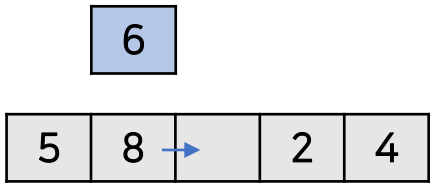
8을 한칸 뒤로 이동
5를 첫번째 자리에 넣는다.

1회차 결과

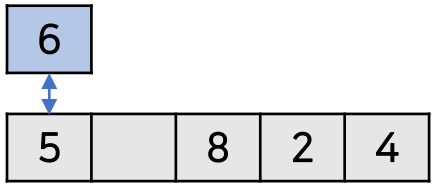
②



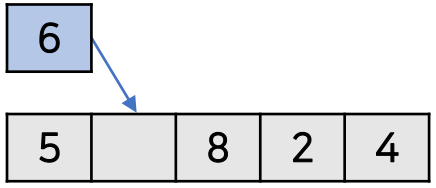
Key: 6 (세번째 값)
두번째 값 8과 비교



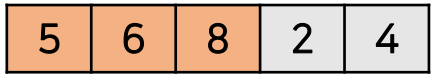
8을 한칸 뒤로 이동
6를 두번째 자리에 넣는다.



첫번째 값 5와 비교

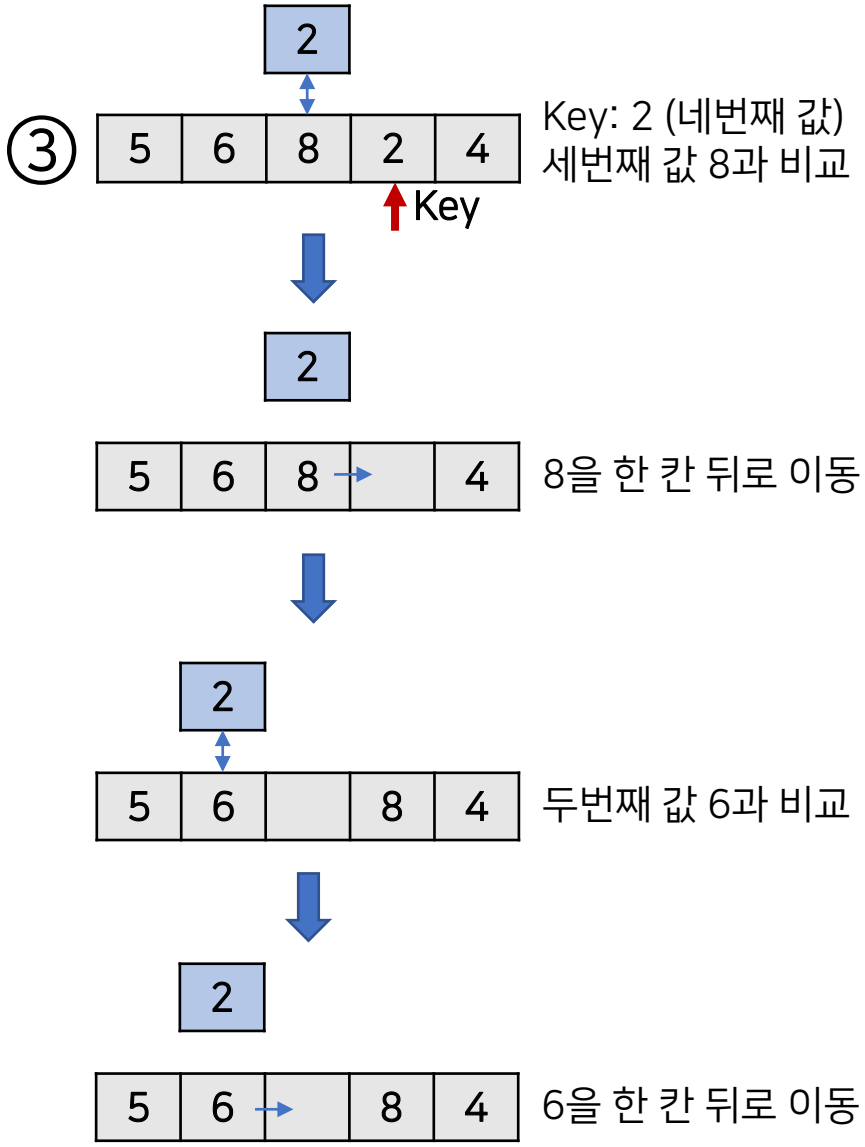


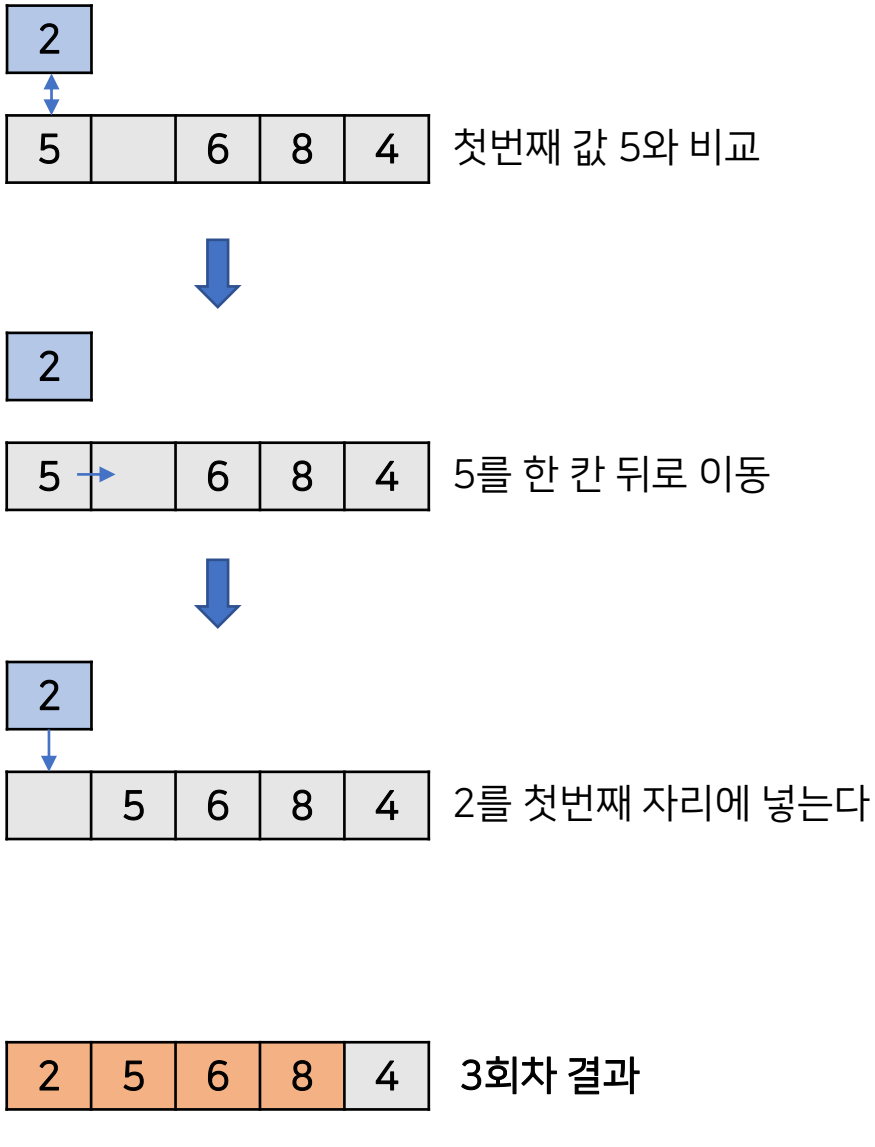
5는 그대로, 6을 두번째 자리에 넣는다.



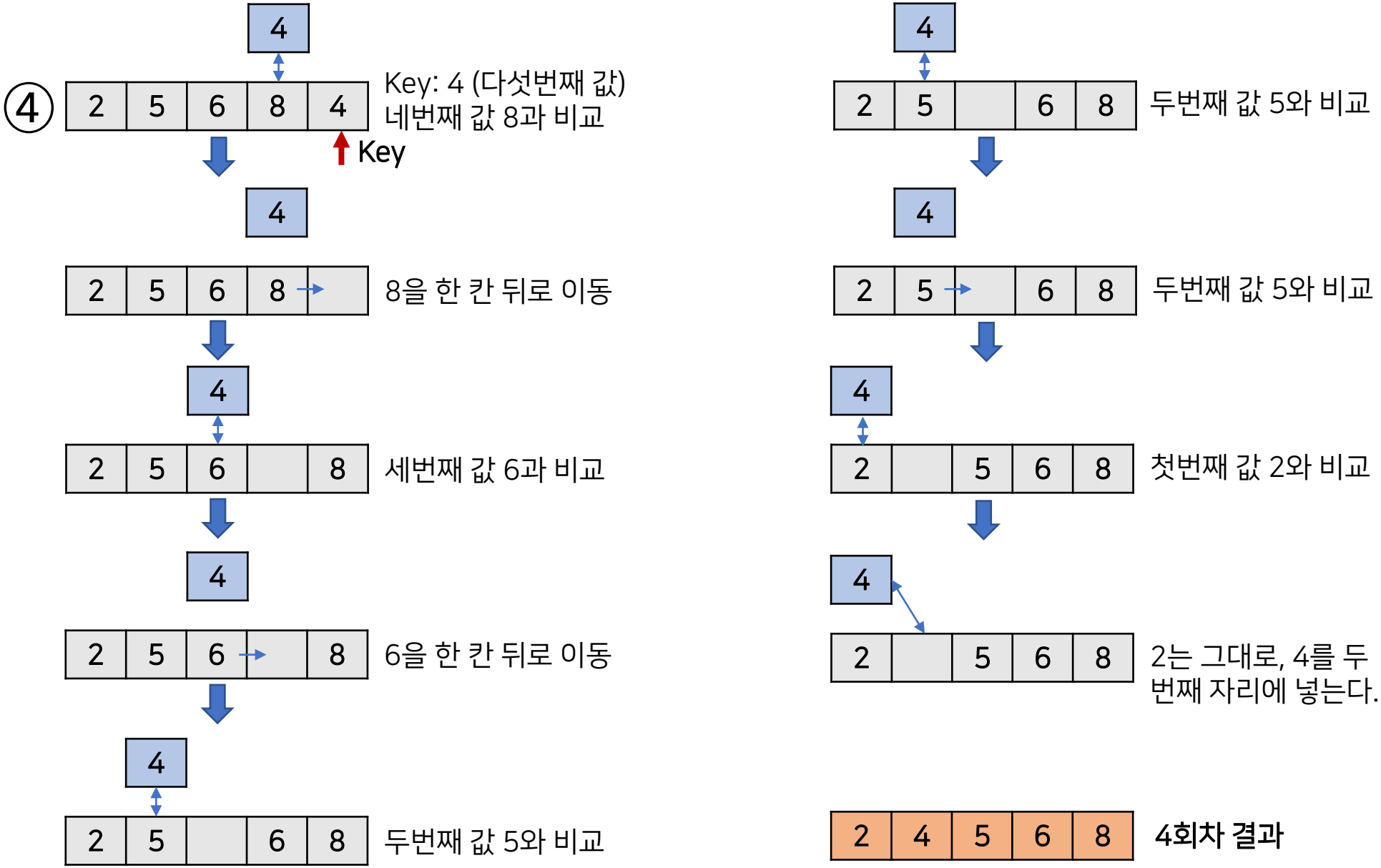
2회차 결과

2 삽입정렬 Insertion Sort





2 삽입정렬 Insertion Sort



2 삽입정렬 Insertion Sort

삽입정렬 알고리즘의 특징

- 장점
  - 안정한 정렬 방법
  - 데이터의 수가 적을 경우 알고리즘 자체가 매우 간단하므로 다른 복잡한 정렬보다 유리할 수 있다.
  - 대부분의 데이터가 이미 정렬되어 있는 경우에 매우 효율적일 수 있다.
- 단점
  - 비교적 많은 데이터들의 이동을 포함한다.
  - 데이터 수가 많고 데이터 크기가 클 경우에 적합하지 않다.

2 삽입정렬 Insertion Sort

삽입정렬의 시간복잡도

- 시간복잡도를 계산한다면
  - 최선의 경우
    - 비교횟수
      - 이동없이 1번의 비교만 이루어진다. $(n-1)$
    - Best $T(n) = O(n)$
  - 최악의 경우
    - 비교횟수
      - $(n-1)+(n-2)+\cdots+2+1=\frac{n(n-1)}{2}=O(n^2)$
    - 교환횟수
      - i+2 만큼 이동 발생
      - $\frac{n(n-1)}{2}+2(n-1)=\frac{n^2+3n-4}{2}=O(n^2)$
    - Worst $T(n) = O(n^2)$

3 버블정렬 Bubble Sort

버블정렬 알고리즘의 개념 요약

- 서로 인전합 두 원소를 검사하여 정렬하는 알고리즘
  - 인접한 2개의 레코드를 비교하여 크기가 순서대로 되어 있지 않으면 서로 교환한다.

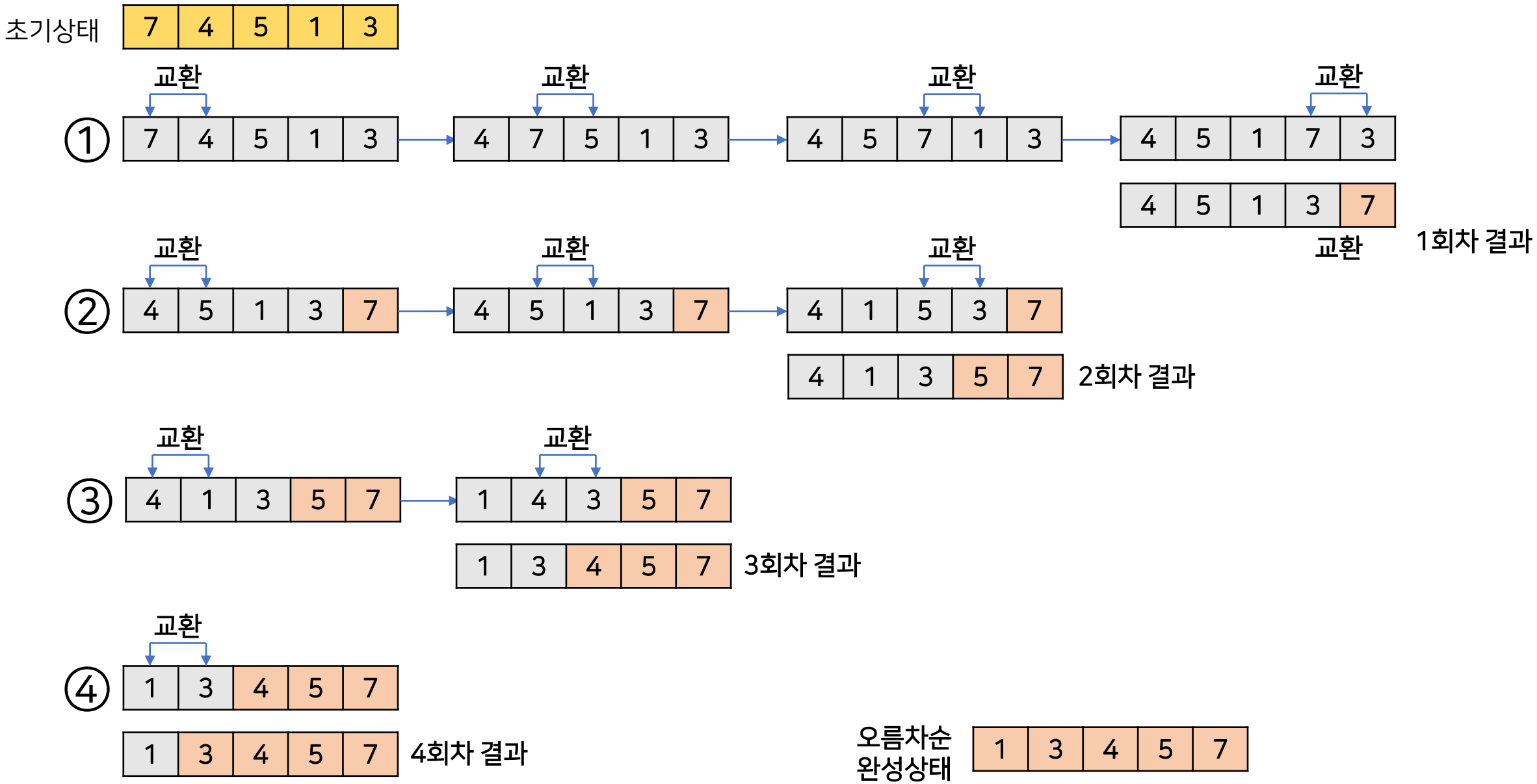
- 선택 정롤과 기본 개념이 유사하다.

버블정렬 알고리즘의 구체적인 개념

- 버블 정렬은 첫번째 자료와 두번째 자료를, 두번째 자료와 세번째 자료를, 세번째 자료와 네번째를, ...이런식으로 (마지막-1)번째 자료와 마지막 자료를 비교하여 교환하면서 자료를 정렬한다.

- 1회전을 수행하고 나면 가장 큰 자료는 맨 뒤로 이동하므로 2회차에서는 맨 끝에 있는 자료는 정렬에서 제외되고, 2회차를 수행하고 나면 끝에서 두번째 자료는 정렬에서 제외된다. 이렇게 정렬을 1회 수행할 때마다 정렬에서 제외되는 데이터가 하나씩 늘어난다.

3 버블정렬 Bubble Sort



3 버블정렬 Bubble Sort

버블정렬 알고리즘의 특징

- 장점
  - 구현이 매우 간단하다.
- 단점
  - 순서에 맞지 않은 요소를 인접한 요소화 교환한다.
  - 하나의 요소가 가장 왼쪽에서 가장 오른쪽으로 이동하기 위해허는 배열에서 모든 다른 요소들과 교환되어야 한다.
  - 특히 특정 요소가 최종 정렬 위치에 이미 있는 경우라도 교환되는 일이 일어난다.
- 일반적으로 자료의 교환 작업(SWAP)이 자료의 이동(MOVE)보다 더 복잡하기 때문에 버블 정렬은 단순성에도 불구하고 거의 쓰이지 않는다.

3 버블정렬 Bubble Sort

버블정렬의 시간복잡도

- 시간복잡도를 계산한다면
  - 비교횟수
    - 최상, 평균, 최악 모두 일정
    - $(n-1), (n-2), \ldots, 2, 1 = \frac{n(n-1)}{2}$
  - 교환횟수
    - 데이터 가 역순으로 정렬되어 있는 최악의 경우, 한번 교환하기 위하여 3번의 이동(SWAP)이 필요하므로 (비교횟수 X 3) 번 = $\frac{3n(n-1)}{2}$
    - 데이터가 이미 정렬되어있는 최상의 경우, 자료의 이동이 발생하지 않는다.
    - $T(n) = O(n^2)$

4 정렬 알고리즘 시간 복잡도 비교

Name	Best	Avg	Worst	Run-time(정수:60,000개) 단위 : sec
삽입정렬	n	n^2	n^2	7.438
선택정렬	n^2	n^2	n^2	10.842
버블정렬	n^2	n^2	n^2	22.894

- 단순(구현 간단)하지만 비효율적인 방법
  - 삽입정렬, 선택정렬, 버블정렬

- 복잡하지만 효율적인 방법
  - 퀵정렬, 힙정렬, 합병정렬, 기수정렬