자료구조

Chap 2. Analysis

2018년 1학기

컴퓨터과학과
민 경 하

Contents

2. Analysis

2.1 Performance

2.2 Asymptotic complexity

2.3 Big-O Notation

2.1 Performance

- Three aspects of performance
  - Best case
    - Game score
  - Average case
    - GPA
    - ERA
  - Worst case
    - ATM
    - Guarantee

Summary

(1) Worst case → guarantee

2.1 Performance

- Space-related performance
  – **Space-complexity**
  "the amount of memory that it needs to run to completion"

- Time-related performance
  – **Time-complexity**
  "the amount of computer time that it needs to run to completion"

2.1 Performance

- Example of space complexity
  - Get n integers and sum them all

```
int i, x, sum;
for ( i = 0, sum = 0; i < n; i++ ) {
    cin >> x;
    sum += x;
}
cout << sum;
```

How many variables this program use? 3

↓

Space complexity is O(1)

```
int i, *x, sum;
x = new int[n];
for ( i = 0; i < n; i++ )
    cin >> x[i];
for ( i = 0, sum = 0; i < n; i++ )
    sum += x[i];
cout << sum;
```

How many variables this program use? n

↓

Space complexity is O(n)

Summary

(1) Worst case → guarantee

(2) Time complexity

2.2 Asymptotic complexity

- Asymptotic complexity
  - To estimate the complexity function for **reasonably large length of input**
  - The size of input ➔ n
  - Represent the complexity as **function of n**

Summary

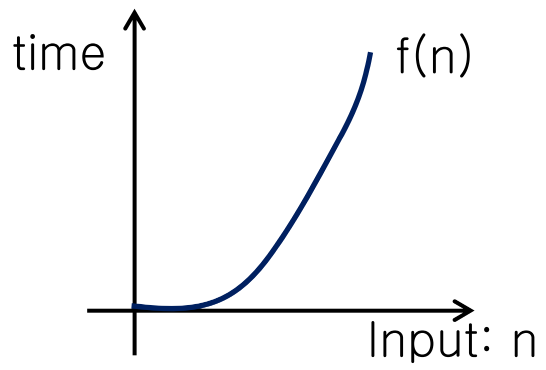
(1) Worst case → guarantee

(2) Time complexity

(3) Asymptotic complexity → very large input + increase of time

2.2 Asymptotic complexity

- Asymptotic complexity
  - Performance
    - Measure it in “WORST CASE”
    - Worst case → Guarantee
  - Performance depends on “input”
    - If input is n, then the performance is f(n)
    - Performance of an algorithm: (n, f(n))



Summary

(1) Worst case → guarantee

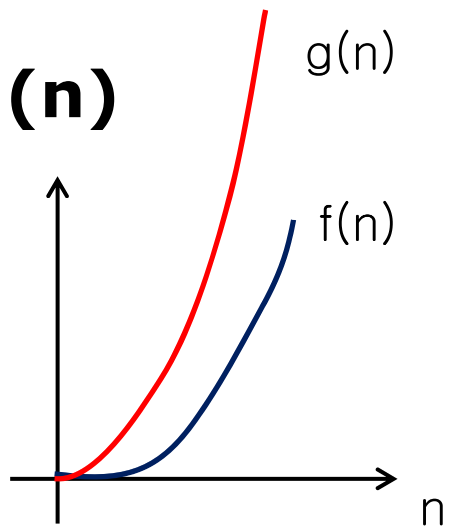
(2) Time complexity

(3) Asymptotic complexity → very large input + increase of time

(4) (input, time) → (n, f(n))

2.2 Asymptotic complexity

- Asymptotic complexity
  - **g(n) is worst of f(n)**
    - In the worst case, **f(n) is better than g(n)**
    - g(n)
      - A standard for measurements
      - 1, n, log n, n^2, n log n, n^n
    - f(n) is better than g(n) → **f(n) ≤ g(n)**
    - **The upper bound of f(n) is g(n)**



Summary

(1) Worst case → guarantee

(2) Time complexity

(3) Asymptotic complexity → very large input + increase of time

(4) (input, time) → (n, f(n))

(5) Standards → 1, n, n^2, 2^n, log n, ...

Summary

(1) Worst case → guarantee

(2) Time complexity

(3) Asymptotic complexity → very large input + increase of time

(4) (input, time) → (n, f(n))

(5) Standards → 1, n, n^2, 2^n, log n, ...

(6) (1) + (5)

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \leq M|g(n)|$ for $n_0 < n$

– To describe an asymptotic upper bound for the magnitude of a function

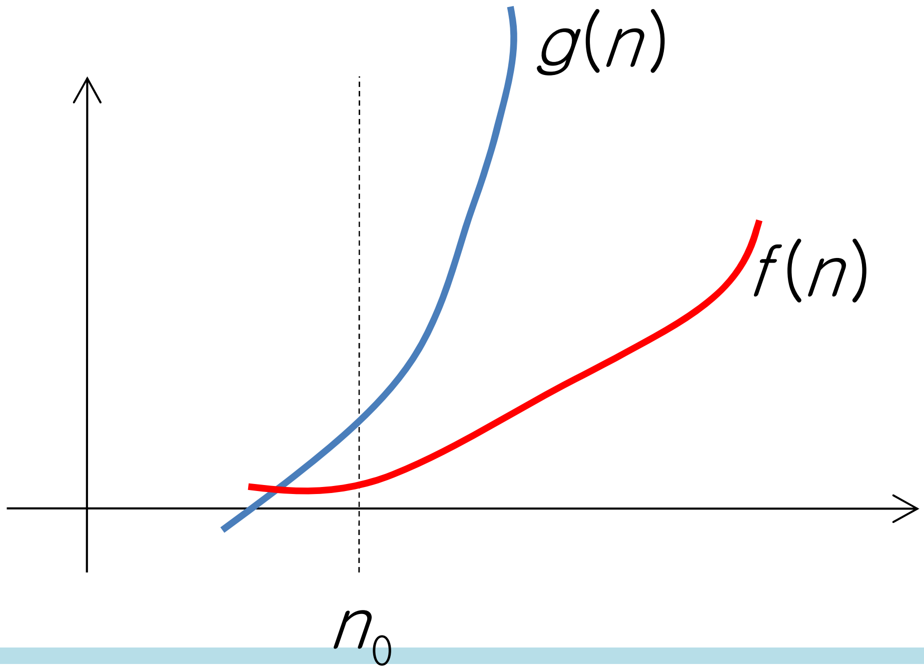
– To characterize a function's behavior for **very large inputs** in a simple but rigorous way that enables comparison to other functions

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– f(n) = O(g(n))



2.3 Big-O Notation

$$f(n) \text{ is } O(g(n)) \text{ as } n \to \infty, \text{ if and only if}$$
$$\exists n_0, \exists M > 0 \text{ such that } |f(n)| \le M|g(n)| \text{ for } n_0 < n$$

– f(n) = O(g(n))
  - For $n > n_0$, f(n) has no chance to be greater than g(n).
  - Suppose f(n) is the time required to execute a function with n inputs.
  - Even at worst case, the function finishes no later than g(n).
  - The upper bound of the time required to finish the function is g(n).
  - The upper bound of f(n) is g(n)

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

- $f(n) = O(g(n))$
  - If f(n) = n, which function of the followings can be g(n)?
    - n
    - n^2
    - n^3
    - n^5
    - e^n

오늘 나온 숙제를 나는 2일이면 다 할 수 있다. 그런데, 교수님은 숙제 기간을 며칠 줄까요?라고 묻는다. 나는 며칠이 필요하다고 해야 할까?

1) 1일
2) 2일
3) 3일
4) 4일
5) 5일

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

- f(n) = O (g(n))
  - f(n) is faster than g(n)
  - g(n) is slower than f(n) → g(n) = Ω (f(n))

- g(n) = Ω (f(n)), if g(n) ≥ M f(n)

A가 3일만에 숙제를 하고 B가 4일만에 숙제를 한다면,
A는 B보다 빠르다 또는 → A = O (B)
B는 A보다 느리다. → B = Ω (A)

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

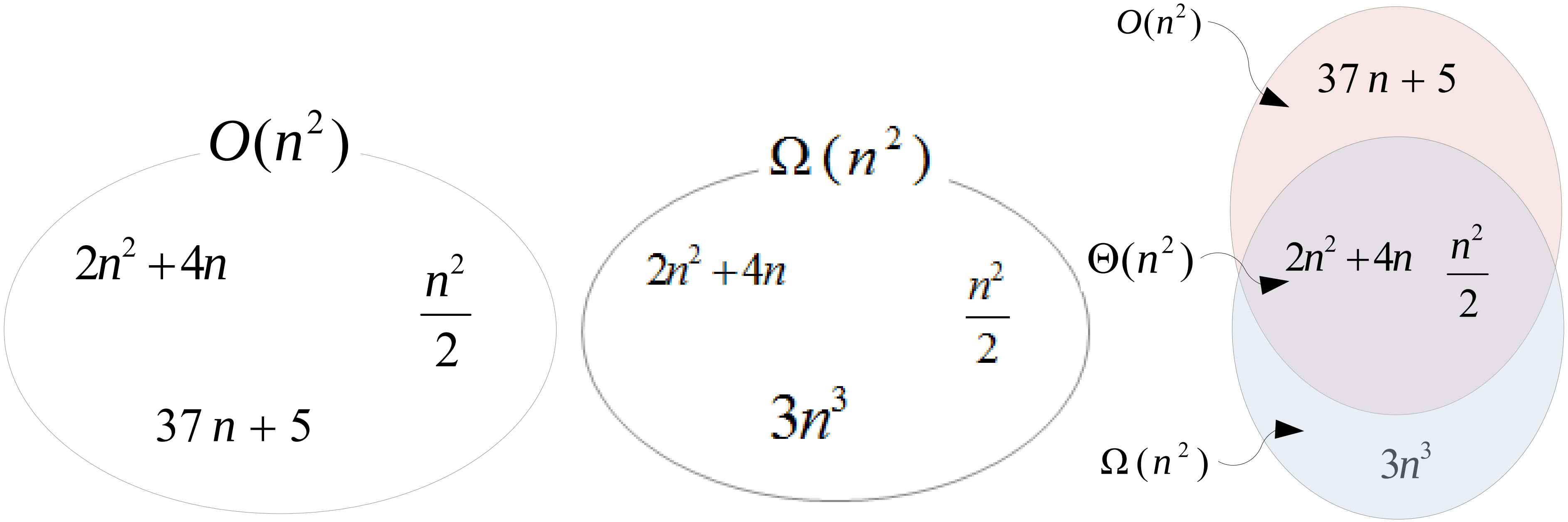
– f(n) = O (g(n)) and f(n) = Ω (g(n))

- f(n) ≤ M g(n) and f(n) ≥ M g(n)
- f(n) = Θ (g(n))

f(n)과 g(n)은 같은 비율로 증가함

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

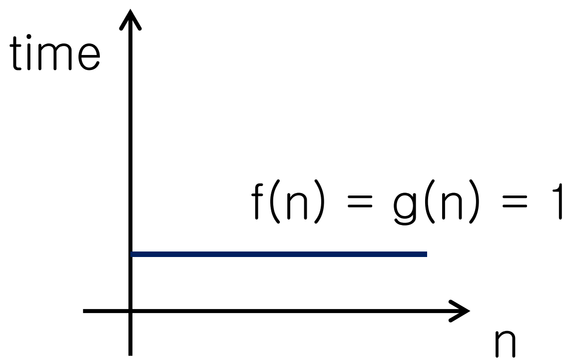


2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 1: g(n) = 1

- f(n) = O(1) → constant time



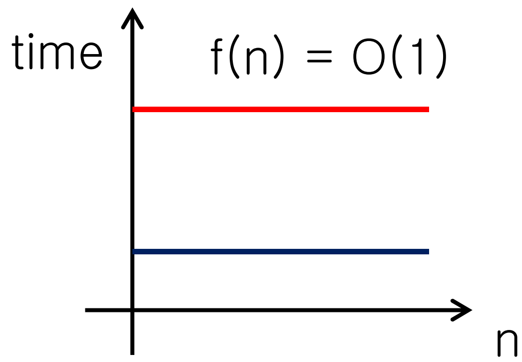
2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 1: g(n) = 1

```
void f ( int n )
{
    printf (“Hello”);
}
```

```
void f ( int n )
{
    printf (“Hello ”);
    printf (“World”);
}
```

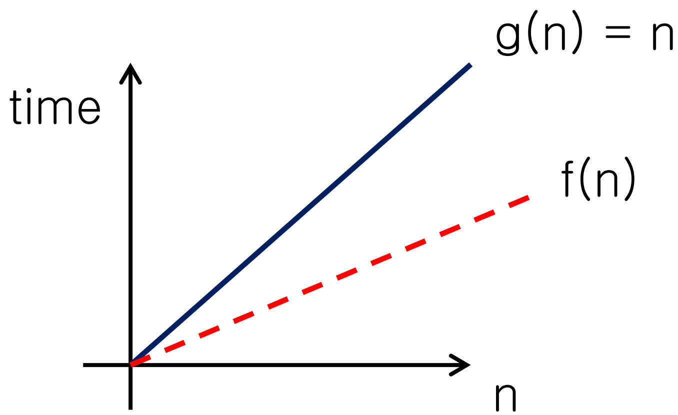


2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 2: g(n) = n

- f(n) = O(n) → linear time



2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 2: g(n) = n

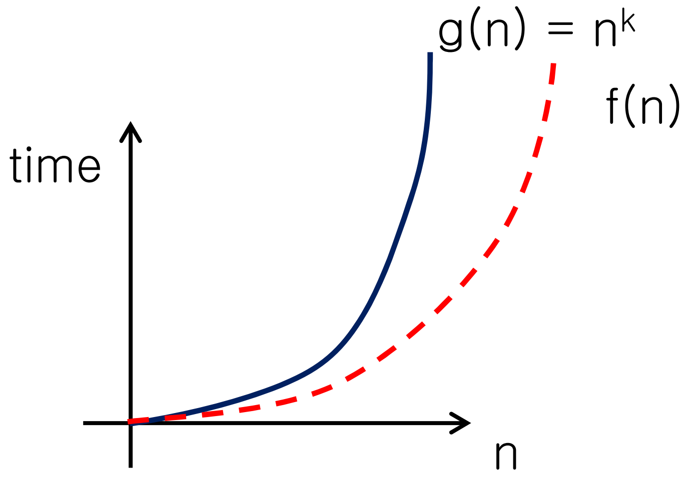
```
i = 0;
while ( i < n ) {
    printf("hello");
    i++;
}
```

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 3: g(n) = n^k

- f(n) = O(n^k) → polynomial time



2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 3: g(n) = n^k

```
for ( i = 0; i < n; i++ ) {
    for ( j = 0; j < n; j++ ) {
        printf("hello");
    }
}
```

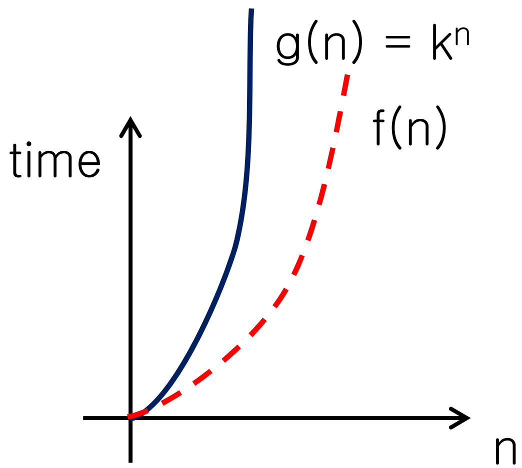
2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 4: $g(n) = k^n$

- $f(n) = O(k^n)$ → exponential time



2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 4: g(n) = k^n

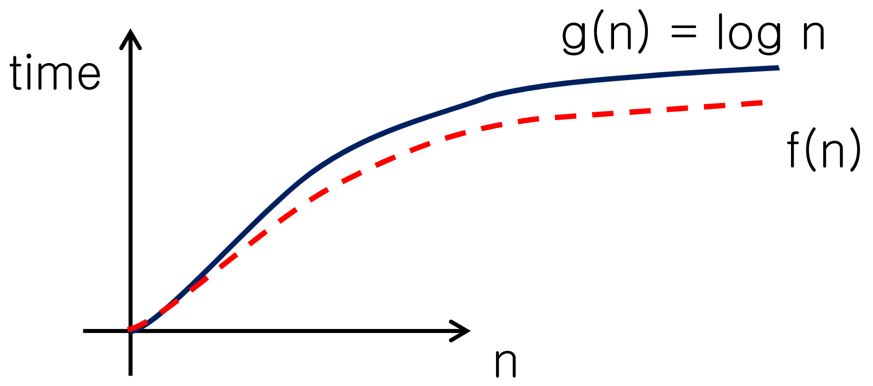
```
int func ( int n )
{
    if ( n == 0 )
        return 0
    if ( n == 1 )
        return 1;

    return func ( n – 1 ) + func ( n – 2 );
}
```

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 5: g(n) = log n

- f(n) = O(log n) → log-n time



2.3 Big-O Notation

$$f(n) \text{ is } O(g(n)) \text{ as } n \to \infty, \text{ if and only if}$$
$$\exists n_0, \exists M > 0 \text{ such that } |f(n)| \leq M|g(n)| \text{ for } n_0 < n$$

– Example 5: g(n) = log n

```
int func ( int n )
{
    for ( k = 1; k < n; k = k * 2 )
        printf("hello");
}
```

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 5: g(n) = log n

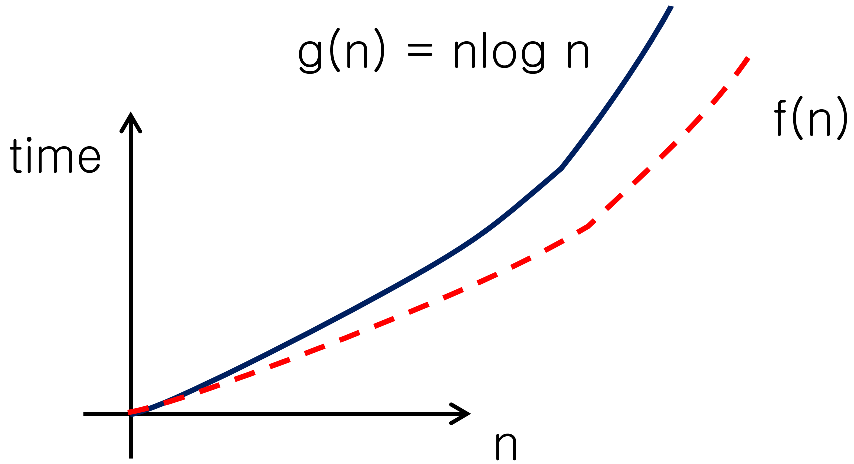
```
int func ( int n )
{
    if ( n == 1 )
        return 1;
    return n * func ( n / 2 );
}
```

2.3 Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n_0 < n$

– Example 6: g(n) = n log n

- f(n) = O(nlog n) → n log-n time



2.3 Big-O Notation

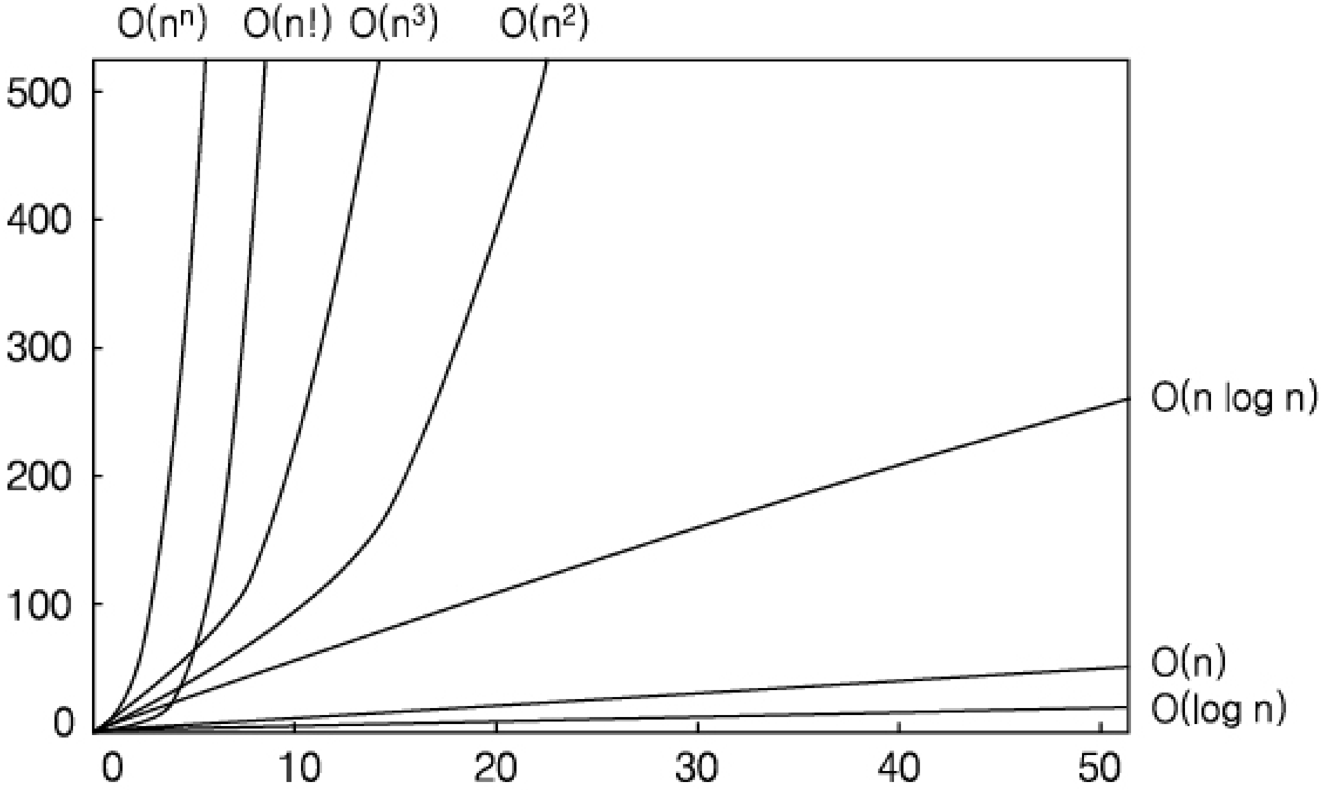
$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \leq M|g(n)|$ for $n_0 < n$

– Example 6: g(n) = n log n

```
void func ( int n )
{
    for ( i = 1; i <= n; i++ ) {
        for ( j = 1; j <= n; j*= 2) {
            print (“hello”);
        }
    }
}
```

2.3 Big-O Notation



2. Analysis

2.1 Performance

2.2 Asymptotic complexity

2.3 Big-O Notation

Contents