SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

APS기본

SAMSUNG SW ACADEMY FOR YOUTH

APS(Algorithm Problem Solving)

LIST 1

- 알고리즘
- 배열
- 버블 정렬(Bubble Sort)

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

알고리즘

알고리즘

- (명) 알고리즘 : 유한한 단계를 통해 문제를 해결하기 위한 절차나 방법이다. 주로 컴퓨터용어로 쓰이며, 컴퓨터가 어떤 일을 수행하기 위한 단계적 방법을 말한다.

- 간단하게 다시 말하면 어떠한 문제를 해결하기 위한 절차라고 볼 수 있다.

- 예를 들어 1부터 100까지의 합을 구하는 문제를 생각해 보자.

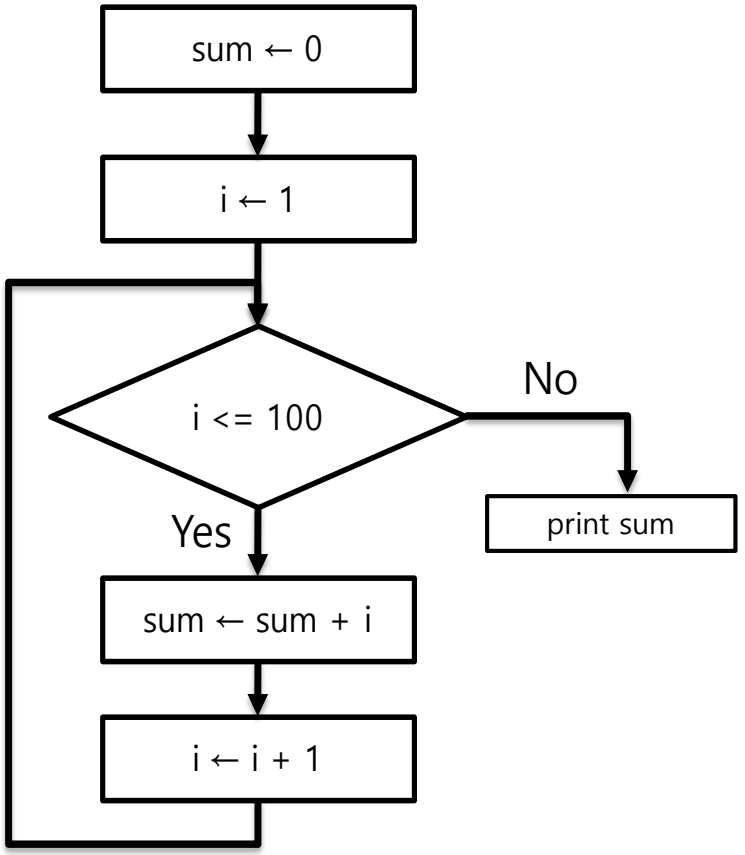
```
    1
+   2
+   3
.....
+ 100
-------
5,050
```

컴퓨터 분야에서 알고리즘을 표현 하는 방법은 크게 두 가지

- 의사코드와 순서도

```
CalcSum( n ) {
    sum ← 0
    for ( i ← 1; i <= n; i ← i + 1) {
        sum ← sum + i;
    }
    return sum;
}
```

Pesudocode 예



순서도 예

알고리즘 성능

- **APS 과정의 목표 중의 하나는 보다 좋은 알고리즘을 이해하고 활용하는 것이다.**

- **무엇이 좋은 알고리즘인가?**
  1. 정확성 : 얼마나 정확하게 동작하는가
  2. 작업량 : 얼마나 적은 연산으로 원하는 결과를 얻어내는가
  3. 메모리 사용량 : 얼마나 적은 메모리를 사용하는가
  4. 단순성 : 얼마나 단순한가
  5. 최적성 : 더 이상 개선할 여지없이 최적화되었는가

알고리즘 성능

- **주어진 문제를 해결하기 위해 여러 개의 다양한 알고리즘이 가능**
  ⇒ 어떤 알고리즘을 사용해야 하는가?
- **알고리즘의 성능 분석 필요**
  - 많은 문제에서 성능 분석의 기준으로 알고리즘의 작업량을 비교한다.
- **예 : 1부터 100까지 합을 구하는 문제**

알고리즘 1	알고리즘 2
1 + 2 + 3 + 4 ..... + 100 5,050	$\frac{100\times(1+100)}{2}=5,050$
99번의 연산 (덧셈 99번)	3번의 연산 (덧셈 1번, 곱셈 1번, 나눗셈 1번)

알고리즘 성능

- 알고리즘의 작업량을 표현할 때 시간복잡도로 표현한다.
- 시간 복잡도(Time Complexity)
  - 실제 걸리는 시간을 측정
  - 실행되는 명령문의 개수를 계산

알고리즘 1	알고리즘 2

CalcSum(n) {
  sum ← 0; // 1번
  for (i ← 1; i <= n; i ← i+1) // 1+1번
  sum ← sum + i; // 1 번
  return sum;
}
	CalcSum(n) {
	  return n*(n+1)/2 //3번
	}

1 + n * 3 = 3n + 1	3번의 연산

알고리즘 성능

시간 복잡도 ≒ 빅-오(O) 표기법

- 빅-오 표기법(Big-Oh Notation)
- 시간 복잡도 함수 중에서 가장 큰 영향력을 주는 n에 대한 항만을 표시
- 계수(Coefficient)는 생략하여 표시
- 예를 들어

$$O(3n + 2) = \underline{O(3n)} = \underline{O(n)}$$

최고차항(3n)만 선택 / 계수 3제거

$$O(2n^2 + 10n + 100) = O(n^2)$$

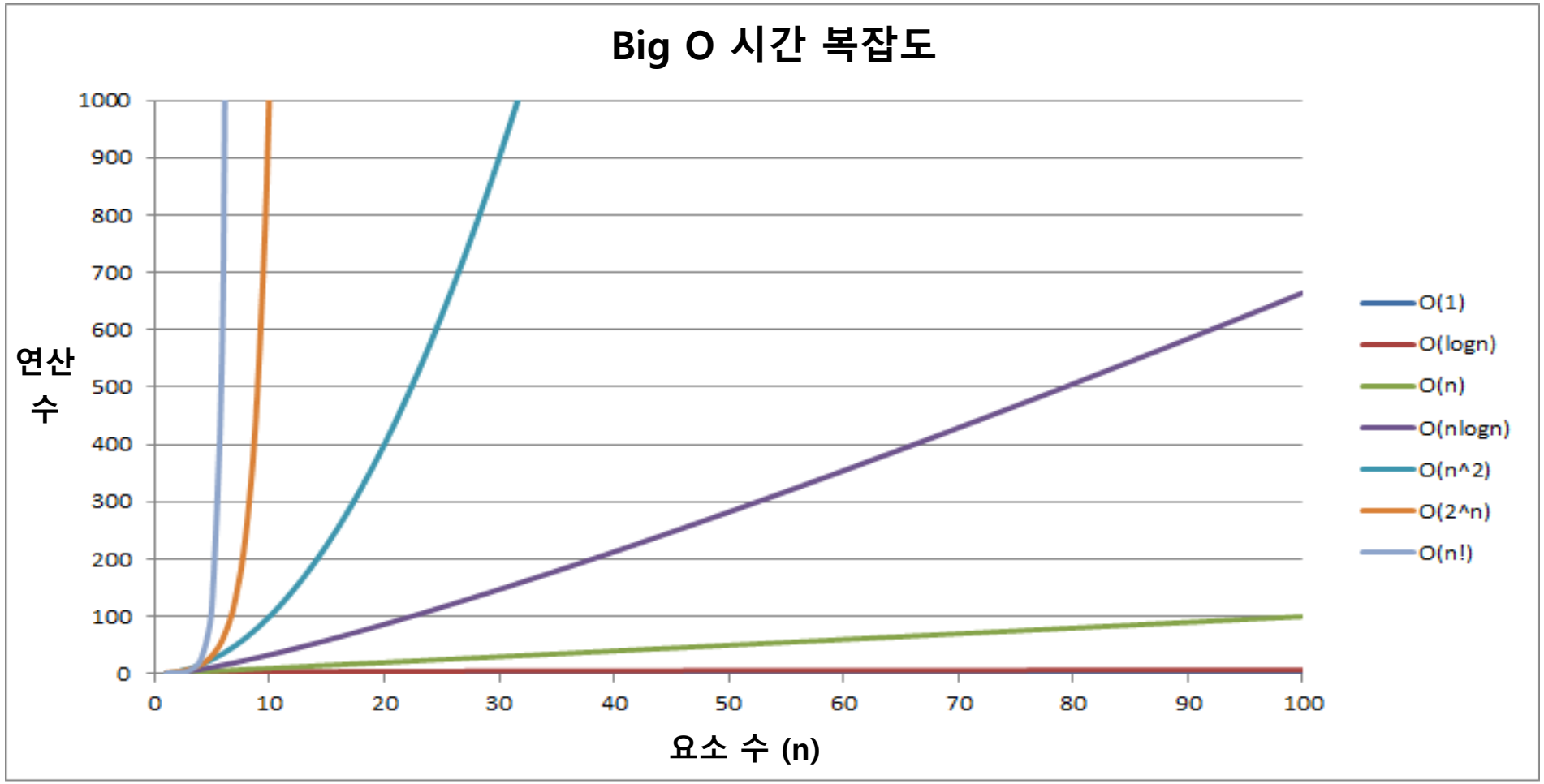
$$O(4) = O(1)$$

- n개의 데이터를 입력 받아 저장한 후 각 데이터에 1씩 증가시킨 후 각 데이터를 화면에 출력하는 알고리즘의 시간복잡도는 어떻게 되나?

O(n)

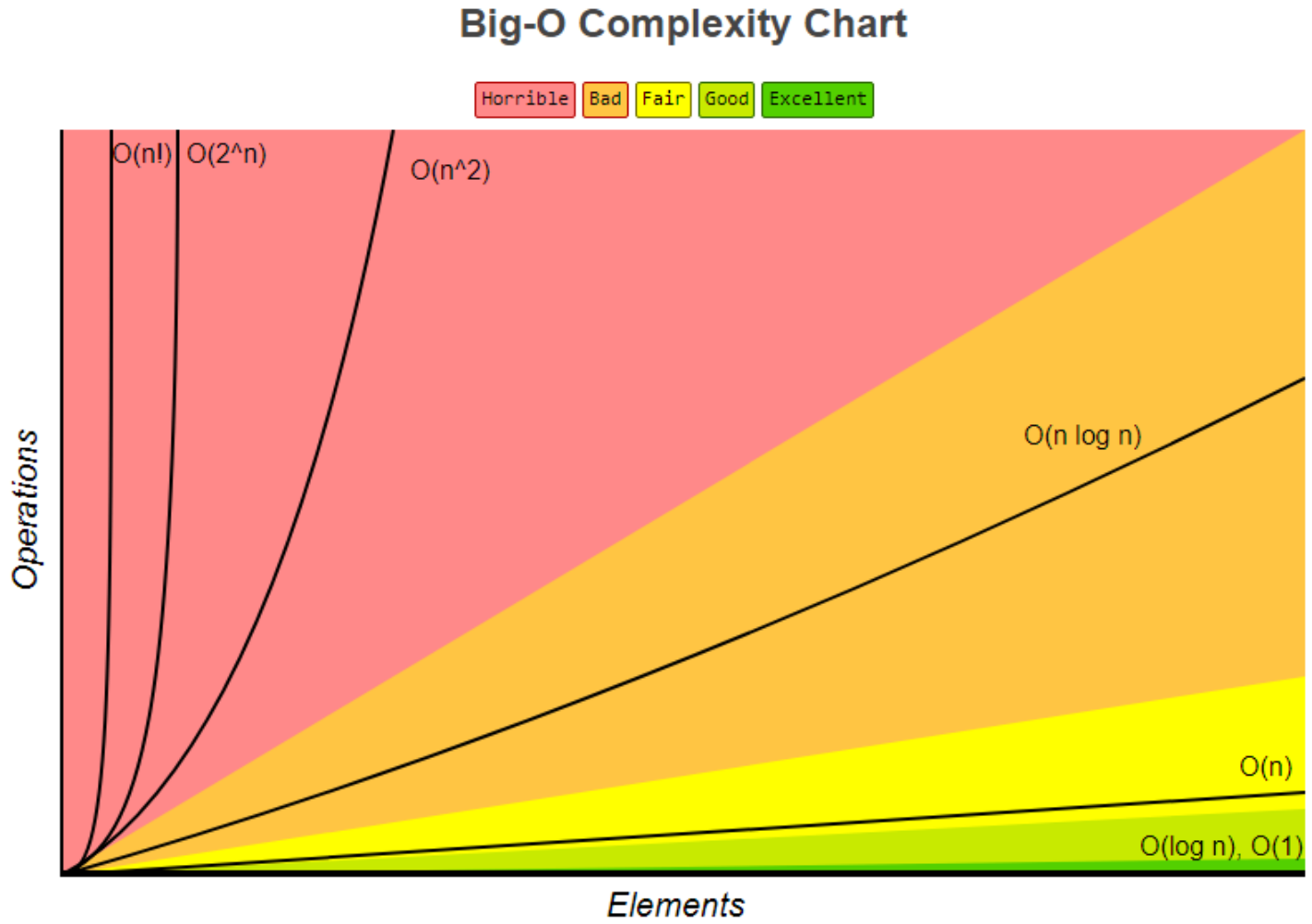
다양한 시간 복잡도의 비교

- 요소 수가 증가함에 따라 각기 다른 시간복잡도의 알고리즘은 아래와 같은 연산 수를 보인다.



다양한 시간 복잡도의 비교

- 요소 수가 증가함에 따라 각기 다른 시간복잡도의 알고리즘은 아래와 같은 연산 수를 보인다.



다양한 시간 복잡도의 비교

시간 복잡도별 실제 실행 시간 비교

1μs = 10^{-6}초
1ms = 10^{-3}초

N	log$N$	N	N log$N$	N^2	N^3	2^N
10	0.003 μs	0.01 μs	0.033 μs	0.1 μs	1 μs	1 μs
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	8 μs	1 ms
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	27 μs	1 s
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	64 μs	18.3 min
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	125 μs	13 days
10^2	0.007 μs	0.10 μs	0.664 μs	10 μs	1 ms	4 X 10^{13} years
10^3	0.010 μs	1.00 μs	9.966 μs	1 ms	1 s	
10^4	0.013 μs	10 μs	130 μs	100 ms	16.7 min	
10^5	0.017 μs	0.10 ms	1.67 ms	10 s	11.6 days	
10^6	0.020 μs	1 ms	19.93 ms	16.7 min	31.7 days	
10^7	0.023 μs	0.01 s	0.23 s	1.16 days	31,709 years	
1 억 10^8	0.027 μs	0.10 s	2.66 s	115.7 days	3.17 X 10^7 years	
10^9	0.030 μs	1 s	29.90 s	31.7 years		

? *5천만에 대한 연산을 수행한다면?*

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

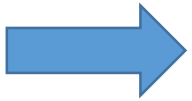
1차원 배열

배열이란 무엇인가

- 일정한 자료형의 변수들을 하나의 이름으로 열거하여 사용하는 자료구조

- 아래의 예는 6개의 변수를 사용해야 하는 경우, 이를 배열로 바꾸어 사용하는 것이다.

```
int num0 = 0;   int num3 = 3;
int num1 = 1;   int num4 = 4;
int num2 = 2;   int num5 = 5;
```



```
int[] nums = { 0, 1, 2, 3, 4, 5 }
```

배열의 필요성

- 프로그램 내에서 여러 개의 변수가 필요할 때, 일일이 다른 변수명을 이용하여 자료에 접근하는 것은 매우 비효율적일 수 있다.

- 배열을 사용하면 하나의 선언을 통해서 둘 이상의 변수를 선언할 수 있다.

- 단순히 다수의 변수 선언을 의미하는 것이 아니라, 다수의 변수로는 하기 힘든 작업을 배열을 활용해 쉽게 할 수 있다.

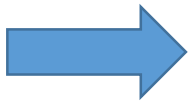
1차원 배열

1차원 배열의 선언

- 자료형 : 배열을 이루는 자료형
- 이름 : 프로그램에서 사용할 배열의 이름
- 길이 : 배열을 이루는 원소의 수

```
int[] nums = new int[6]
```

(1차원 배열 선언의 예)



0	0	0	0	0	0

1차원 배열의 접근

- nums[0] = 10; // 배열 nums의 0번째 원소에 10을 저장
- nums[idx] = 20; // 배열 nums의 idx번째 원소에 20을 저장

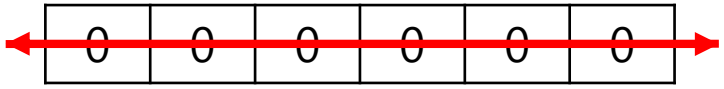
1차원 배열

1차원 배열의 순회

- 배열의 요소를 빠짐 없이 조사하는 방법

```
for(        ;        ;        ) {


}
```



SW 문제를 완벽하게 풀기 위한 5단계

1. 지문을 읽는다.
2. 문제를 이해한다.
3. 문제를 손으로 푼다.
4. 푼 걸 코딩한다.
5. 디버깅하고 검증한다.

출처 : SW Expert Academy

연습 문제

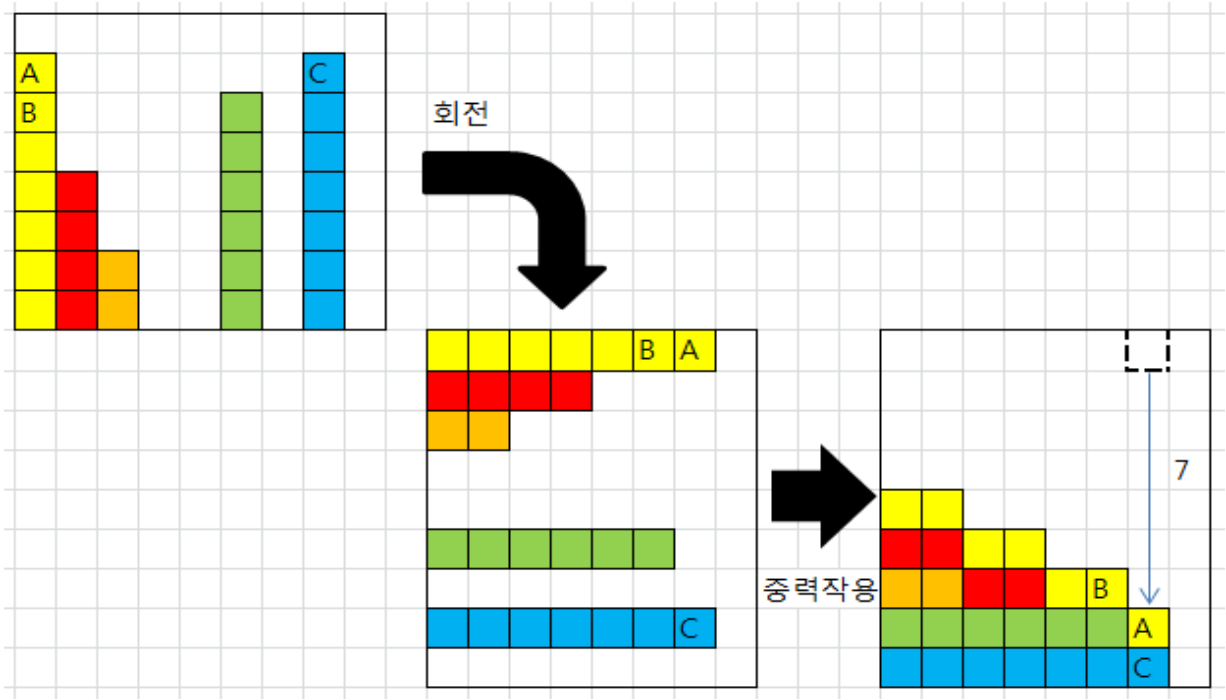
배열 활용 예제 : Gravity

- 상자들이 쌓여있는 방이 있다. 방이 오른쪽으로 90도 회전하여 상자들이 중력의 영향을 받아 낙하한다고 할 때, 낙차가 가장 큰 상자를 구하여 그 낙차를 리턴 하는 프로그램을 작성하시오.
- 중력은 회전이 완료된 후 적용된다.
- 상자들은 모두 한쪽 벽면에 붙여진 상태로 쌓여 2차원의 형태를 이루며 벽에서 떨어져서 쌓인 상자는 없다.
- 방의 가로길이는 항상 100이며, 세로 길이도 항상 100이다.
- 즉, 상자는 최소 0, 최대 100 높이로 쌓을 수 있다.

연습 문제

그림 설명

- 아래 예) 총 26개의 상자가 회전 후, 오른쪽 방 그림의 상태가 된다. A 상자의 낙차가 7로 가장 크므로 7을 리턴하면 된다.
- 회전 결과, B상자의 낙차는 6, C상자의 낙차는 1이다.



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

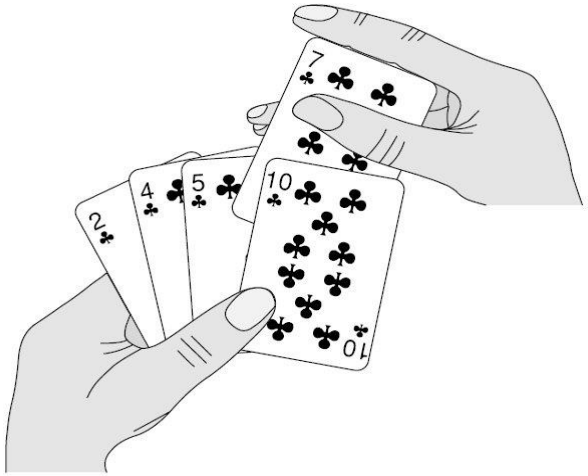
정렬(Sort)

정렬

- 2개 이상의 자료를 특정 기준에 의해 작은 값부터 큰 값(오름차순 : ascending), 혹은 그 반대의 순서대로(내림차순 : descending) 재배열하는 것

- 키
  - 자료를 정렬하는 기준이 되는 특정 값

<서류를 번호대로 정렬하기>



<카드를 번호대로 정렬하기>

정렬의 종류

- **대표적인 정렬 방식의 종류**
  - 버블 정렬 (Bubble Sort)
  - 선택 정렬 (Selection Sort)
  - 삽입 정렬 (Insertion Sort)
  - 카운팅 정렬 (Counting Sort)
  - 병합 정렬 (Merge Sort)
  - 퀵 정렬 (Quick Sort)

- **APS 과정을 통해 자료구조와 알고리즘을 학습하면서 다양한 형태의 정렬을 학습하게 된다.**

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

버블 정렬
(Bubble Sort)

버블 정렬 (Bubble Sort)

인접한 두 개의 원소를 비교하며 자리를 계속 교환하는 방식

정렬 과정

- 첫 번째 원소부터 인접한 원소끼리 계속 자리를 교환하면서 맨 마지막 자리까지 이동한다.
- 한 단계가 끝나면 가장 큰 원소가 마지막 자리로 정렬된다.
- 교환하며 자리를 이동하는 모습이 물 위에 올라오는 거품 모양과 같다고 하여 버블 정렬이라고 한다.

시간 복잡도

- $O(n^2)$

버블 정렬 (Bubble Sort)

- {55, 7, 78, 12, 42}를 버블 정렬하는 과정
  - 첫 번째 패스

55	**7**	78	12	42
7	**55**	**78**	12	42
7	55	**78**	**12**	42
7	55	12	**78**	**42**
7	55	12	42	**78**

정렬된 원소 미정렬 원소

버블 정렬 (Bubble Sort)

- 두 번째 패스

7	**55**	12	42	78
7	**55**	**12**	42	78
7	12	**55**	**42**	78
7	12	42	**55**	**78**

정렬된 원소 미정렬 원소

버블 정렬 (Bubble Sort)

- 세 번째 패스

7	**12**	42	55	78
7	**12**	**42**	55	78
7	12	**42**	**55**	**78**

정렬된 원소 미정렬 원소

버블 정렬 (Bubble Sort)

- 네 번째 패스

7	12	42	55	78

7	12	42	55	78

- 정렬 끝

7	12	42	55	78

정렬된 원소 미정렬 원소

버블 정렬 (Bubble Sort)

배열을 활용한 버블 정렬

- 앞서 살펴 본 정렬 과정을 코드로 구현하면 아래와 같다.

```
BubbleSort(int[] a, int N) { // a[] : 정렬할 배열, N : 배열의 크기
    for i from N-1 to 0 decreasing by 1 {
        for j from 0 to i {
            if(a[j] > a[j+1]) then {
                temp ← a[j];
                a[j] ← a[j+1];
                a[j+1] ← temp;
            }
        }
    }
}
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미