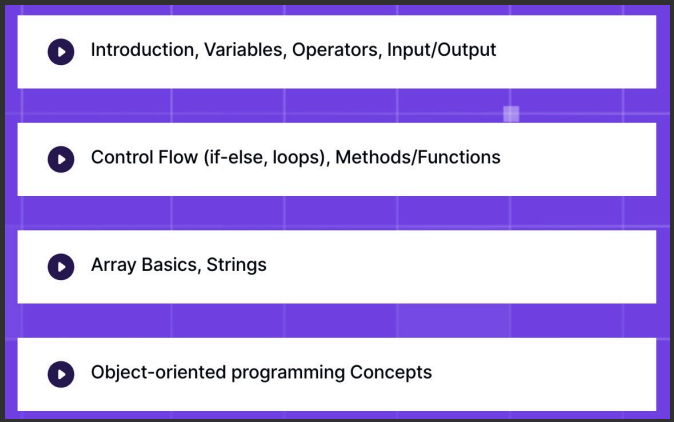
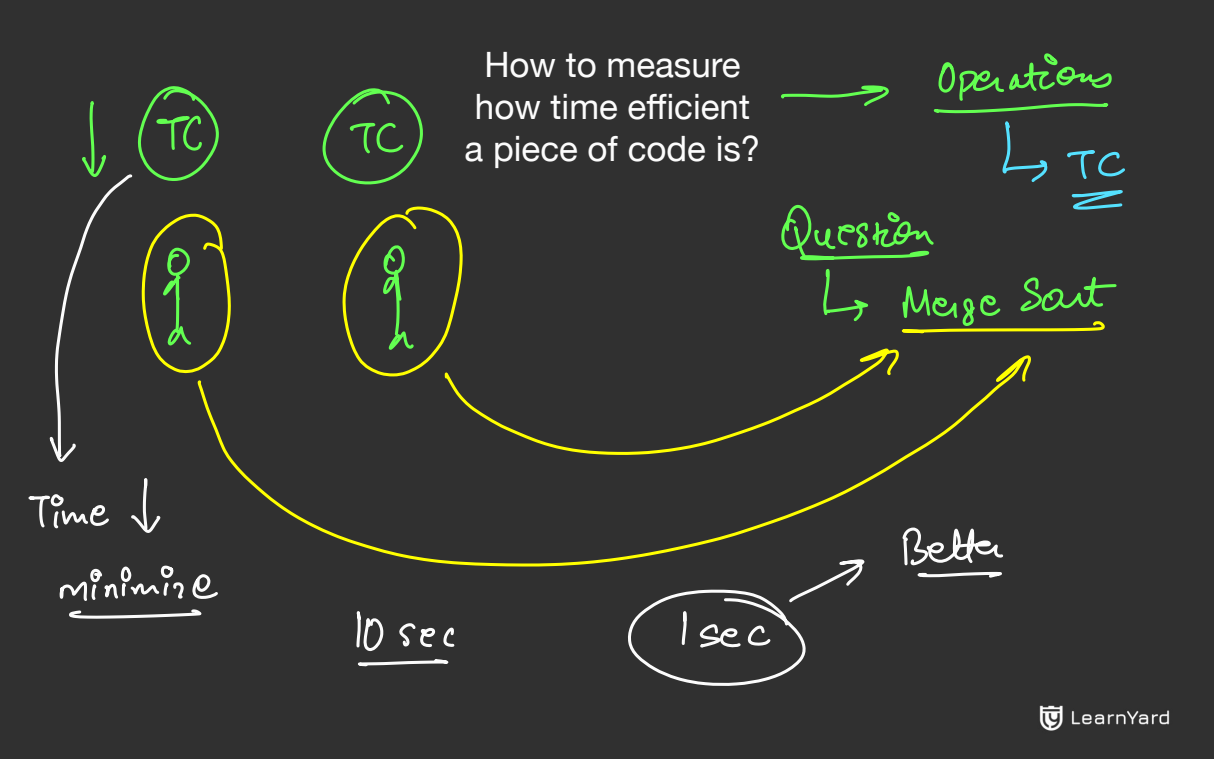
L13
Time & Space Complexity

For discord mail to support@learnyard.com

Recap


Introduction, Variables, Operators, Input/Output
Control Flow (if-else, loops), Methods/Functions
Array Basics, Strings
Object-oriented programming Concepts


How to measure how time efficient a piece of code is?
Operations
TC
TC
TC
Question
Merge Sort
Time
minimize
10 sec
1 sec
Better

100 GB RAM → < 3.12

3.12 sec (16 GB)

↳ RAM

```java
import java.io.*;
import java.util.*;

class PalindromeNumber {

    1 usage
    static boolean isPal(int n)
    {
        int ReversedNumber = 0;

        int temp = n;
        while(temp != 0)
        {
            int LastDigit = temp % 10;

            ReversedNumber = ReversedNumber * 10 + LastDigit;

            temp = temp / 10;
        }

        return ReversedNumber==n;
    }

    public static void main (String[] args) {

        int number = 9559;

        System.out.println(isPal(number));

    }
}
```

Palindrome

121 (121)

RUN

3 min

10 min

2 GB → 5.12 sec

20 GB → 1.2 sec

LearnYard

But, the time taken also depends on:

never compare

Device

Programming Language

C++ >>

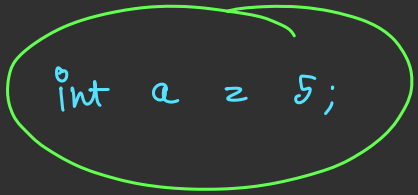
```
Enter a number: 5
Factorial of N = 5 is 120.
logout
```

Java

Input(s) given

Then, what's the right way to measure time complexity?

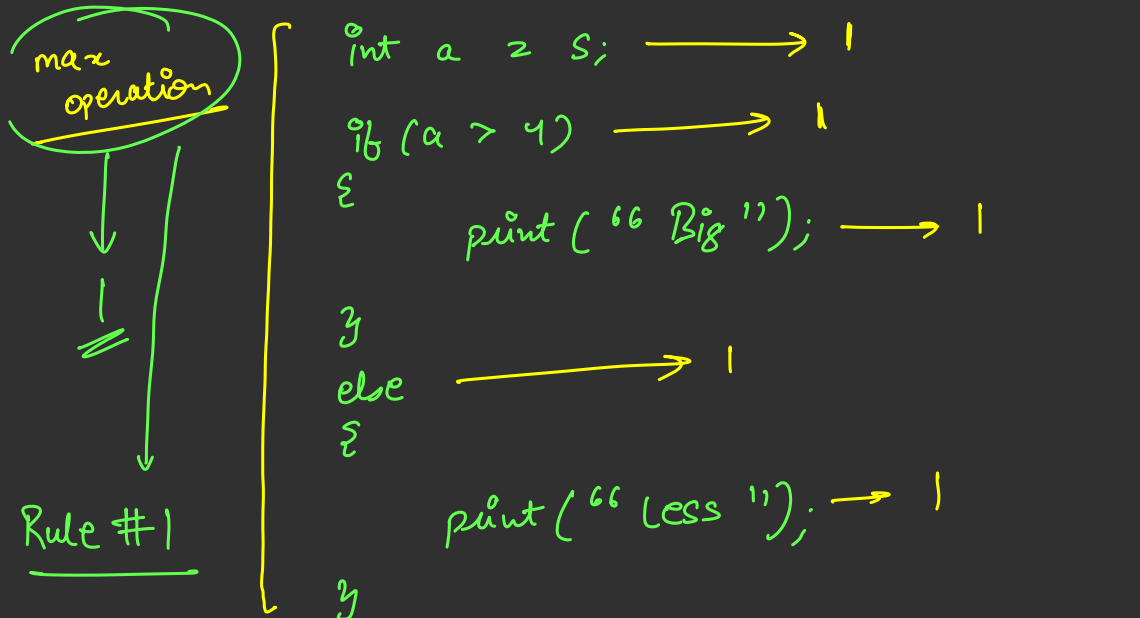
→ Operations



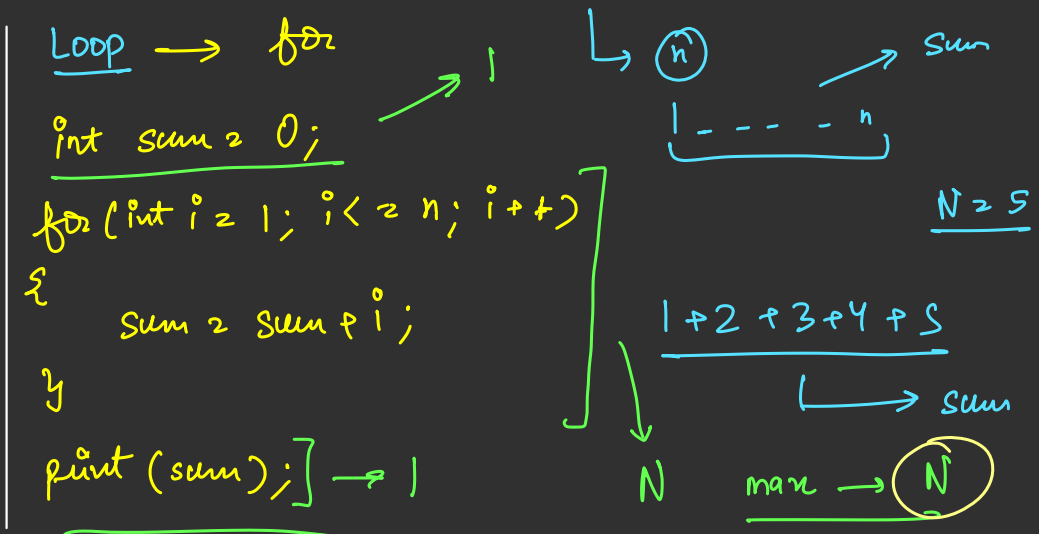
→ 1 operation

cout / System.out.println

→ 1 single operation


max operation
1
Rule #1
int a = 5; → 1
if (a > 4) → 1
{
print ("Big"); → 1
}
else → 1
{
print ("Less"); → 1
}

Example - Finding the sum of 1st N natural numbers



Example - Finding the sum of 1st N natural numbers

formula → op = 1

```
int sum = (n * (n+1)) / 2;
print(sum);
```

n → Sum

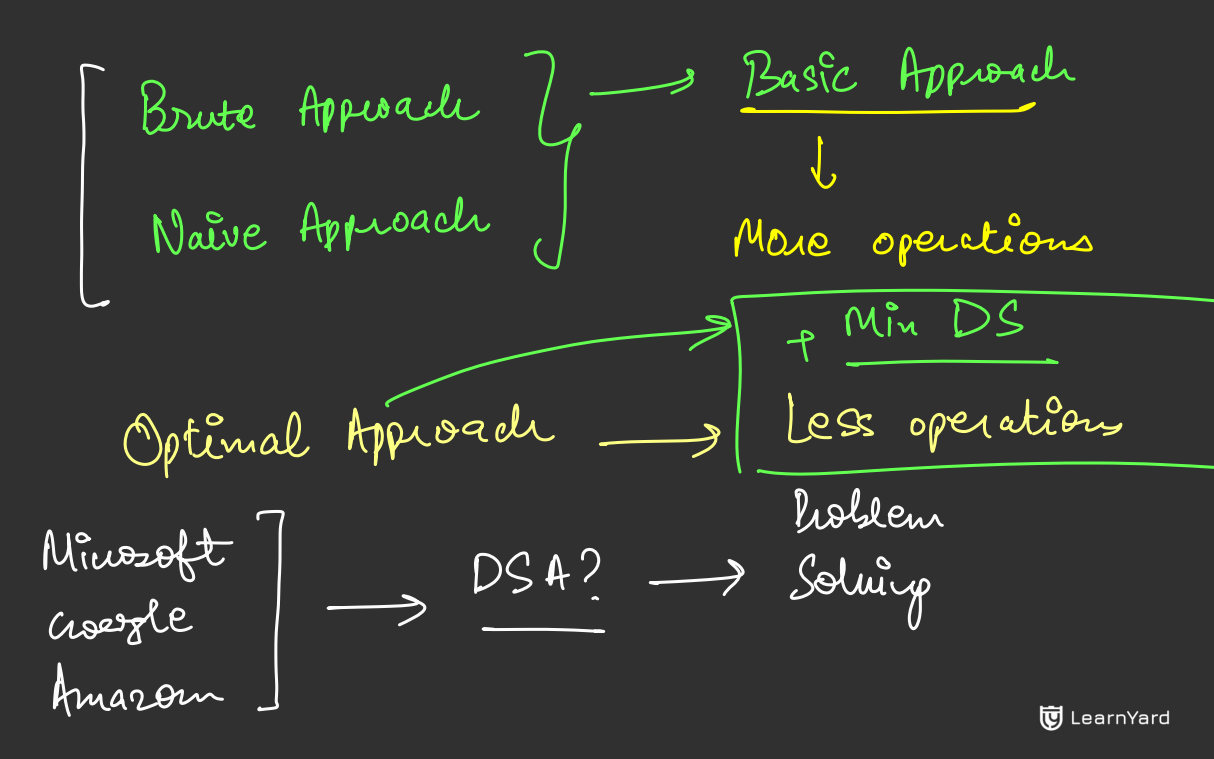
1 - - - - n

N = 5

1 + 2 + 3 + 4 + 5 → sum

max operation → 1

$$\frac{n * (n+1)}{2}$$ → Sum of N natural number


Brute Approach
Naive Approach
Basic Approach
More operations
Optimal Approach
+ Min DS
Less operations
Microsoft
Google
Amazon
DSA?
Problem Solving

Formal Definition

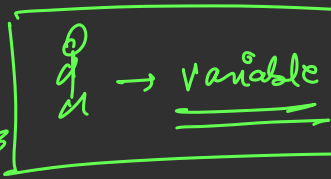
Time complexity is a measure of how the running time of an algorithm grows as the size of the input increases.

#1 → Max operation

#2 → ignore constant

Space Complexity → Data Structure

What about Space Complexity?



73 bytes?

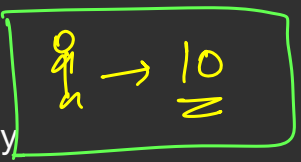
100 KB?

24 MB?

Data structure

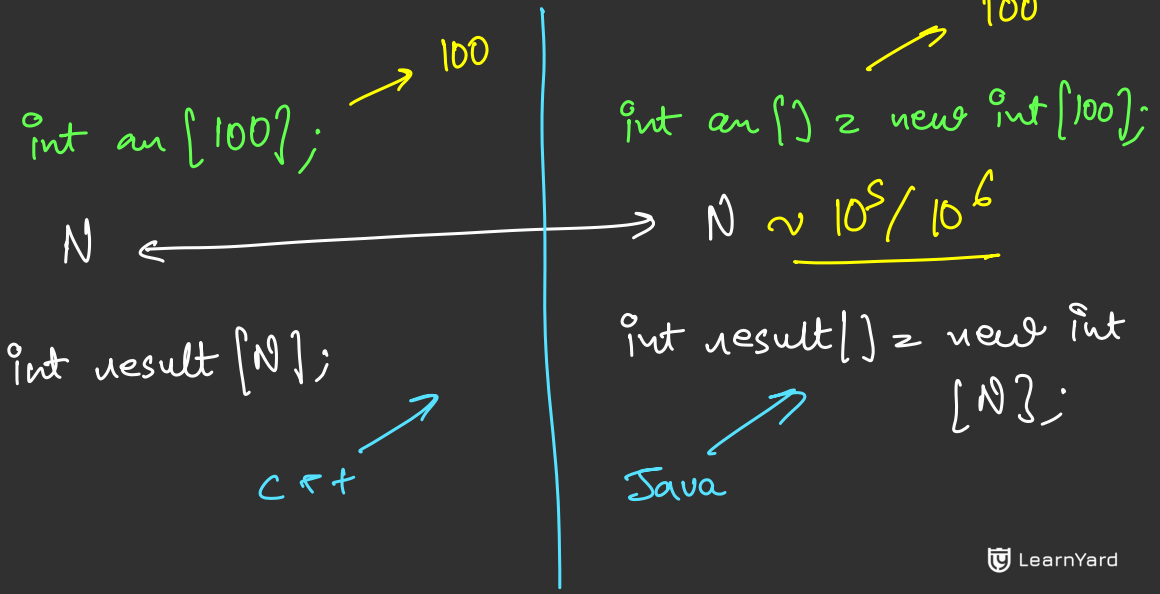
↓

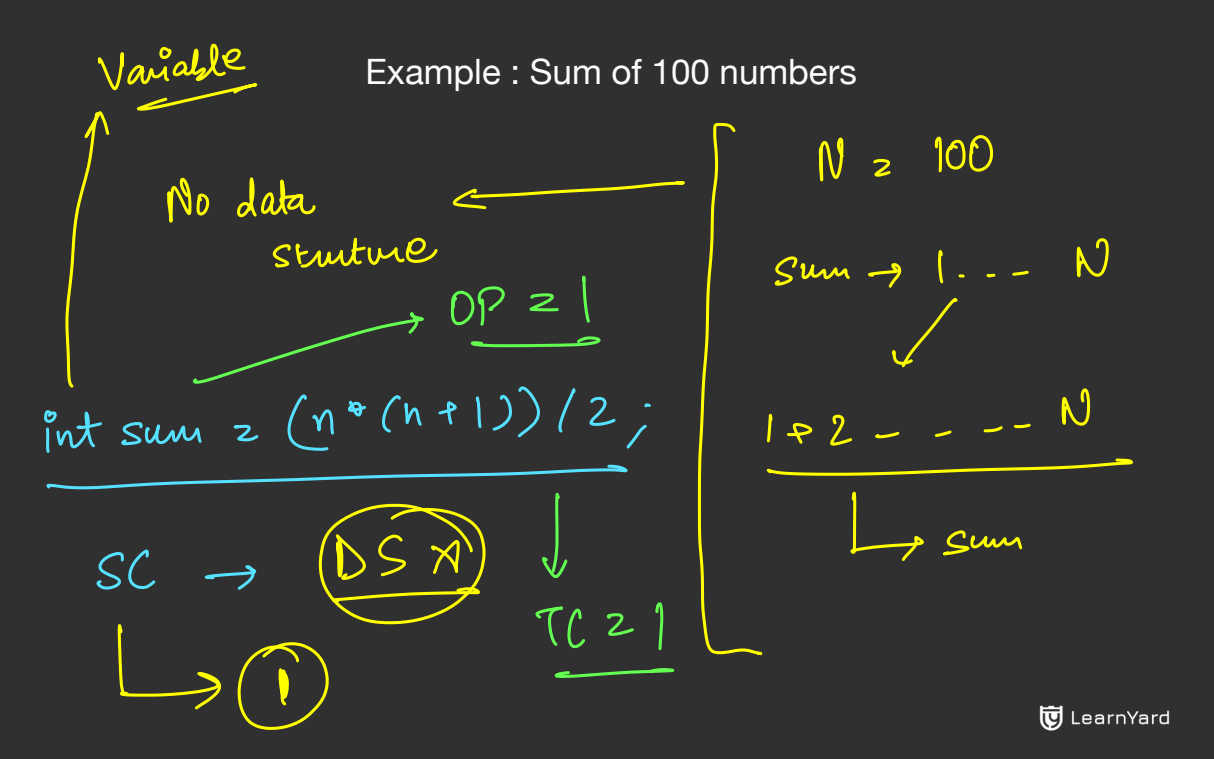
number of elements



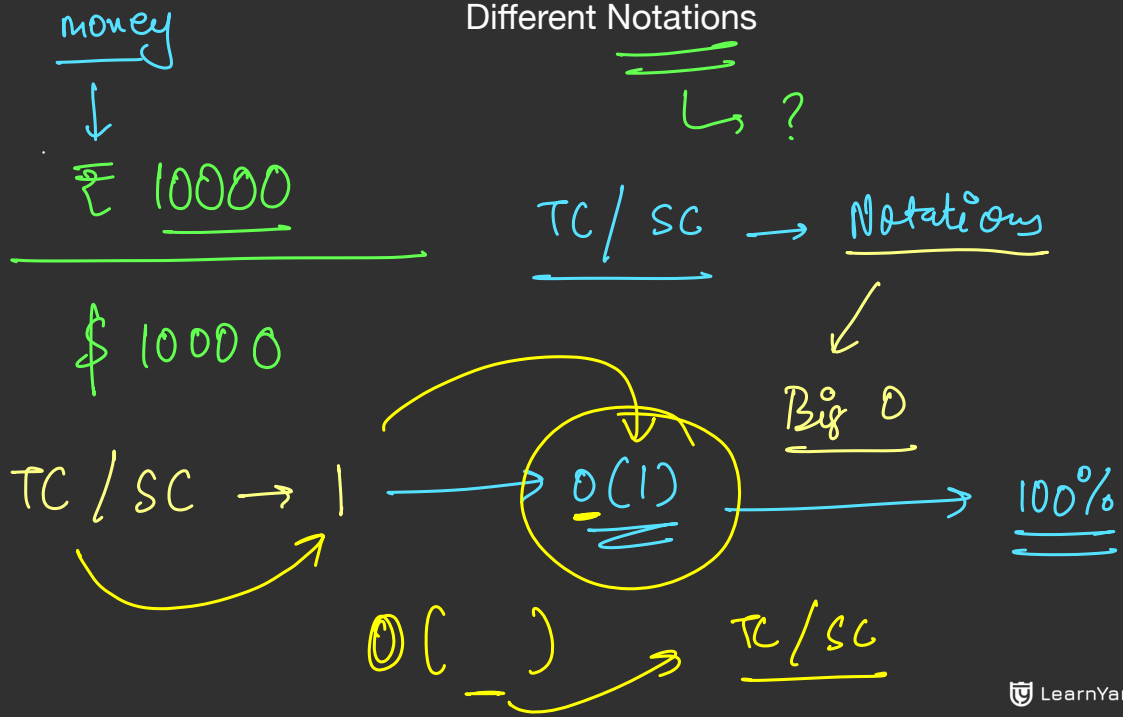
Similar to TC, this is not the right way

Space complexity measures how the memory/space required by an algorithm grows as the size of the input increases.




Variable
Example : Sum of 100 numbers
No data
stuture
OP = 1
int sum = (n*(n+1))/2;
SC → DSA
1
TC = 1
N = 100
Sum → 1 ... N
1+2 - - - - N
Sum

Different Notations



N times

```
for (int i = 1; i <= n; i++)
{
    for (int j = 1; j <= i; j++)
    {
        print(j);
    }
}
```

1, 2, 3 ... N

Total - $\frac{N * (N+1)}{2}$

$1 + 2 + 3 + 4 + \dots + N$

Here are some common time complexities (ordered best to worse):

1. O(1): Constant time complexity
2. O(log n): Logarithmic time complexity
3. O(n): Linear time complexity
4. O(nlogn): Linearithmic time complexity
5. $O(n^2)$: Quadratic time complexity
6. $O(2^n)$: Exponential time complexity → Recursion

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)$

$\rightarrow O(1)$

$\rightarrow O(\log N)$

$N \leq 10^6$

Class 8

↳ log

$\log_{10} 10^6 \longrightarrow 6$

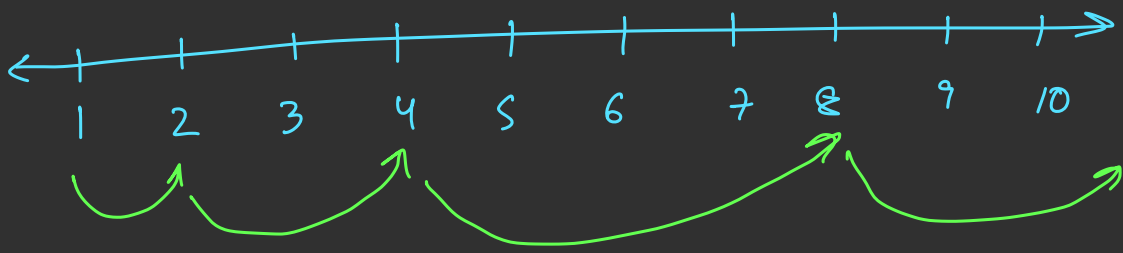
$\log_a a^b \longrightarrow b$

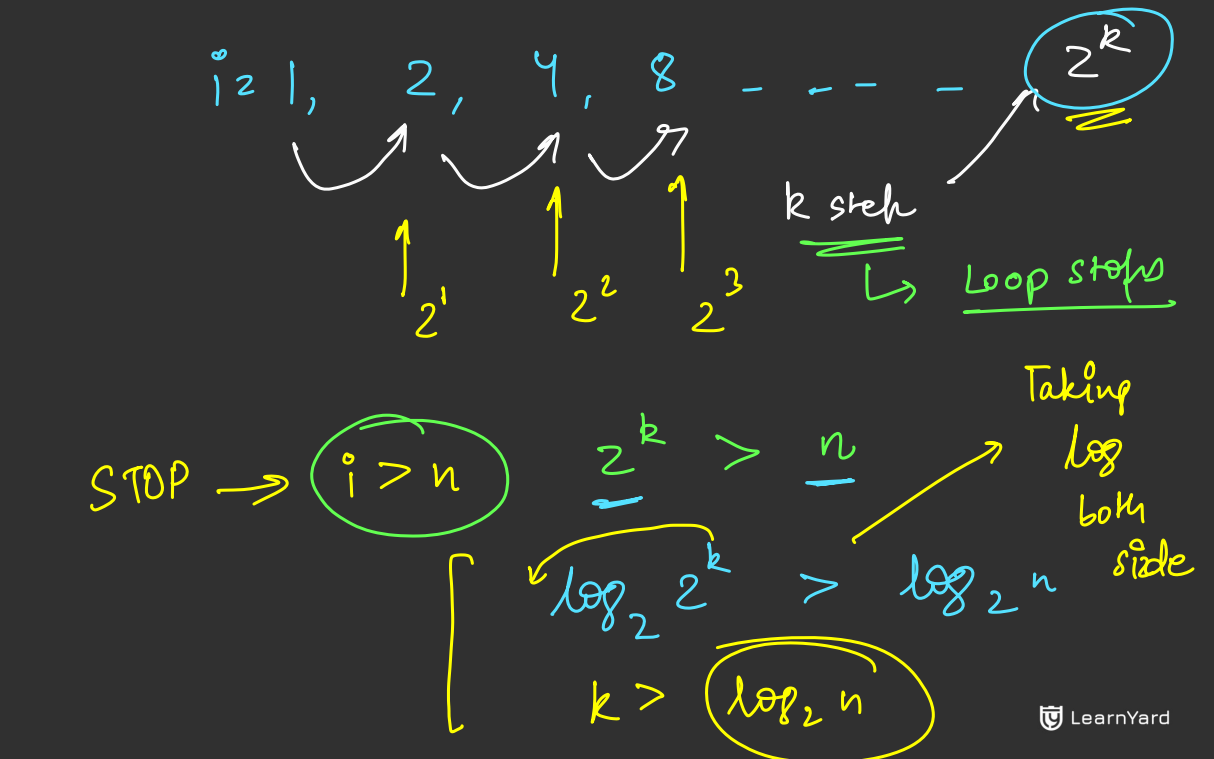
```
for (int i = 1; i <= n; i = i * 2)
{
    print (i);
}
```

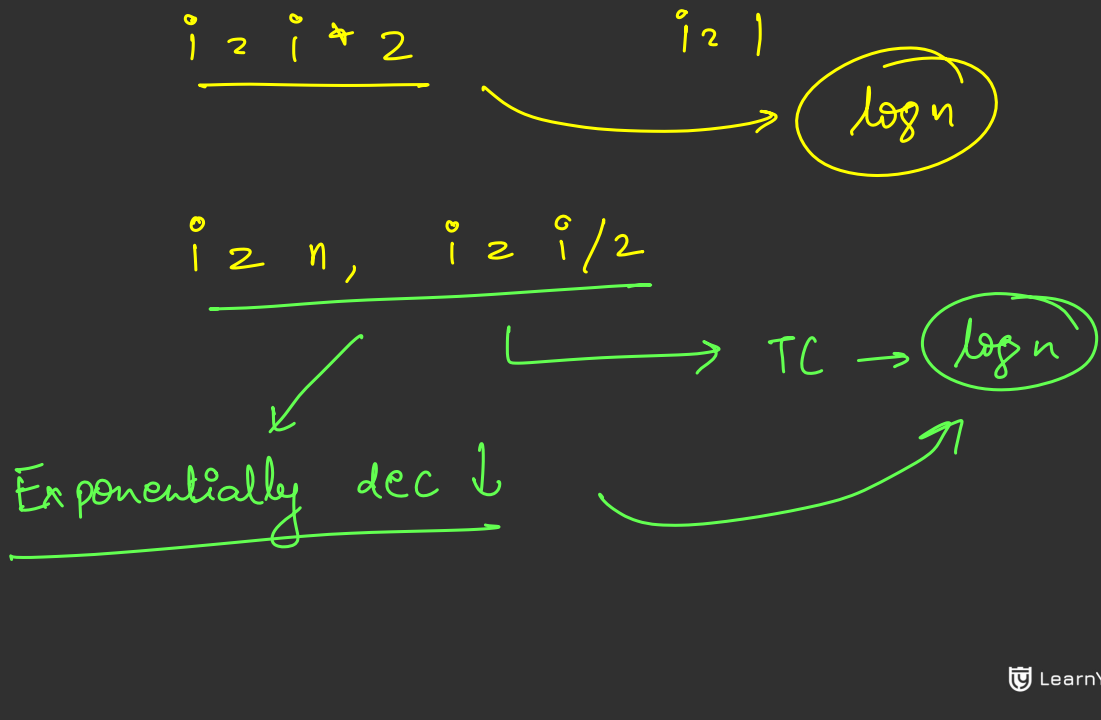
3 ⟶ log n

i = 1
i = 2
i = 4
i = 8

Exponentially inc !




i = 1, 2, 4, 8 - - - - 2^k
2^1 2^2 2^3
k steps
Loop stops
STOP → i > n
$2^k > n$
Taking log both side
$\log_2 2^k > \log_2 n$
$k > \log_2 n$
LearnYard



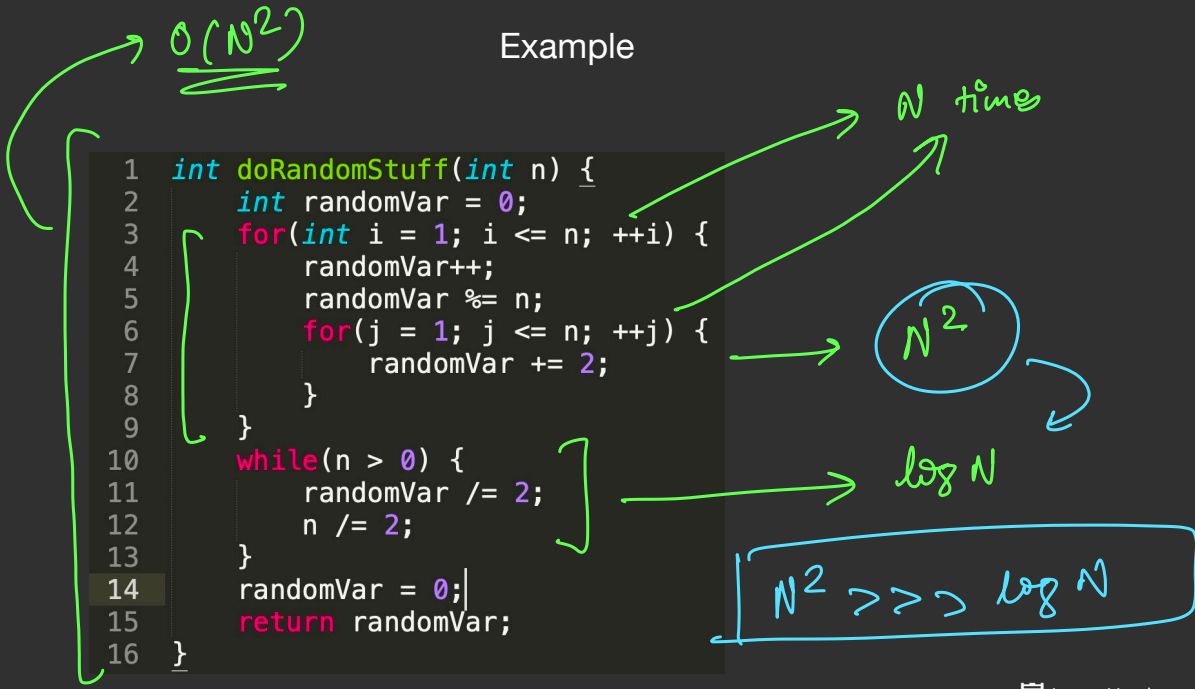
Comparing different time complexities

Time Comp / Input Size	O(LogN)	O(√N)	O(N)	O(NLogN)	O(N*N)	O(N*N*N)	O(2^N)
20	~20 ns	~22 ns	~100 ns	~450 ns	~2 µs	~40 µs	~2 ms
50	~30 ns	~35 ns	~250 ns	~2 µs	~12 µs	~625 µs	~ 2 months
500	~45 ns	~111 ns	~3 µs	~25 µs	~1 ms	~650 ms	Out of Syllabus xD
5000 (5*10^3)	~60 ns	~350 ns	~25 µs	~300 µs	125 ms	~11 mins	Out of Syllabus xD
1 million (10^6)	~100 ns	~5 µs	~5 ms	~100 ms	~1.5 hours	~159 years	Out of Syllabus xD
100 mil (10^8)	~135 ns	~50 µs	~500 ms	~13 secs	~1.6 years	~159 megayears	Out of Syllabus xD
1 billion (10^9)	~150 ns	~0.2 ms	5 secs	~2.5 mins	~159 years	~159 eons	Out of Syllabus xD
1 trillion (10^12)	~200 ns	~5 ms	~1.5 hours	~5.5 hours	~159 megayears	~159 billion eons	Out of Syllabus xD
10^15	~250 ns	~200 ms	~2 months	~8 years	~159K eons	Out of Syllabus xD	Out of Syllabus xD
10^18	~300 ns	~ 5 secs	~159 years	~95 centuries	~159 billion eons	Out of Syllabus xD	Out of Syllabus xD

Approximate Time Taken
(assuming 2*10^8 operations per second)

How to calculate time complexity?

Example



Let’s try some problems

Let's start with a simple one. Time Complexity of the following code?

```
int Sum1ToN(int n) {
  int ans = n*(n+1)/2;
  return ans;
}
```

- O(N)
- O(1)
- O(N*N)
- O(LogN)

Still easy, what is the time complexity of the following code?

```
int getSum(int n, int m) {
  int ans = 0;
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j <= m; ++j)
      ans++;
  return ans;
}
```

Copy

N times

M times

TC → O(N * M)

- O(N)
- O(M)
- O(N + M)
- ✓ O(N*M)

Recorded Example

What is the time complexity of the following code?

```
int doRandomStuff(int n, int m) {
  int ans = 0;
  for(int i = 1; i <= n; ++i) {
    int var = n;
    while(var > 0) {
        // do some O(1) operation.
        var /= 2;
    }
  }

  for(int j = 1; j <= m; ++j)
    ans++;
  return ans;
}
```

N time

log N time

N

M time

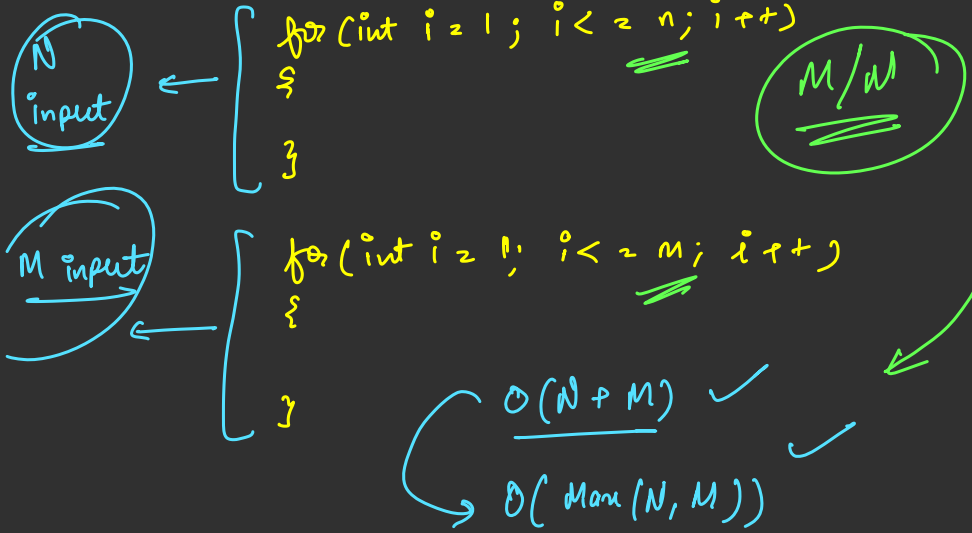
$N \leq 10^3$

$M \leq 10^7$

$O(N \log N + M)$

$\hookrightarrow \text{Max}(N \log N, M)$

- O(N*LogN)
- O(N + LogN + M)
- O(N + M)
- ✓ O(N*LogN + M)



Thank You!

Reminder: Going to the gym & observing the trainer work out can help you know the right technique, but you'll muscle up only if you lift some weights yourself.

So, PRACTICE, PRACTICE, PRACTICE!