

자료구조-알고리즘-시간복잡도

:: 자료구조 예

• 변수(variable)

배열(array), 리스트(list)
 접근: 원소의 index
 읽기, 쓰기: a[0]
 삽입: append
 삭제: pop

• gcd_sub, gcd_mod, gcd_rec

:: 인류 최초의 알고리즘 : 최대공약수(GCD)

나머지를 활용한 방법이 더 효율적임.

gcd_sub의 경우
gcd(2,100) 경우 100을 한번 빼면 98 .. 이런 식으로 계산이 50번 이루어짐

gcd_mod의 경우 gcd(2,100)을 나머지로 한번만에 끝내버림. gcd_rec = gcd(a, b%a) or gcd(a%b, b)

최대공약수 알고리즘 정리 페이지 \rightarrow Code page

:: 알고리즘의 시간복잡도 (time complexity)

```
algorithm arrayMax(A,n)
input: n개의 정수를 저장한 배열 A
output: A의 수 중에서 최대값
currrentMax = A[0]
for i = 1 to n-1 do
    if currentMax < A[i]
        currentMax = A[i]
return currentMax
```

방법

1. 모든 입력에 대해 기본 연산 횟수를 더한 후 평균 현실적으로 불가능

2. 가장 안 좋은 입력(worstcase input)에 대한 기본 연산 횟수를 측정

T(n) = 2n-1

알고리즘 수행 시간 = 최악의 입력에 대한 기본연산 횟수

```
algorithm sum1(A,n):
    sum = 0
    for i = 0 to n-1 do
        if A[i] % 2 == 0:
```

sum += A[i]
return sum

- 1. sum = 0이다를 대입함으로써 1번
- 2. %와 == 기본연산으로 2번
- 3. +=로 인해 2번

worst case는 모든 값이 짝 수일 경우 전부 해당이 되기 때문에 최악의 입력

• 항상 참이 되기 때문

T(n) = 4n+1

4n인 이유는 for문 안에서 기본연산으로 인해 4번인데 for문은 n번만큼 반복하므로 4n이다. +1은 제일 첫번째 sum = 0으로 인한 1번

:: Big-O 표기법

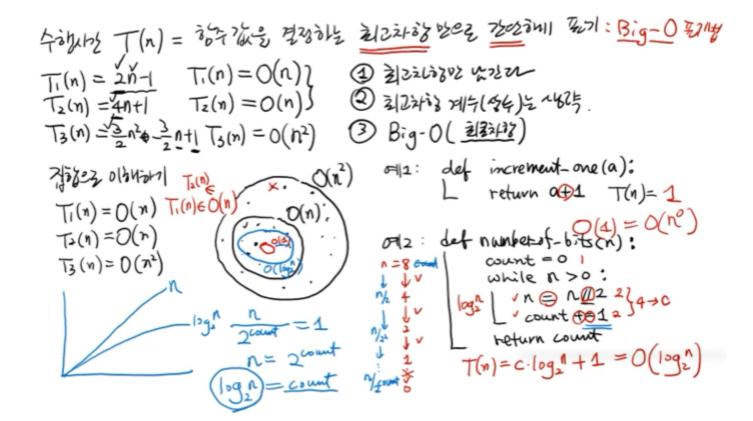
알고리즘의 수행시간 = 최악의 경우의 입력에 대한 기본연산 횟수

수행시간 T(n) = 함수값을 결정하는 최고차항만으로 간단하게 풀기 : **Big-O표기법**

T1(n) = 2n - 1 T1(n) = O(n)

T2(n) = 4n + 1 T1(n) = O(n)

- 1. 최고차항만 남긴다.
- 2. 최고차항 계수(상수)는 생략
- 3. Big-O(최고차항)



자료구조-알고리즘-시간복잡도