

Examples:  

$$f(n) = \sqrt{n} \qquad g(n) = 2n$$

$$f(n) = \frac{\sqrt{n}}{2n} = \frac{1}{2\sqrt{n}} \xrightarrow{n \to \infty} 0$$

$$5n^2 - 4n - 100 \quad g(n) = n^2$$

$$\frac{f(n)}{g(n)} = \frac{5n^2 - 4n - 100}{n^2} = 5 - \frac{4}{n} - \frac{100}{n^2} \xrightarrow{n \to \infty} 5$$

$$F(n) = \sqrt[3]{n} \qquad g(n) = \sqrt{n}$$

$$\frac{f(n)}{g(n)} = \frac{\sqrt[3]{n}}{\sqrt{n}} = \frac{n^{\frac{1}{2}}}{n^{\frac{1}{2}}} = n^{\frac{1}{2} - \frac{1}{2}} = n^{-\frac{1}{6}} = \frac{1}{n^{\frac{1}{6}}} \xrightarrow{n \to \infty} 0$$

$$f(n) = n^{2} \qquad g(n) = n \log n$$

$$\frac{f(n)}{g(n)} = \frac{n^{2}}{n \log n} = \frac{n}{\log n} \xrightarrow{n \to \infty} \infty$$

# Analysis of Time Complexity

- (1) Determine the **input size** (*n*)
- (2) Determine the **basic operations**
- (3) Let  $m{c}(m{n})$  be the maximum count of the basic operations as function of  $m{n}$
- (4) Let  $oldsymbol{d}(oldsymbol{n})$  be the minimum count of the basic operations as function of  $oldsymbol{n}$
- (5) The **upper bound** of the time complexity is O(c(n))
- (6) The **lower bound** of the time complexity is  $\Omega(d(n))$
- (7) If  $O(c(n)) = \Omega(d(n))$ , then the **exact bound** of the time complexity is  $\Theta(c(n))$

### Main Rules of Asymptotic Notations

- 1. Drop constant factors
  - $\checkmark \quad 6n-3=0(n)$
  - ✓  $2n^2 + 1000 = O(n^2)$
  - $\checkmark \quad 4n\log n + 10 = O(n\log n)$
- 2. Drop lower-order terms
  - ✓  $n^3 + n^2 + n + 1 = O(n^3)$
  - $\checkmark \quad n + \log n = O(n)$
  - $\checkmark \quad n\log n + n = O(n\log n)$
  - $\checkmark \quad \log n + \log \log n = O(\log n)$

# $\bigcirc$ Big Oh Rules:

- **1**. Ignore constant factors.
- **2**. IF we have 2 functions f1(n), f2(n) and f1(n) = O(g1(n)), f2(n) = O(g2(n)) then

f1(n) \* f2(n) = O (g1(n) \* g2(n)).

Ex:  $f1(n) = O(n^2)$  and f2(n) = O(n) $f1(n) * f2(n) = O(n^2 * n)$  $= O(n^3)$ 

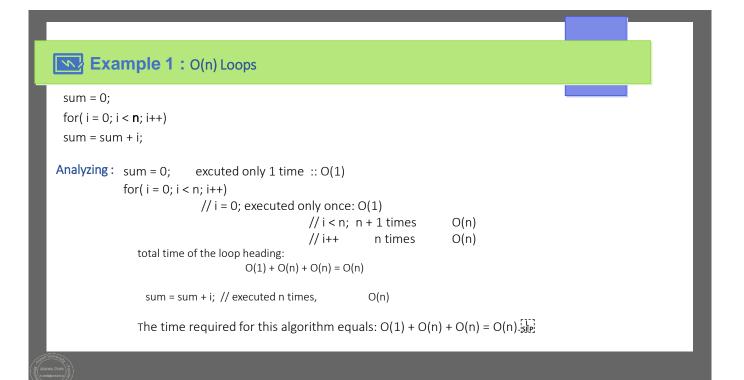
# Analysis of Time Complexity

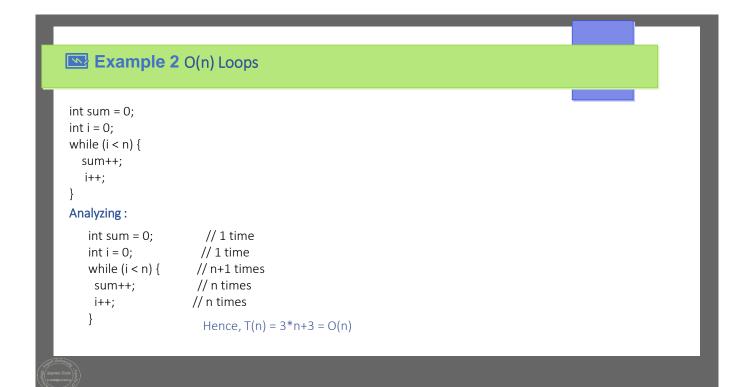
#### Counting the Number of Operations

- 1. The running time equals the number of primitive operations (steps) executed before termination.
- 2. Each operation takes a certain time.

### Analysis of Loops:

• Simple Loops: The running time of a for loop is at most the running time of the statements inside the loop times the number of iterations.





### Example 3 O(1) Loops

A loop or recursion that runs a constant number of times is considered as O(1).

```
Int sum = 0;
for (int i = 1; i <= 10; i++) {
    sum = sum + a[i]
}</pre>
```

#### • Nested Loop:

Time complexity of nested loops is equal to the number of times the innermost statement is executed.

#### Example 4 O(n<sup>2</sup>) Loops

```
sum = 0;
for( i = 0; i < n; i++)
for( j = 0; j < n; j++)
```

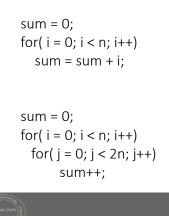
sum++;

The running time =  $O(1) + O(n^*n) + O(n)$ = $O(1) + O(n^2) + O(n)$ =  $O(n^2)$ 

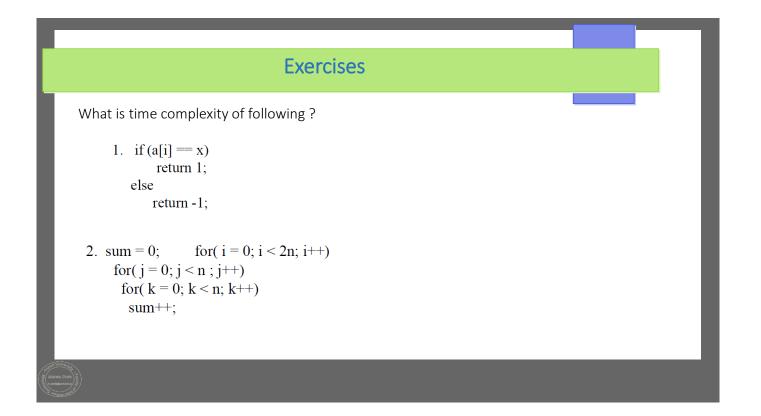
## • Consecutive program fragments

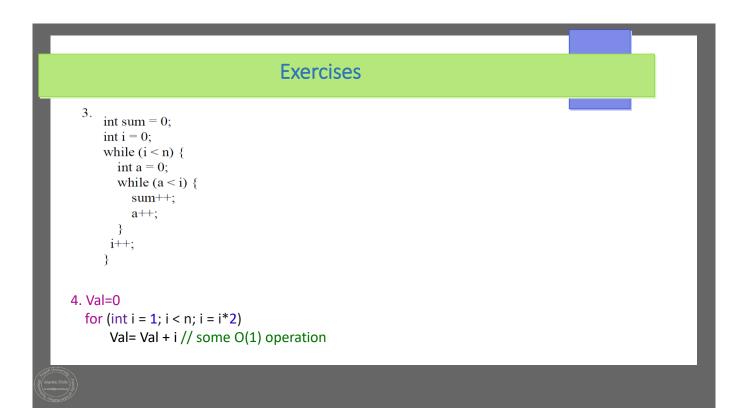
The total running time is the maximum of the running time of the individual fragments

#### Example 5 O(n<sup>2</sup>) Loops



○ If statement	
IF Condition	
S1;	
else	
S2;	
The running time is the maximum of the running times of S1 and S2.	





# Exercises

#### What is time complexity of fun()?

