

알고리즘1

교재 참고 예제로 구성되어 있습니다
본 문서는 학습 참고용으로만 사용하시고 웨이나 타인에게 배포를 금합니다

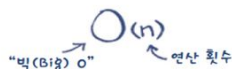
알고리즘이란?

- 어떠한 문제를 해결하기 위해 정해놓은 일련의 절차나 방법
-
- 주어진 입력을 출력으로 만드는 과정을 구체적이고 명료하게 표현 한 것.
-
- 올바른 알고리즘이란 어떤 경우에도 실행결과가 동일하게 나오는 것을 의미한다
-
- 어떤 경우는 맞고 어떤 경우는 틀리면 안된다

빅오 표기법 (Big O Notation)

- 알고리즘의 성능을 수학적으로 표기해 주는 표기법
- 알고리즘의 시간과 공간 복잡도를 표현함
- 입력 데이터 크기 증가할 때 알고리즘 연산 시간(횟수)의 증가 방식

■ $O(\log n)$

 "빅(Big) O" 연산 횟수

■ Ex) 100개의 원소를 가진 리스트에 대한 단순 / 이진 탐색 시간 비교

- 10억 개로 늘어난다면?
- 이진 탐색과 단순 탐색의 실행 시간은 같은 비율로 증가하지 않음

	단순 탐색	이진 탐색
100개	100밀리 초	7밀리 초
10,000개	10초	14밀리 초
1,000,000,000개	11일	32밀리 초



빅오 표기법 (Big O Notation)

16개의 사각형 칸이 생기도록 격자 만들기

[1] 한 번에 하나의 상자 그리는 경우

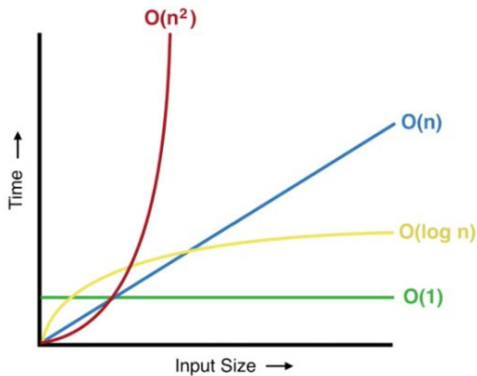
- 16단계 필요 => **$O(n)$ 시간**

[2] 종이를 한 번 접는것이 하나의 연산인 경우

- 한번 접을 때마다 사각형의 수가 2배로 증가
- 4단계 필요 => **$O(\log n)$ 시간**



Big-O



- 알고리즘의 실제 러닝타임을 표기하는것이라기 보다
- 데이터나 사용자의 증가률에 따른 알고리즘의 성능을 예측하는것이 목표임
- $O(1)$ => 언제나 일정한 속도로 결과를 반환
- constant time
- $O(n)$ ==> 입력데이터의 크기에 비례해서 처리 시간이 걸림
- linear time
- $O(n^2)$ ==> 가로,세로 만큼늘어남
- quadratic time

```
F(int[] n){  
    for i=0 to n.length  
        print i;  
}
```

```
F(int[] n){  
    for i=0 to n.length  
        for j=0 to n.length  
            print i+j;  
}
```

Big-O

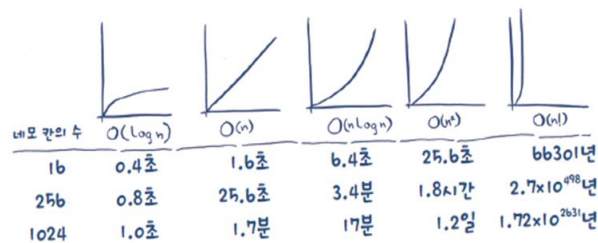
$O(\log n) : 2^x = n$

이진검색 경우 해당

```
ex) binary search
F(k,arr,s,e){
    if(s>e) return -1;
    m=(s+e)/2;
    if(arr[m]==k) return m;
    else if(arr[m]>k) return F(k, arr, s,m-1);
    else return F(k,arr, m+1,e);
}
```

❖ 많이 사용하는 빅오 실행 시간 예시

- $O(\log n)$
 - 이진 탐색
- $O(n)$
 - 단순 탐색
- $O(n \cdot \log n)$
- $O(n^2)$
- $O(n!)$



기본 알고리즘

- Q1. 최대값 구하기
- Q2. 세 값의 대소관계와 중앙값 구하기
- Q3. 루프문을 이용한 곱셈표 출력하기
- Q4. 배열 요소의 최대값 구하기
- Q5. 두 배열의 값이 같은지 비교하기
- Q6. 배열1의 모든 요소를 배열2에 카피하기
- Q7. 정수를 임의의 기수로 변환하기
- Q8. 문자열에서 특정 문자 갯수 구하기
- Q9. 알파벳 대소문자 변환
- Q10. 가장 긴 단어 찾기
-

Q1. 세 개의 최댓값

1. 2개의 정수값을 입력 받아 최대값을 구하시오.
2. 3개의 정수값을 입력받아 최댓값을 구하시오
3. 4개의 정수값을 입력받아 최댓값을 구하시오



실행 결과

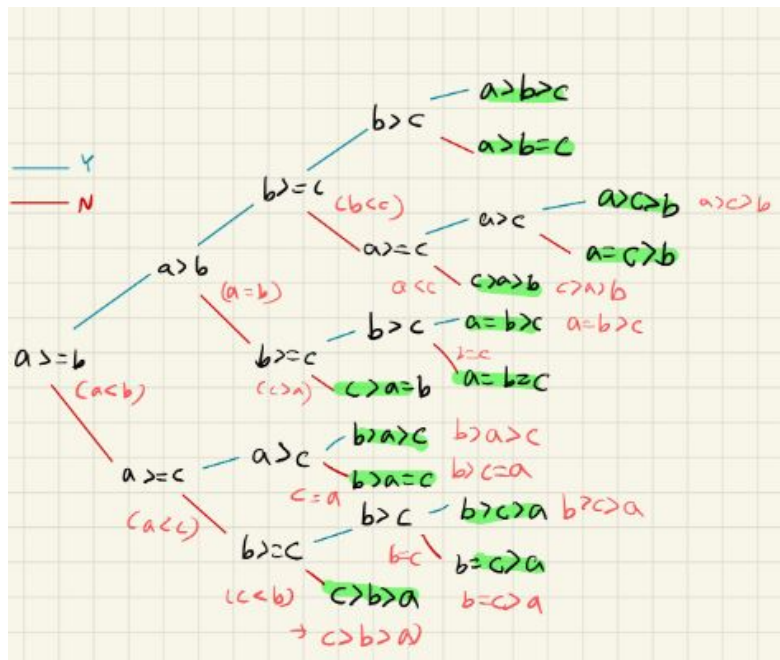
세 정수의 최댓값을 구합니다.
a의 값 : 1
b의 값 : 3
c의 값 : 2
최댓값은 3입니다.

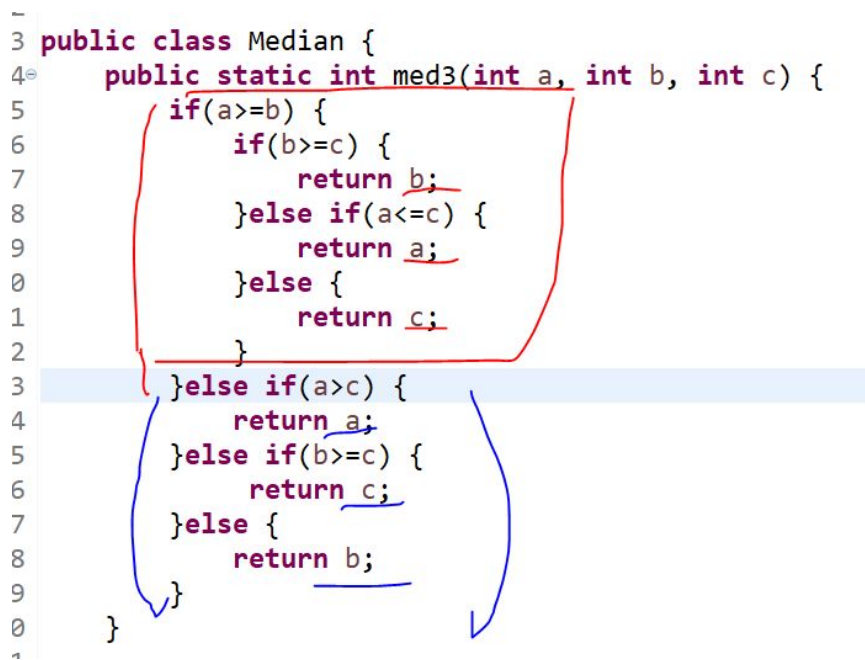
Q2. 세 정수의 중앙값 구하기

- 세 값의 대소 관계의 조합은 13가지 종류가 나올 수 있다.
- 이 조합을 나열하면 나무형태(tree)가 되어 결정 트리(decision tree)라고 한다
-
- 결정 트리는 왼쪽 끝($a \geq b$)에서 시작해 오른쪽 끝으로 이동한다
- 조건이 성립하면 검은색 줄, 조건이 성립하지 않으면 빨간색 줄을 통해 이동한다.
- 이렇게 결정트리를 미리 그려 두면 알고리즘이 모든 경우를 빼먹지 않고 제대로 작동하는지 쉽게 파악할 수 있다.

결정트리

- a, b, c 세 값이 주어졌을 때
- 결정 트리는 왼쪽 끝($a \geq b$)에서 시작해 오른쪽 끝으로 이동한다
- 조건이 성립하면 검은색 줄, 조건이 성립하지 않으면 빨간색 줄을 통해 이동한다.





```

3 public class Median {
4     public static int med3(int a, int b, int c) {
5         if(a>=b) {
6             if(b>=c) {
7                 return b;
8             }else if(a<=c) {
9                 return a;
10            }else {
11                return c;
12            }
13        }else if(a>c) {
14            return a;
15        }else if(b>=c) {
16            return c;
17        }else {
18            return b;
19        }
20    }
21 }

```

Q3 곱셈표 출력

- 중첩 반복문을 이용해 오른쪽 그림과 같은 곱셈표를
- 그림과 같은 형식으로 출력하세요

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Q4. 배열요소의 최댓값 구하기

- 사람수를 입력받고 해당 사람수 만큼
- 키 값을 랜덤하게 100~190사이의 키값으로 생성해 배열에 저장하세요
- 저장된 키 값들 중 가장 큰 사람의 키값을 구해 출력하세요
- 가장 작은 사람의 키값도 구해 출력하세요

키의 최대값을 구합니다.

사람수를 입력하세요: 5

랜덤한 키값 생성-----

height[0] : 175

height[1] : 138

height[2] : 182

height[3] : 170

height[4] : 132

제일 큰 키: 182

제일 작은 키: 132

Q5. 두 배열의 비교

- 여러 개의 배열을 처리하는 알고리즘을 살펴보자.
- 두 배열 arr1과 arr2의 모든 요소값이 같은가를 판단하는 프로그램을 작성하세요

요숫수 입력:

4

arr1[0] : 11

arr1[1] : 22

arr1[2] : 33

arr1[3] : 44

요숫수 입력:

4

arr2[0] : 11

arr2[1] : 22

arr2[2] : 33

arr2[3] : 44

배열 a와 b는 같습니다.

요숫수 입력:

4

arr1[0] : 11

arr1[1] : 22

arr1[2] : 33

arr1[3] : 44

요숫수 입력:

4

arr2[0] : 11

arr2[1] : 22

arr2[2] : 3

arr2[3] : 44

배열 a와 b는 같지 않습니다.

Q5. 두 배열의 비교 판단 조건

1. 먼저 두 배열 `arr1, arr2`의 길이(크기)를 비교한다.

길이가 다르면 배열이 같지 않다.

2. 반복문을 돌면서 두 배열을 처음부터 스캔하면서 `arr1[i]`와 `arr2[i]`의

값을 비교한다. 이 과정에서 값이 다른 요소를 발견하면 `false`를 `return`한다

3. 반복문이 중단되지 않고 끝까지 실행된 경우 두 배열을 같다고 판단할 수 있다.

이때 `true`를 반환한다

Q6. 배열 카피하기

배열 **a**를 생성해서 값을 입력받아 저장하고,

배열 **b**를 생성해서 값을 입력받아 저장한 뒤

[1] 배열 **a**의 요소를 **b**로 카피하는 프로그램을 작성하세요

[2] 배열 **a**의 요소를 배열 **b**에 역순으로 카피하는 프로그램을 작성하세요

요솟수 입력:

4

a[0] : 11

a[1] : 22

a[2] : 33

a[3] : 44

요솟수 입력:

6

b[0] : 1

b[1] : 2

b[2] : 3

b[3] : 4

b[4] : 5

b[5] : 6

배열 **a**를 **b**로 카피 완료

11,22,33,44,5,6,

Q6. 역순으로 카피하기

요숫수 입력:

4

a[0] : 11

a[1] : 22

a[2] : 33

a[3] : 44

요숫수 입력:

6

b[0] : 1

b[1] : 2

b[2] : 3

b[3] : 4

b[4] : 5

b[5] : 6

배열 a를 b로 카피 완료

44,33,22,11,5,6,

Q7. 기수 변환

- 10진수 정수를 입력하면 n진수로 변환하여 출력하는 프로그램을 작성하세요
-

기수란?

- 수를 나타내는데 기초가 되는 수
- 10진법에서는 0~9까지의 정수가 기수가 되고
- 2진법에서는 0, 1 이 기수가 된다

10진수를 기수 변환 합니다.
음수가 아닌 정수(양수) 입력:

45

몇 진수로 변환할까요? (2~36): 2

45를 2진수로 변환: 101101

한 번 더 할까요? (1. 예 2. 아니오):

1

10진수를 기수 변환 합니다.
음수가 아닌 정수(양수) 입력:

45

몇 진수로 변환할까요? (2~36): 8

45를 8진수로 변환: 55

한 번 더 할까요? (1. 예 2. 아니오):

1

10진수를 기수 변환 합니다.
음수가 아닌 정수(양수) 입력:

45

몇 진수로 변환할까요? (2~36): 16

45를 16진수로 변환: 2D

한 번 더 할까요? (1. 예 2. 아니오):

Q8. 문자 개수 찾기

한 개의 문자열을 입력받고, 특정문자를 입력받아 해당 특정 문자가
입력받은 문자열에 몇 개 있는지 알아내는 프로그램을 작성하세요

- 대소문자를 구분하지 않으며, 문자열 길이는 100을 넘지 않는다.
- 문자열은 알파벳으로만 구성되어야 한다

문자열을 입력하세요:

Banana is a Good~!!

검색할 문자 한자를 입력하세요:

a

a는 4개 있습니다

문자열을 입력하세요:

Good job!!

검색할 문자 한자를 입력하세요:

g

g는 1개 있습니다

문자열을 입력하세요:

Hava a nice day~

검색할 문자 한자를 입력하세요:

q

q는 0개 있습니다

Q9. 알파벳 대소문자 변환

대문자와 소문자가 같이 존재하는 문자열을
입력받아 대문자는 소문자로 소문자는 대문자로
변환하여 출력하는 프로그램을 작성하세요.

[입력]

첫 줄에 문자열이 입력된다. 문자열의 길이는 100을
넘지 않습니다.

문자열은 영어 알파벳으로만 구성되어 있습니다.



```
<terminated> UpperLowerChange [Java Application] <
알파벳 문자열을 입력하세요:
HiJavaHelloAlgorithm
hIjAVAhELLOaLGORITHM
```

[hint] 알파벳 소문자 아스키코드에서
32를 빼주면 대문자가 됨

Q10. 가장 긴 단어 찾기

한 개의 문장이 주어지면 그 문장 속에서 가장 긴 단어를 출력하는 프로그램을 작성하세요. 문장속의 각 단어는 공백으로 구분됩니다.

[입력]

첫 줄에 길이가 100을 넘지 않는 한 개의 문장이 주어집니다. 문장은 영어 알파벳으로만 구성되어 있습니다.

[출력]

첫 줄에 가장 긴 단어를 출력한다. 가장 길이가 긴 단어가 여러개일 경우 문장속에서 가장 앞쪽에 위치한 문자열을 출력합니다

문자열을 입력하세요:

```
Never mind I am Fine  
Never
```

String클래스의 split() 또는
indexOf()를 활용해보자