L19
Time & Space Complexity

Join Discord - https://bit.ly/ly-discord

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?

5

0.2 sec — C++

$\boxed{0.3 \text{ sec}}$ C++

8

```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));

    }
}
```

RUN

But, the time taken also depends on:

Device

Programming Language

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

Input(s) given

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

$n = 5$

$n = 50$

Find a formula for the
number of steps that
are needed to be done
by the code for a given input.

$$\frac{n * (n+1)}{2}$$

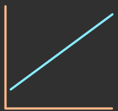
1

=

1

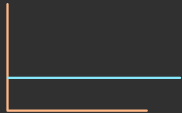
Example - Finding the sum of 1st N natural numbers

.01s



```
for(int i=1; i<=n; i++)
    sum += i;
```

$O(n^1)$ $O(n)$



$$\frac{n \times (n+1)}{2}$$

n^0

$O(1)$

$n = 1$	1	0.01	1
$n = 10$	10	0.10	1
$n = 100$	100	1	1

Formal Definition

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

What about Space Complexity?

73 bytes?

100 KB?

24 MB?

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.

Example : Sum of 100 numbers

1

58
+ 82
+ 98
+ 100
+ 67
+ 48
+ 672
+ 498

100

1623

1000

O(n)

2

100

running_sum = 0

O(1) 1623

LearnYard

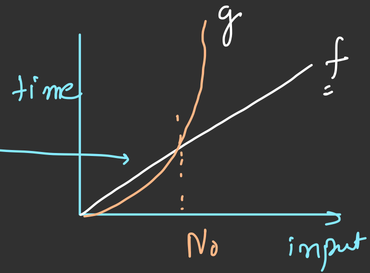
Different Notations

O → upper bound

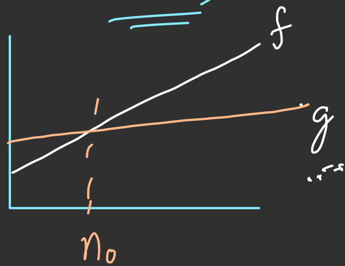
Ω → lower

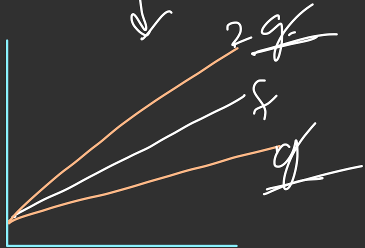
Θ → exact

$O(g)$



$O(n)$





LearnYard

L

2	3	6	1	7	8	10	5

find

2 best

7 Avg

5 Worst

n n

n

$10n$ $10n$

$O(n)$

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. $O(n\log n)$: Linearithmic time complexity
5. $O(n^2)$: Quadratic time complexity
6. $O(2^n)$: Exponential time complexity

LearnYard

Why is time complexity important in DSA?

$O(1)$ ✓ $\quad \underline{\underline{1\ sec}} \quad \underline{\underline{2 \times 10^8}}$

$O(N)$ ✓

$O(\log N)$ ✓ $\quad \underline{\underline{O(N^2)}}$

$O(N^2)$ ✗ $\quad \downarrow$

$\underline{\underline{x^2}}$

$1\ sec = \boxed{2 \times 10^8 \text{ Ins.}}$

$\underline{\underline{2 \times 10^8\ sec}} \quad \underline{\underline{4 \times 10^{16}}}$

Input

2×10^8

1 sec

	Instruction	
$O(N)$	2×10^8	1
$O(N^2)$	4×10^{16}	2×10^8 sec
$O(N^3)$	8×10^{24}	4×10^{16} sec
$O(N^{1/2})$		

$$\sqrt{2 \times 10^8} = 1.414 \times 10^4$$

$$1\text{ sec} = 2 \times 10^8 \text{ ins.}$$

$$? = 4 \times 10^6 \text{ ins.}$$

$$\frac{\overset{2}{\cancel{4}} \times \cancel{10^{16}}\,10^8}{\cancel{2 \times 10^8}}$$

$$\boxed{7.07 \times 10^{-5}}$$

$$\frac{1.414 \times 10^4}{2 \times 10^8}$$

LearnYard

Input		Instruction		time
10^4	✓ $O(N)$	10^4	$\rightarrow$	5×10^{-5} sec
	✓ $O(N^2)$	10^8	$\rightarrow$	$.5$ sec
	$O(N^3)$	10^{12}	$\rightarrow$	5×10^3 sec
	✓ $O(\sqrt{N})$	10^2		

$$\frac{10^{12}}{2 \times 10^8} \qquad \frac{10^8}{2 \times 10^8} \qquad \frac{10^4}{2 \times 10^8} \qquad \underline{\underline{5 \times 10^{-5}}}$$

Comparing different time complexities

2×10^8

1 sec

10^4

Input Size \ Time Comp	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

$1 + n + n + n^2 + \log_2 n + 1 + 1$

$\cancel{4} + 2n + n^2 + \log_2 n$

```
int doRandomStuff(int n) {
    int randomVar = 0;
    for(int i = 1; i <= n; ++i) {
        randomVar++;
        randomVar %= n;
        for(j = 1; j <= n; ++j) {
            randomVar += 2;
        }
    }
    while(n > 0) {
        randomVar /= 2;
        n /= 2;
    }
    randomVar = 0;
    return randomVar;
}
```

n

1

n

n

n^2

$2 \times n$

$2 + 2 + \cdots\cdots n$

$n \quad n \quad n \quad n \quad n$

$i=1 \quad 2 \quad 3 \quad 4 \quad n$

$n \times n \; 2$

$\log_2 n$

1

1

$K+1 = \log_2 n + 1$

$n \quad \frac{n}{2} \quad \frac{n}{4} \quad \frac{n}{8} \quad \frac{n}{16} \quad \cdots \quad 1$

$\frac{n}{2^0} \quad \frac{n}{2^1} \quad \frac{n}{2^2} \quad \frac{n}{2^3} \quad \frac{n}{2^4} \quad \cdots \quad \frac{n}{2^K}$

$1 = \frac{n}{2^K}$

$2^K = n$

$\log 2^K = \log n$

$K \log 2 = \log n$

$K = \log n / \log 2$

$= \log_2 n$

$\cancel{4} + \cancel{2n} + 3n^2 + \cancel{\log_2 n}$

slower

fast

$4 + 2000 + \boxed{1000000} + 9.96$

$O(n^2)$

1000

$O(3n^2) \approx O(n^2)$

$O\left(\frac{n^2}{2}\right) = O(n^2)$

for (1 —— n)
{
for (1 —— k) → k

$(k+n) \times n$

for (1 —— n) → n
}

LearnYard

① for (1 — 100) → $O(1)$

② for (1 — n) → 10 $O(n)$

400 1000

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;
}
```

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

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;
}
```

- 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!