2. Prologue

2.1 Introduction

2.2 Computational complexity

2.3 Time complexity of common functions

2.4 Recurrence relation

2.5 Fibonacci

2.3 Time complexity of common functions

- Common functions used to measure and to compare performances

(1) Constant time: $O(1)$
(2) Linear time: $O(n)$
(3) Polynomial time: $O(n^k)$
(4) $O(\log n)$
(5) $O(n \log n)$
(6) NP-complete: $O(k^n)$, $O(n^n)$

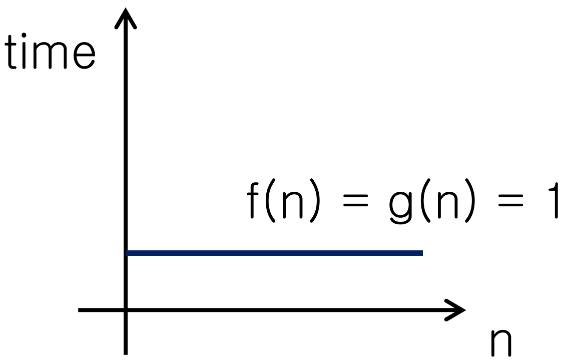
$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^k) < O(k^n) < O(n^n)$$

2.3 Time complexity of common functions

(1) Constant time

– $g(n) = 1$

- $f(n) = O(g(n)) = O(1)$ → constant time



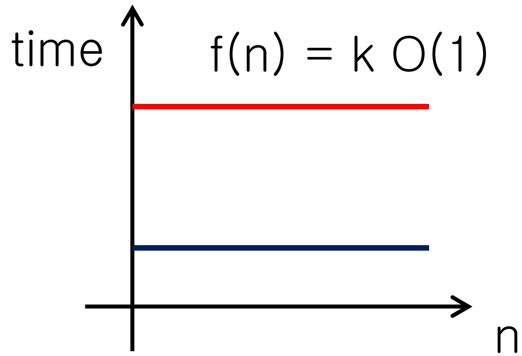
2.3 Time complexity of common functions

(1) Constant time

– $g(n) = 1$

- $f(n) = O(g(n)) = O(1)$ → constant time

```
void print_name (char *name) {
    printf("%s\n", name);
}
```



당신의 이름은?

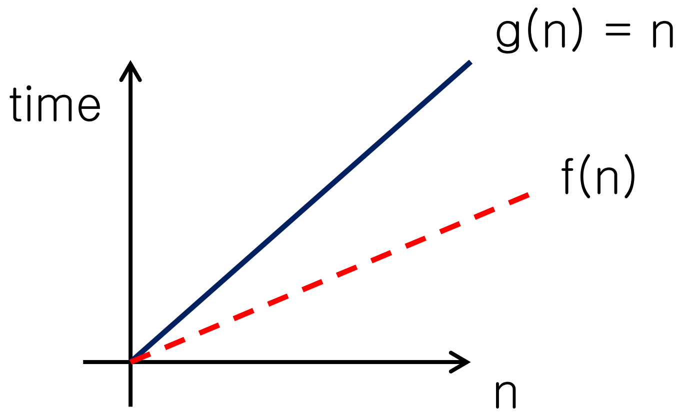
김철

허버트 블레인 볼페슐레겔슈타인하우젠베르거도르프 시니어
https://www.hani.co.kr/arti/specialsection/esc_section/478688.html

2.3 Time complexity of common functions

(2) Linear time

- $g(n) = n$
  - $f(n) = O(g(n)) = O(n) \rightarrow$ linear time



2.3 Time complexity of common functions

(2) Linear time

- $g(n) = n$
  - $f(n) = O(g(n)) = O(n) \rightarrow$ linear time

```
void func ( int n )
{
    for ( i = 0; i < n; i++ )
        Read A[i];
}
```

```
void func ( int n )
{
    for ( i = 0; i < 10*n; i++ )
        Read A[i];
}
```

2.3 Time complexity of common functions

(2) Linear time

– g(n) = n

- f(n) = O(g(n)) = O(n) → linear time

```
void func ( int n )
{
    i = 0;
    while ( i < n ) {
        Read A[i];
        i++;
    }
}
```

2.3 Time complexity of common functions

(2) Linear time

– $g(n) = n$
  - $f(n) = O(g(n)) = O(n) \rightarrow$ linear time
    – $n > m$

```
void func ( int n )
{
    for ( i = 0; i < n; i++ ) {
            Read A[i];
            Write A[i];
    }
    for ( i = 0; i < m; i++ ) {
            Read A[i];
            Write A[i];
    }
}
```

2.3 Time complexity of common functions

(2) Linear time

– g(n) = n

- f(n) = O(g(n)) = O(n) → linear time

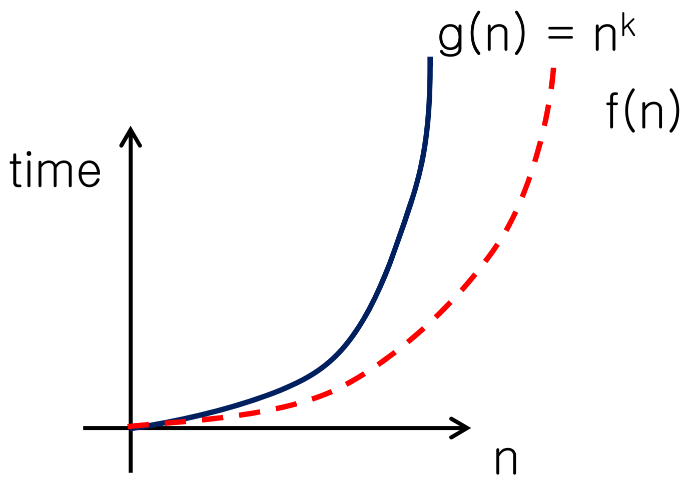
```
void func ( int n )
{
    if ( X is true ) {
        for ( i = 0; i < n; i++ ) {
            Read A[i];
            Write A[i];
        }
    }
    else {
        x = 20;
        printf("%d", x);
    }
}
```

2.3 Time complexity of common functions

(3) Polynomial time

– $g(n) = n^k$

- $f(n) = O(g(n)) = O(n^k)$ → polynomial time



2.3 Time complexity of common functions

(3) Polynomial time

– $g(n) = n^k$

- $f(n) = O(g(n)) = O(n^k) \rightarrow$ polynomial time

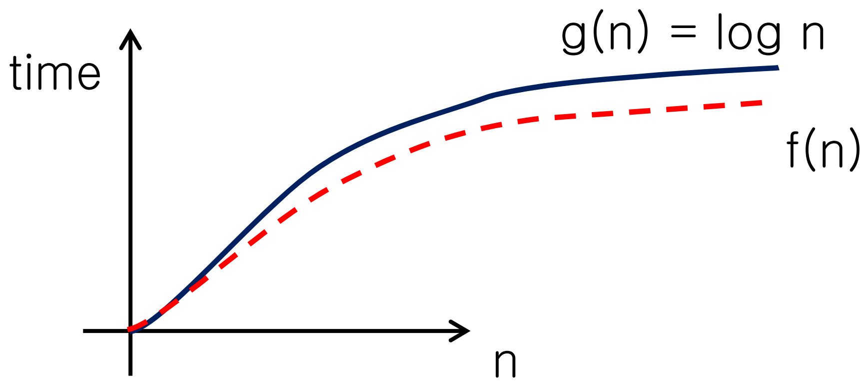
```
void func ( int n )
{
    for ( i = 0; i < n; i++ ) {
        for ( j = 0; j < n; j++ ) {
            Read A[i, j];
            Write A[i, j];
        }
    }
}
```

2.3 Time complexity of common functions

(4) log-n time

– $g(n) = \log n$

- $f(n) = O(g(n)) = O(\log n)$ → log-n time



2.3 Time complexity of common functions

(4) log-n time

– $g(n) = \log n$

- $f(n) = O(g(n)) = O(\log n) \rightarrow$ log-n time

```
void func ( int n )
{
    for ( i = 1; i < n; i *= 10 ) {
        Read A[i, j];
        Write A[i, j]; }
}
```

n	1	10	100	1,000	10,000	100,000	1,000,000	10,000,000	100,000,000
f(n)	1	2	3	4	5	6	7	8	9

2.3 Time complexity of common functions

(4) log-n time

– $g(n) = \log n$

- $f(n) = O(g(n)) = O(\log n) \rightarrow$ log-n time

```
int BS( int x, int n, int S[] ) {
    if ( n == 1 )
        return (S[0] == x);
    if ( x > S[n/2] )
        return BS( x, n/2, S[n/2+1, …, n-1] );
    else
        return BS( x, n/2, S[0, …, n/2] );
}
```

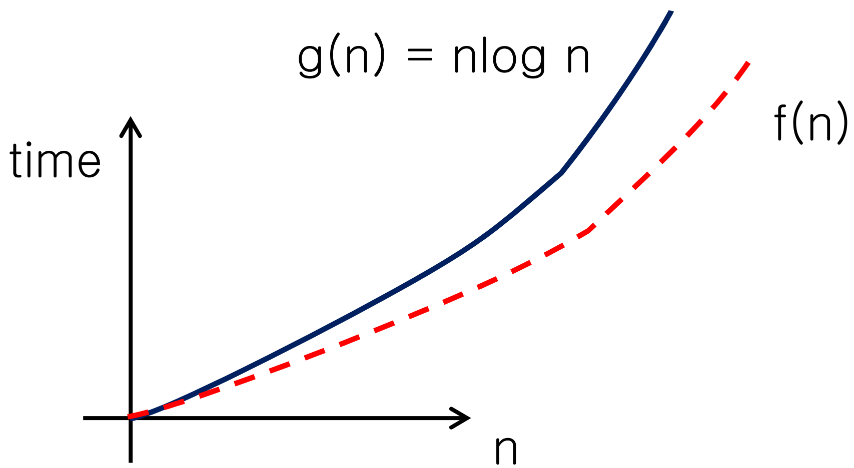
- $f(n) = f(n/2) + 1$

2.3 Time complexity of common functions

(5) n log-n time

– g(n) = n log n

- f(n) = O(g(n)) = O(nlog n) → n log-n time



2.3 Time complexity of common functions

(5) n log-n time

– g(n) = n log n

- f(n) = O(g(n)) = O(nlog n) → n log-n time

```
void func ( int n )
{
    for ( i = 1; i <= n; i++ ) {
        for ( j = 1; j <= n; j*= 2) {
            Read A[i, j];
            Write A[i, j];
        }
    }
}
```

2.3 Time complexity of common functions

(5) n log-n time

– $g(n) = n \log n$

- $f(n) = O(g(n)) = O(n\log n)$ → n log-n time

```
void MergeSort( int n, int S[] ) {
    if ( n > 1 ) {
        MergeSort (n/2, S[0, …, n/2 - 1]);
        MergeSort (n/2, S[n/2, …, n - 1]);
        Merge ( n, S );
    }
}
```

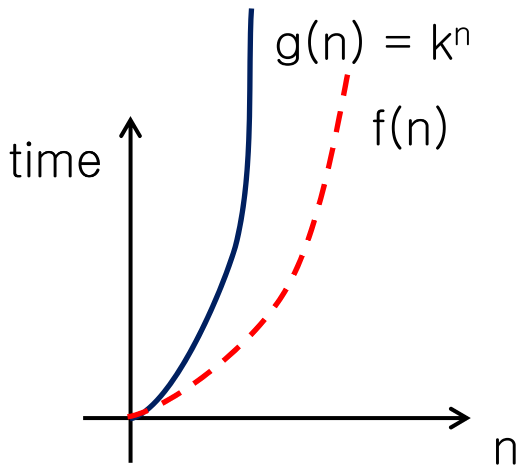
- $f(n) = 2 f(n/2) + n$

2.3 Time complexity of common functions

(6) Exponential time

– $g(n) = k^n$

- $f(n) = O(k^n)$ → exponential time



2.3 Time complexity of common functions

(6) Exponential time

– $g(n) = k^n$

- $f(n) = O(g(n)) = O(k^n)$ → exponential time

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

    return Fib (n-1) + Fib (n-2);
}
```

- $f(n) = f(n-1) + f(n-2)$

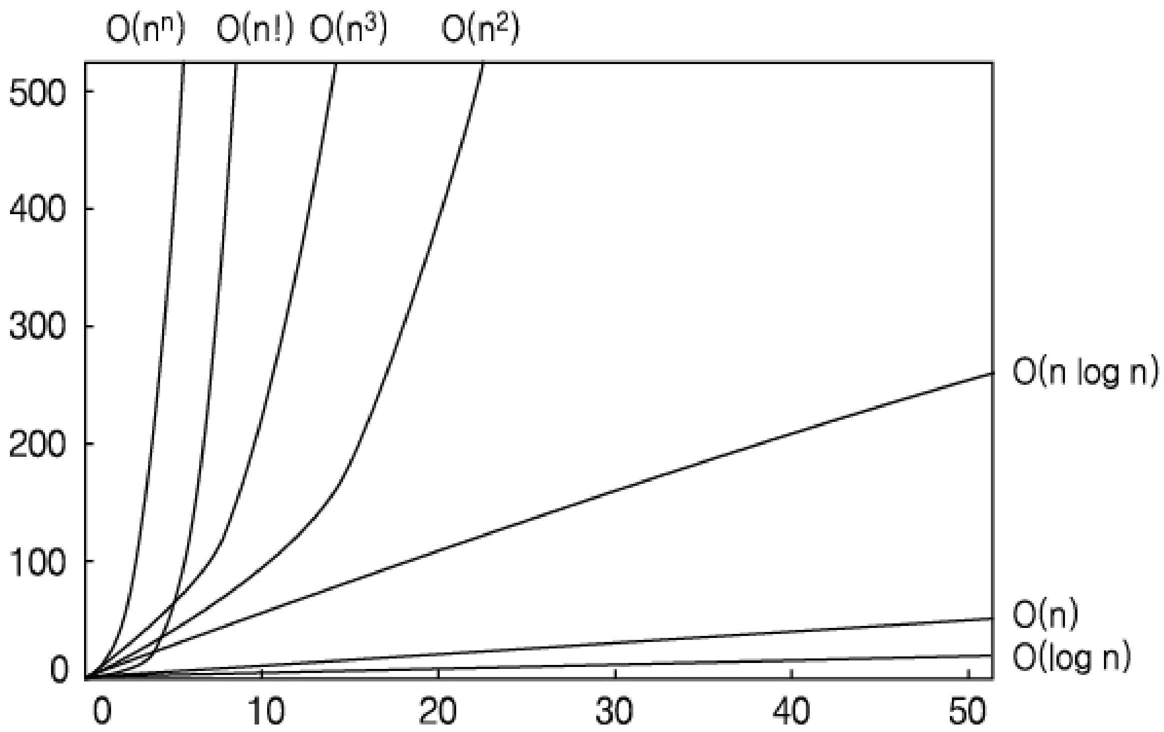
2.3 Time complexity of common functions

(7) NP-complete

- $g(n) = k^n$
- $g(n) = n^n$
- $g(n) = n!$

2.3 Time complexity of common functions

(8) Comparison



Quiz3

What is the time complexity for this code?

```
void function ( int n )
{
    int i, count = 0;
    for ( i = 1; i*i <= n; i++ )
        count++;
}
```

Quiz4

What is the time complexity for this code?

```
void function ( int n )
{
    int i, j, count = 0;
    for ( i = n/2; i <= n; i++ )
        for ( j = 1; j <= n; j *= 2 )
            count++;
}
```

Quiz5

What is the time complexity for this code?

```
void function ( int n )
{
    if ( n < 3 )
        return;

    for ( int i = 1; i <= n; i++ )
        for ( int j = 1; j <= n; j++ )
            printf (“*”);

    function ( n – 3 );
}
```