Complexity of an Algorithm

By
Dr. Nuttachot Promrit

```
char h = 'y'; // This will be executed 1 time
int abc = 0; // This will be executed 1 time

for (int i = 0; i < N; i++) {
        cout << “Hello World”;
}
```

int i = 0 → 1

i < N → N + 1

i++ → N

cout << “Hello World”; → N

1 + N + 1 + N + N = 3N +2

```
for ( i = 0; i < N; i++ )
    statement;
```

N

```
for ( i = 0; i < N; i++ ) {
    for ( j = 0; j < N; j++ )
        statement;
}
```

N*N

```
while ( low <= high ) {
        mid = ( low + high ) / 2;
        if ( target < list[mid] )
                high = mid - 1;
        else if ( target > list[mid] )
                low = mid + 1;
        else break;
}
```

1 100000

2 50000

3 25000

4 12,500

5 6,250

6 3,125

7 1,562

8 781

9 390

10 195

11 97

12 48

13 24

14 12

15 6

16 3

17 1

2 4 8 16 32 64 128 256 512 1024 2048 4096 8,192 16,384 32,768 65,536 131,072

Order of increasing complexity

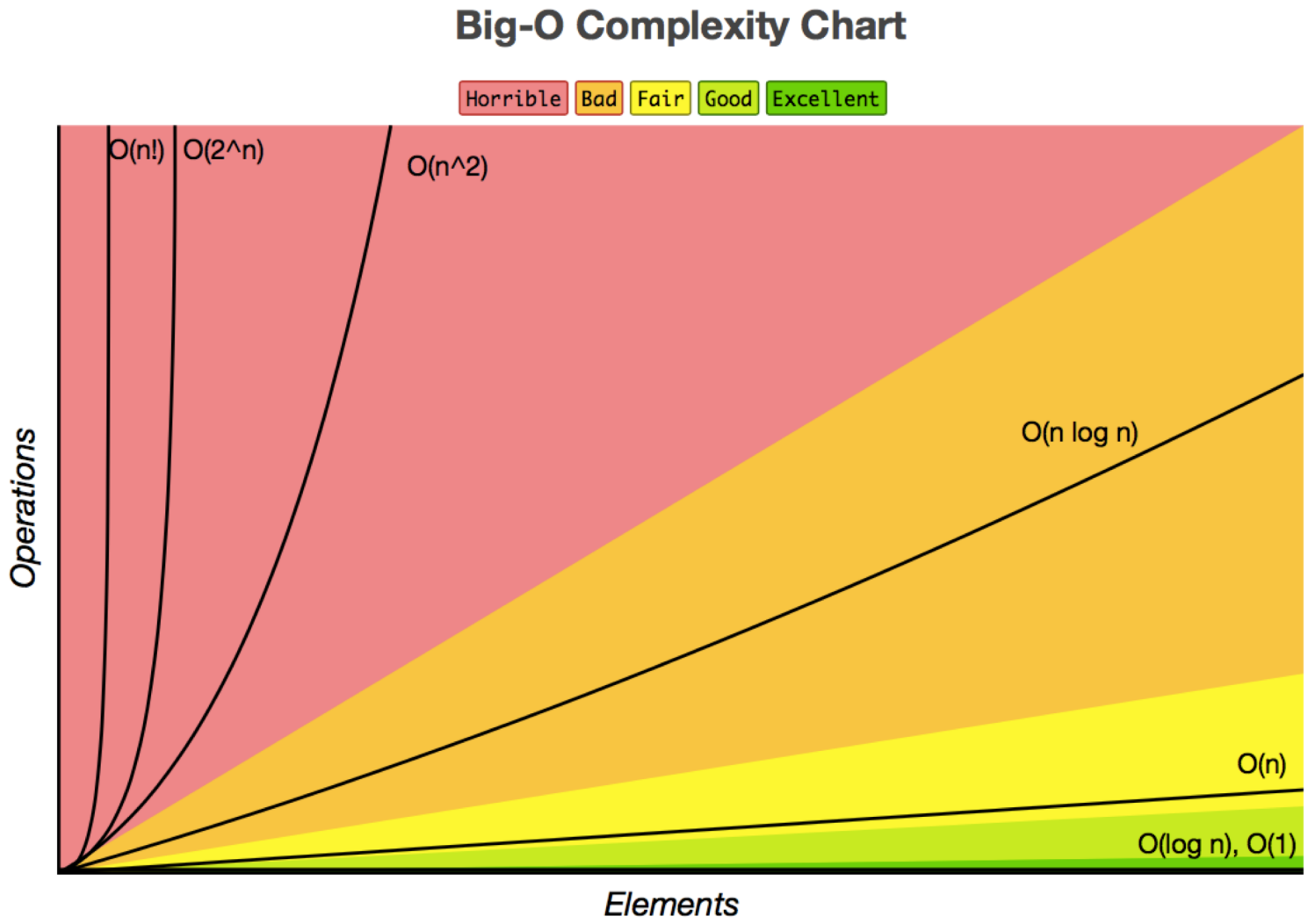
Order of growth for some common function:

- $O(1) < O(\log_x n) < O(n) < O(n \log_2 n) < O(n^2) < O(n^3) < O(2^n)$

Notasi	n = 8	n = 16	n = 32
$O(\log_2 n)$	3	4	5
$O(n)$	8	16	32
$O(n \log_2 n)$	24	64	160
$O(n^2)$	64	256	1024
$O(n^3)$	512	4096	32768
$O(2^n)$	256	65536	4294967296

```
while ( low <= high ) {
        mid = ( low + high ) / 2;
        if ( target < list[mid] )
                high = mid - 1;
        else if ( target > list[mid] )
                low = mid + 1;
        else break;
}
```

Log (N)



Class	Name	Example
1	constant	access array element
$\log n$	logarithmic	binary search
n	linear	find median
$n \log n$	"n-log-n"	mergesort
n^2	quadratic	insertion sort
n^3	cubic	matrix multiplication
a^n	exponential	generating all subsets
$n!$	factorial	generating all permutations

Pic from http://bigocheatsheet.com

Big *O* Notation

- Example of algorithm for common function:

O(n) Linear	`int counter = 1; int i = 0;`
	
	`for (i = 1; i <= n; i++) {`
	`  cout << "Arahan cout kali ke " << counter << "\n";`
	`  counter++;`
	`}`

O(n $\log_x$n) Linear Logarithmic	`int counter = 1; int i = 0; int j = 1;`
	
	`for (i = x; i <= n; i = i * x) { // x must be > than 1`
	`  while (j <= n) {`
	`  cout << "Arahan cout kali ke " << counter << "\n";`
	`  counter++; j++;`
	`  }`
	`}`

Big *O* Notation

- Example of algorithm for common function:

$O(n^2)$ Quadratic	`int counter = 1;`
	`int i = 0;`
	`int j = 0;`
	
	`for (i = 1; i <= n; i++) {`
	`  for (j = 1; j <= n; j++) {`
	`  cout << "Arahan cout kali ke " << counter << "\n";`
	`  counter++;`
	`  }`
	`}`

Big *O* Notation

- Example of algorithm for common function:

$O(n^3)$
Cubic
	`int counter = 1;`
	`int i = 0;`
	`int j = 0;`
	`int k = 0;`
	
	`for (i = 1; i <= n; i++) {`
	`for (j = 1; j <= n; j++) {`
	`for (j = 1; j <= n; j++) {`
	`cout << "Arahan cout kali ke " << counter << "\n";`
	`counter++;`
	`}`
	`}`
	`}`

Big *O* Notation

- Example of algorithm for common function:

$O(2^n)$ Exponential	`int counter = 1;`
	`int i = 1;`
	`int j = 1;`
	
	`while (i <= n) {`
	`  j = j * 2;`
	`  i++;`
	`}`
	
	`for (i = 1; i <= j; i++) {`
	`  cout << "Arahan cout kali ke " << counter << "\n";`
	`  counter++;`
	`}`