Forza

숙명여자대학교 컴과 동아리 – 1주차 : 소트알고리즘을 이용한 배열의 활용

(첫 주부터 미안 얘들아♥ 이런걸 내가 만들 줄 몰랐어..ㅠㅠ made 소리)

자료구조

```
int main(void)
{
    // 배열의 선언
    int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    . . . .
    // 배열에 저장된 값의 합
    for(idx=0; idx<10; idx++)
        sum += arr[idx];
    . . . .
}
```

자료구조

알고리즘

알고리즘은 자료구조에 의존적이다!

자료구조란?

"프로그램 이란 데이터를 표현하고, 그렇게 표현된 데이터를 처리 하는 것이다"

Bubble Sort : 이해

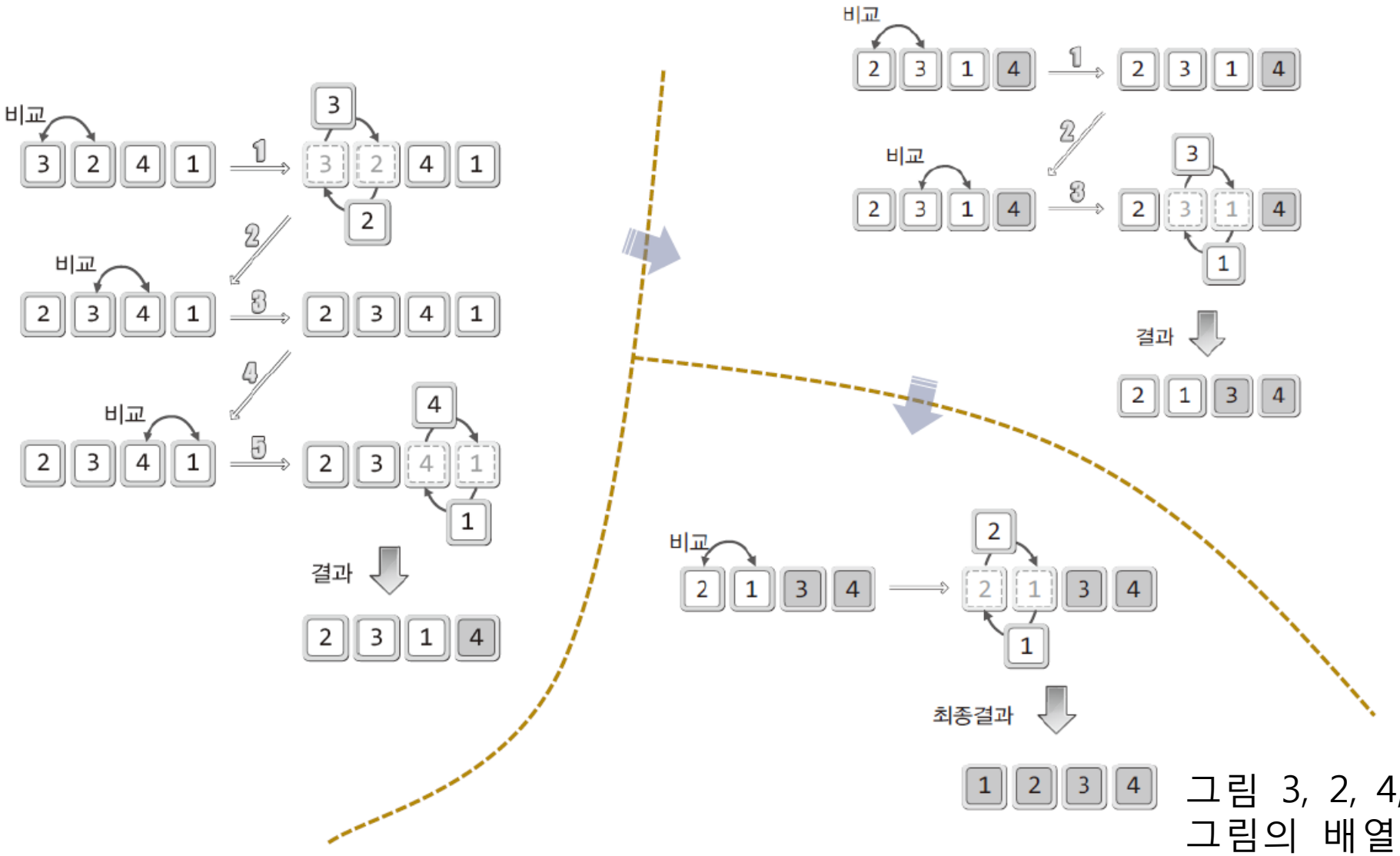


그림 3, 2, 4, 1이 순서대로 저장된 다음 그림의 배열을 '오름차순'으로 정렬하는 과정

Bubble Sort : 구현

버블 정렬?

인접한 두 개의 데이터를
비교해가면서 정렬을
진행하는 방식

앞에서부터 순서대로 비교하고
교환하는 일련의 과정이
거품이 일어나는 모습에 비유되어
버블 정렬이라 이름 지어진 것

```
void BubbleSort(int arr[], int n)
{
    int i, j;
    int temp;

    for(i=0; i<n-1; i++)
    {
        for(j=0; j<(n-i)-1; j++)
        {
            if(arr[j] > arr[j+1])
            {
                // 데이터의 교환 /////
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

버블 정렬의 실질적 구현 코드

```
int main(void)
{
    int arr[4] = {3, 2, 4, 1};
    int i;

    BubbleSort(arr, sizeof(arr)/sizeof(int));

    for(i=0; i<4; i++)
        printf("%d ", arr[i]);

    printf("\n");
    return 0;
}
```

실행결과

```
1 2 3 4
```

Bubble Sort : 구현(2)

swap함수 호출

```
void bubble_sort(element list[], int n){
        int i, j;
        int flag = 1 //swap 발생
        element next;

        for(i = n-1; flag>0; i--){
                flag=0;
                for(j=0; j<i; j++)
                        if(list[j]>list[j+1]){
                                swap(&list[j], &list[j+1]);
                                flag=1; //swap occurs
                        }
                }
}
```

15	4	8	3	50	9	20
4	8	3	15	9	20	50
4	3	8	9	15	20	50
3	4	8	9	15	20	50
3	4	8	9	15	20	50

Bubble Sort : 성능 평가

성능 평가의 두 가지 기준

- 비교의 횟수 : 두 데이터간의 비교연산의 횟수
- 이동의 횟수 : 위치의 변경을 위한 데이터의 이동횟수

```
for(i=0; i<n-1; i++)
{
    for(j=0; j<(n-i)-1; j++)
    {
        if(arr[j] > arr[j+1]) { . . . . }
    }
}
```

비교 연산

최악의 경우

비교의 횟수와 이동의 횟수는 일치한다.

비교의 횟수

$$(n-1) + (n-2) + \cdots + 2 + 1$$

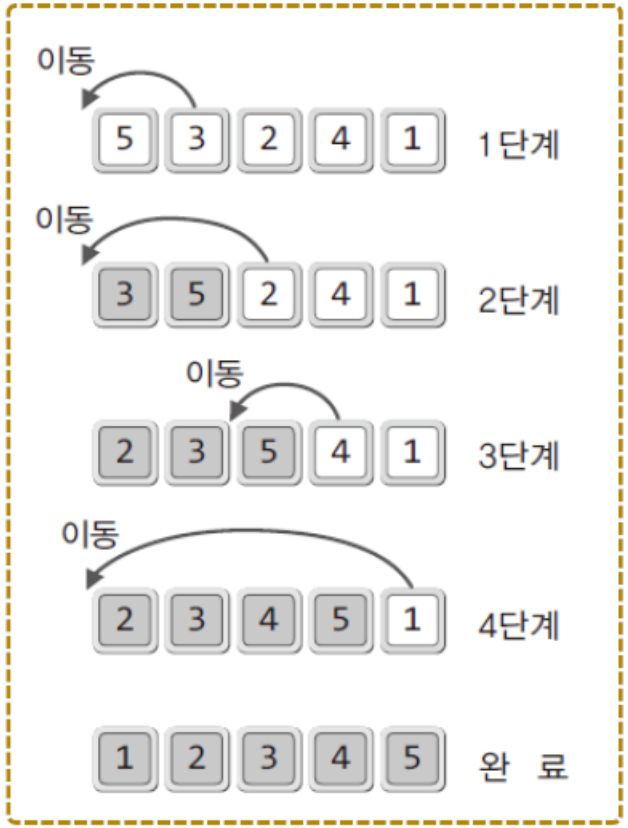
$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \frac{n^2-n}{2}$$

$$O(n^2)$$

클릭하면 생성으로

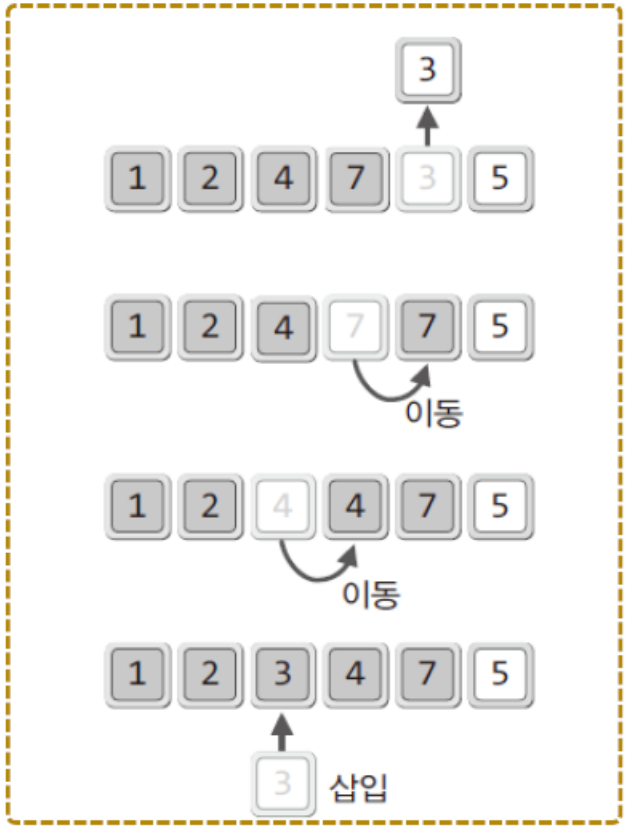
Insertion Sort : 이해

정렬이 완료된 영역과 그렇지 않은 영역을 구분하는 방법



구현을 고려하면!

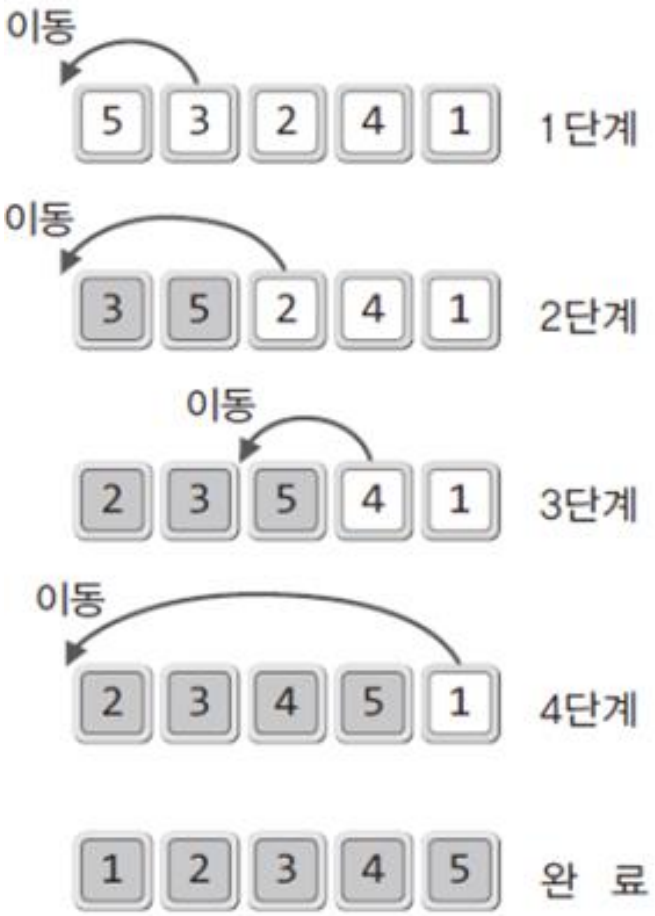
뒤로 밀어 내는 방법을 포함한 그림



삽입 정렬?

그림의 배열은 정렬이 완료된 부분과 완료되지 않은 부분으로 나뉘어 있다. 이렇듯 삽입 정렬은 정렬 대상을 두 부분으로 나눠서, 정렬 안된 부분에 있는 데이터를 정렬 된 부분의 특정 위치에 '삽입'해 가면서 정렬을 진행하는 알고리즘이다.

Insertion Sort : 이해



1. 첫 번째 데이터와 두 번째 데이터를 비교하여, 정렬된 상태가 되도록 두 번째 데이터를 옮기면서 정렬을 시작
2. 첫 번째 데이터와 두 번째 데이터가 정렬이 완료된 영역을 형성
3. 세 번째, 네 번째 데이터가 정렬이 완료된 영역으로 삽입이 되면서 정렬을 이어 나간다. 정렬된 상태로 삽입하기 위해서는 특정위치를 비워야 하고, 비우기 위해서는 데이터들을 한 칸씩 뒤로 미는 연산을 수행해야 한다.

->"정렬이 완료된 영역의 다음에 위치한 데이터가 그 다음 정렬대상이다."

Insertion Sort : 이해

3의 자리를 찾자!

3과 7 비교 후, 7을 한 칸 뒤로 이동

3과 4 비교 후, 4를 한 칸 뒤로 이동

3과 2 비교 후, 3을 삽입!

과제(힘내>_<)

next 잘보세요!!

```
void insertion_sort(element list[], int n){
        int i, j;
        element next;

        for(i = 1; i<n; i++){
                //list 배열 활용 빈칸 채우기
                for(j=i-1; j>=0 && next.key < list[j].key; j--)
                        //list 배열 활용 빈칸 채우기
                                //list 배열 활용 빈칸 채우기

                }
}
```

	[0]	[1]	[2]	[3]	[4]
-	5	4	3	2	1
1	4	5	3	2	1
2	3	4	5	2	1
3	2	3	4	5	1
4	1	2	3	4	5