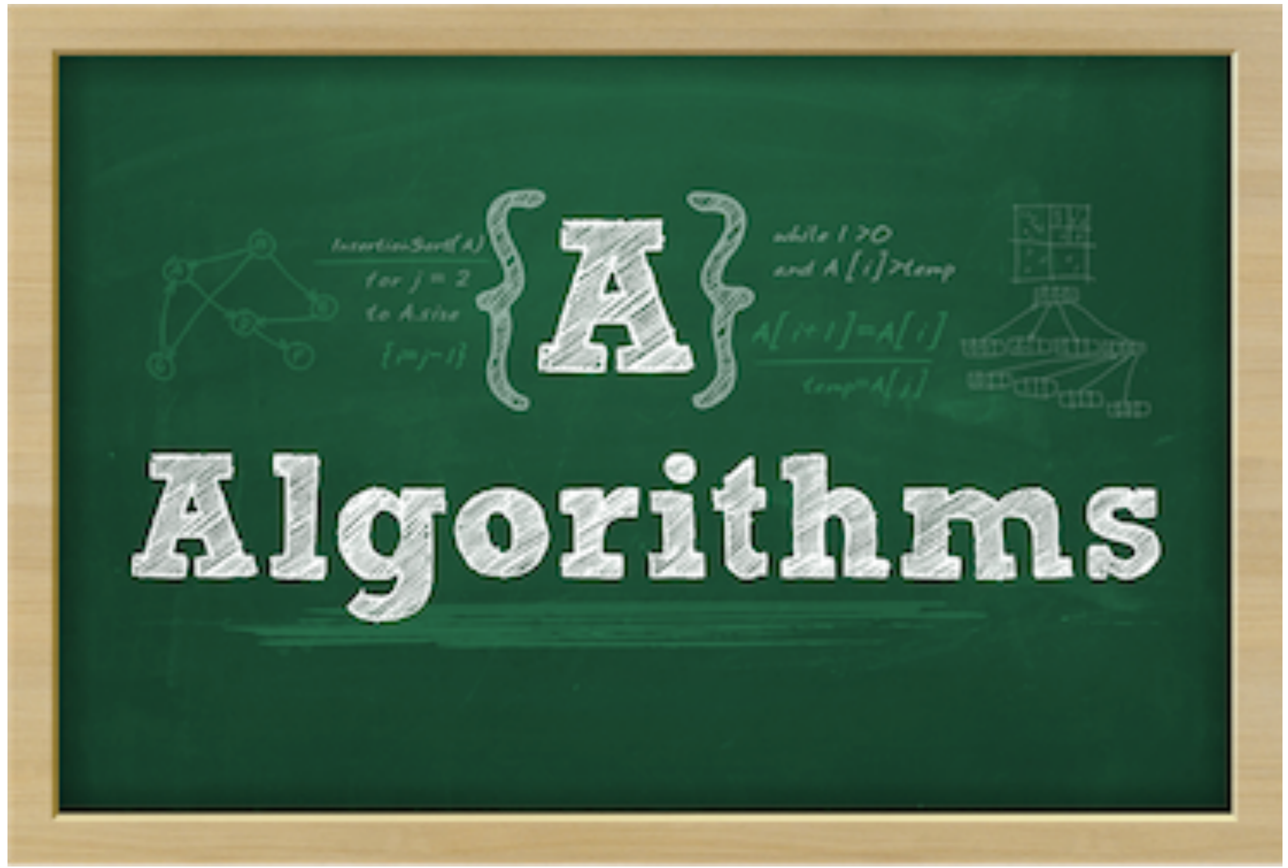


제1장

알고리즘의 분석: 점근적 분석법

알고리즘의 분석

- 알고리즘의 실행 시간 및 기타 자원의 사용량을 분석
- 기타 자원으로는 메모리, 저장장치, 통신 등
- 주로 실행시간의 분석에 집중. 왜?

시간복잡도(time complexity)

- **실행시간은 실행환경에 따라 달라짐**
  - 하드웨어, 운영체제, 언어, 컴파일러 등
- **실행 시간을 측정하는 대신 "연산의 실행 횟수를 카운트"**
- **연산의 실행 횟수는 입력 데이터의 크기에 관한 함수로 표현**
- **데이터의 크기가 같더라도 실제 데이터에 따라서 달라짐**
  - 최악의 경우 시간복잡도 (worst case)
  - 평균 시간복잡도 (average case)

점근적(Asymptotic) 분석

- **점근적 표기법을 사용**
  - 데이터의 개수 n→∞일때 수행시간이 증가하는 growth rate로 시간복잡도를 표현하는 기법
  - Θ-표기, O-표기 등을 사용
- **유일한 분석법도 아니고 가장 좋은 분석법도 아님**
  - 다만 (상대적으로) 가장 간단하며
  - 알고리즘의 실행환경에 비의존적임
  - 그래서 가장 광범위하게 사용됨

점근적 분석의 예: 선형 시간복잡도

```
int sum(A[], int n)
{
    sum ← 0 ;
    for i ← 1 to n
       sum=sum+A[i];
    return sum;
}
```

이 알고리즘에서 가장 자주 실행되는 문장이며, 실행 횟수는 항상 n번이다. 가장 자주 실행되는 문장이 n번이라면 모든 문장의 실행 횟수의 합은 n에 선형적으로 비례하며, 모든 연산들의 실행횟수의 합도 역시 n에 선형적으로 비례한다.

선형 시간복잡도를 가진다고 말하고 O(n)이라고 표기한다.

선형 시간복잡도

```
int search(x[], int target)
{
    for i ← 1 to n
      if x[i]==target
        return i;
    }
    return -1;
}
```

이 알고리즘에서 가장 자주 실행되는 문장이며, 실행 횟수는 최악의 경우 n번이다.

최악의 경우 시간복잡도는 O(n)이다.

Quadratic

```
boolean isDistinct( x[] )
{
  for i ← 1 to n
    for j ← i+1 to n
      if x[i]==x[j]
        return false;
  return true;
}
```

최악의 경우 배열에 저장된 모든 원소 쌍들을 비교하므로 비교 연산의 횟수는 $n(n-1)/2$이다.
따라서 시간복잡도는 $O(n^2)$이다.

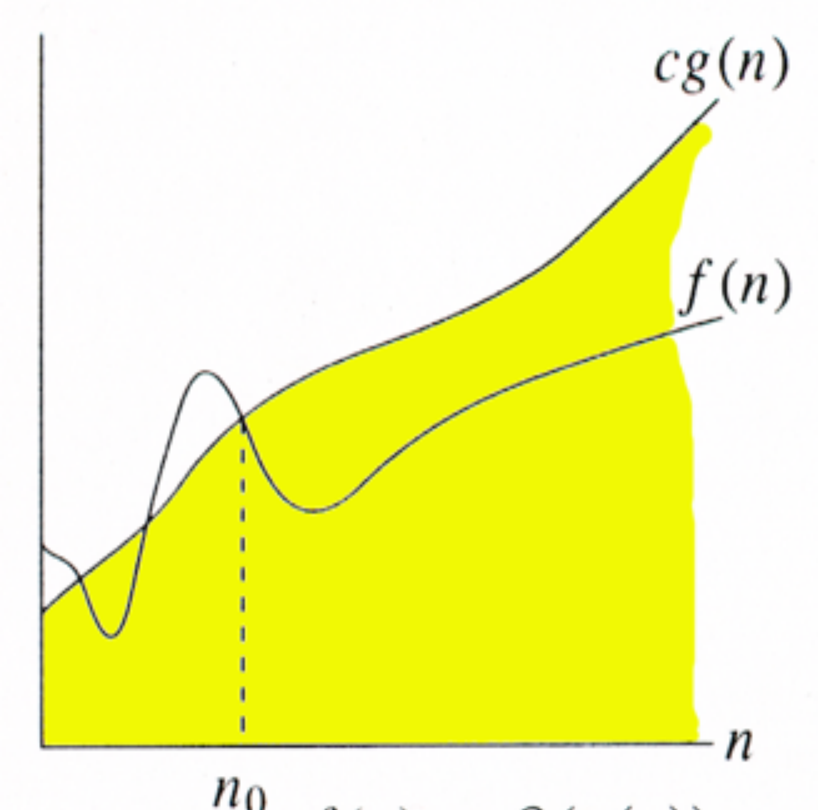
이진검색

```
int binarySearch(int n, int [] data, int target) {
   int begin = 0, end = n-1;
   while(begin <= end)  {
       int middle = (begin +end)/2;
       if (data[middle]==target)
           return middle;
       else if (data[middle]<target)
           begin = middle+1;
       else
           end = middle-1;
   }
   return -1;
}
```

시간복잡도는 ?

점근표기법: O-표기

$$f(n) \in O(g(n)) \text{ if there exist constant } c > 0 \text{ and } n_0 \geq 0$$
$$\text{such that for all } n \geq n_0 \text{ we have } f(n) \leq cg(n)$$



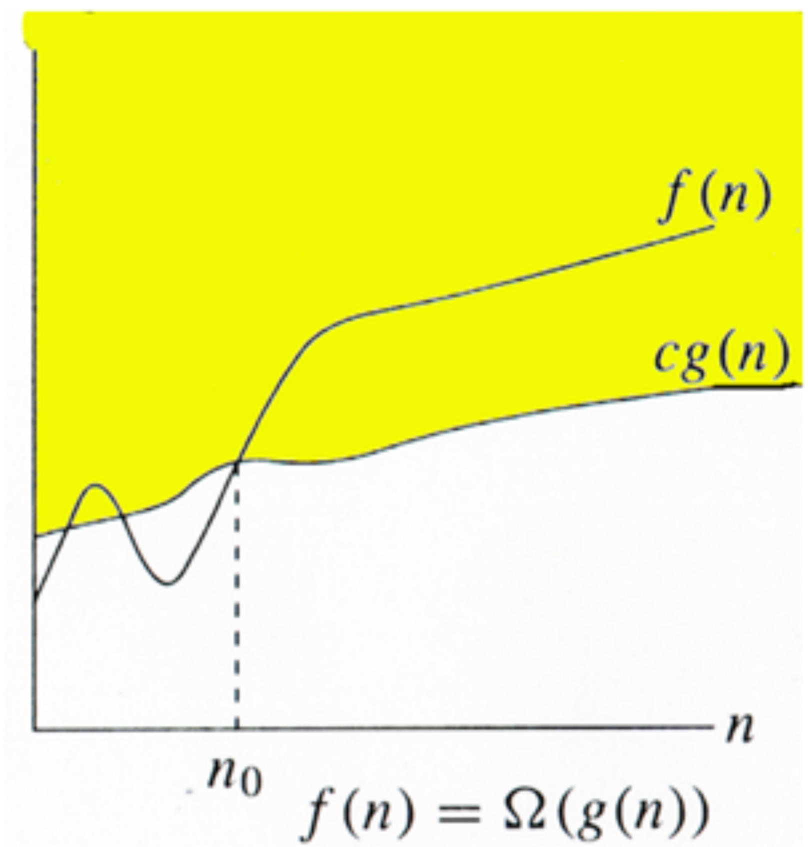
$f(n) = O(g(n))$

$$f(n) = 32n^2 + 17n^2 - 32$$
$$f(n) \in O(n^2)$$
$$\text{but also } f(n) \in O(n^3) \text{ and } O(n^2 \log n)$$

upper bound를 표현

점근표기법: Ω-표기

$$f(n) \in \Omega(g(n)) \text{ if there exist constant } c > 0 \text{ and } n_0 \geq 0$$
$$\text{such that for all } n \geq n_0 \text{ we have } f(n) \geq cg(n).$$



$$f(n) = 32n^2 + 17n^2 - 32$$
$$f(n) \in \Omega(n^2)$$
$$\text{but also } f(n) \in \Omega(n) \text{ and } \Omega(n \log n)$$

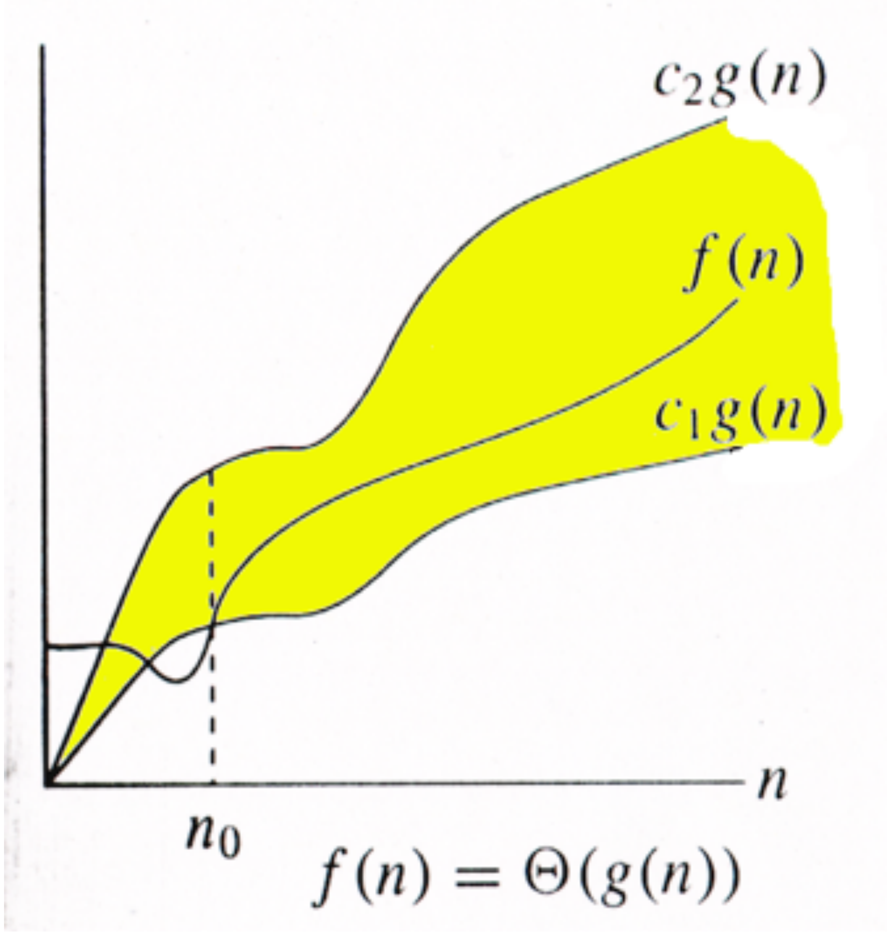
lower bound를 표현

점근표기법 : Θ-표기

$$f(n) \in \Theta(g(n)) \text{ if there exist constants } c_1 > 0, c_2 > 0, \text{ and } n_0 \geq 0$$
$$\text{such that for all } n \geq n_0 \text{ we have } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$$

or

$$f(n) \in \Theta(g(n)) \text{ if } f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$$



$$f(n) = 32n^2 + 17n^2 - 32$$
$$f(n) \in \Theta(n^2)$$
$$\text{but } f(n) \notin \Theta(n^3), f(n) \notin \Theta(n)$$

upper bound와
lower bound를 동시에 표현

- $f(n) \in O(g(n))$을 $f(n) = O(g(n))$으로 쓰는 경우가 많음
- 차수가 $k \geq 0$인 모든 다항식은 $O(n^k)$이다.

$$\begin{aligned} f(n) &= c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0 \\ &= O(n^k) \end{aligned}$$

- 차수가 p인 다항식과 q인 다항식의 합은 $O(n^{\max\{p,q\}})$ 이다.

$$\text{If } g(n) = O(n^p) \text{ and } h(n) = O(n^q),$$
$$\text{then } f(n) = g(n) + h(n) = O(n^{\max(p,q)})$$

Θ-표기에 대해서도 성립함

Exercise

- **다음 테이블을 YES 혹은 NO로 채워라.**

A	B	$A = O(B)?$	$A = \Theta(B)?$
$\log^k n$	n^ϵ		
n^k	c^n		
$\sqrt{n}$	$n^{\sin n}$		
2^n	$2^{n/2}$		
$n^{\log c}$	$c^{\log n}$		
$\log(n!)$	$\log(n^n)$		

A와 B는 n에 관한 함수. 나머지는 상수들