Runtime and Space Complexity

Course on C-Programming & Data Structures: GATE - 2024 & 2025

Vishvadeep Gothi • Lesson 20 • Dec 19, 2022

Data Structure:
Introduction

By: Vishvadeep Gothi

Data Structure

Chapter Number	Chapter Name
1	Data Structure Basics
2	Arrays
3	Linked List
4	Queue & Stack
5	Tree
6	Graph
7	Hashing

Data Structure

- Mathematical and logical model of organizing the interrelated data.

Types of Data Structure

- **Types:**
  1. **Linear:** Elements are arranged in linear (sequential) fashion.
  2. **Non-Linear:** Elements are arranged in non-linear fashion.

Classification of Data Structure

Operations on Data Structures

Operations on Data Structures

Operations on Data Structures

Algorithm

Analysis of Algorithm

Analysis of Algorithm

- **Space Complexity** → no. of spaces (no. of element storage) needed by algo.
- **Run-Time Complexity**

→ Time needed to run an algo

Eg :- →

	A1	A2
	t_1	t_2

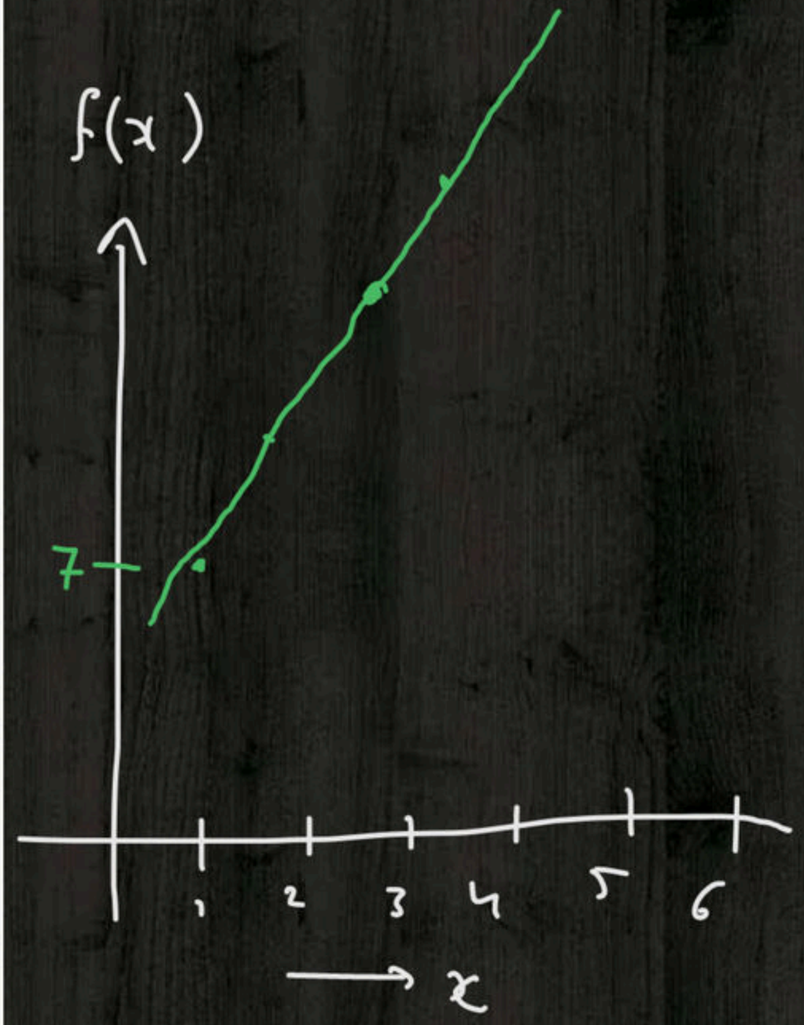
$t_1 < t_2$

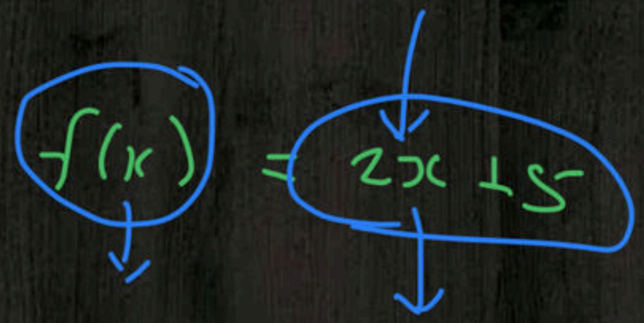
A1 is efficient

How to calculate time

- On wall clock
  - ① implement algo in a prog. language
  - ② Run program on a system & measure time to run.
  - ⇓
  - time is dependent on so many external factors ⇒ no any consistency in result
- Based on no. of operations or steps performed
  - ⇓
  - based on rate of growth

Analysis of Algorithm





$x = 1$, $f(x) = 7$

$x = 2$ $f(x) = 9$

$x = 3$ $f(x) = 11$

$x = 4$ $f(x) = 13$

$x = 5$ $f(x) = 15$

Rate of growth :-

An algo takes n numbers as input

n	no. of steps performed to run algo
1	4
2	5
3	6
4	7
5	8
⋮	
n	n+3

as the size of input increases, the run time increases with what rate?

↓

linear, logarithm, quadratic, exponential. ⋮

run time

If rate of growth	Complexity
Constant	1
Logarithmic	$\log n$
Linear	n
Linear logarithmic	$n \log n$
Quadratic	n^2
Quadratic log.	$n^2 \log n$
Cubic	n^3
:	
Exponential	2^n

example:- Array of n elements

Total no. of steps
= 1 + n + 1 + n
= 2n + 2
T(n) = 2n + 2

Algo:-

① take a variable sum, with initial value zero ——— ①

② Run a loop for n times with i ranging from 0 to n-1
for(i=0; i<n; i++) ⟶ n+1
{
sum = sum + array element at index i
sum = sum + A[i] ——— n
}

ex:- input :- array of size n

A[], starting index => 0

last index => n-1

algo:-

result = (A[0] + A[n-1]) / 2 ——— 1

return result ——— 1

Run time = 2

n	T(n)
5	2
6	2
100	2
⋮	
100100	2

Assume:- There are 2 algos for some problem

A1 ⇓ A2

Run Time ⇒ $n \log n$ n^2

A1 is efficient

Example: 1

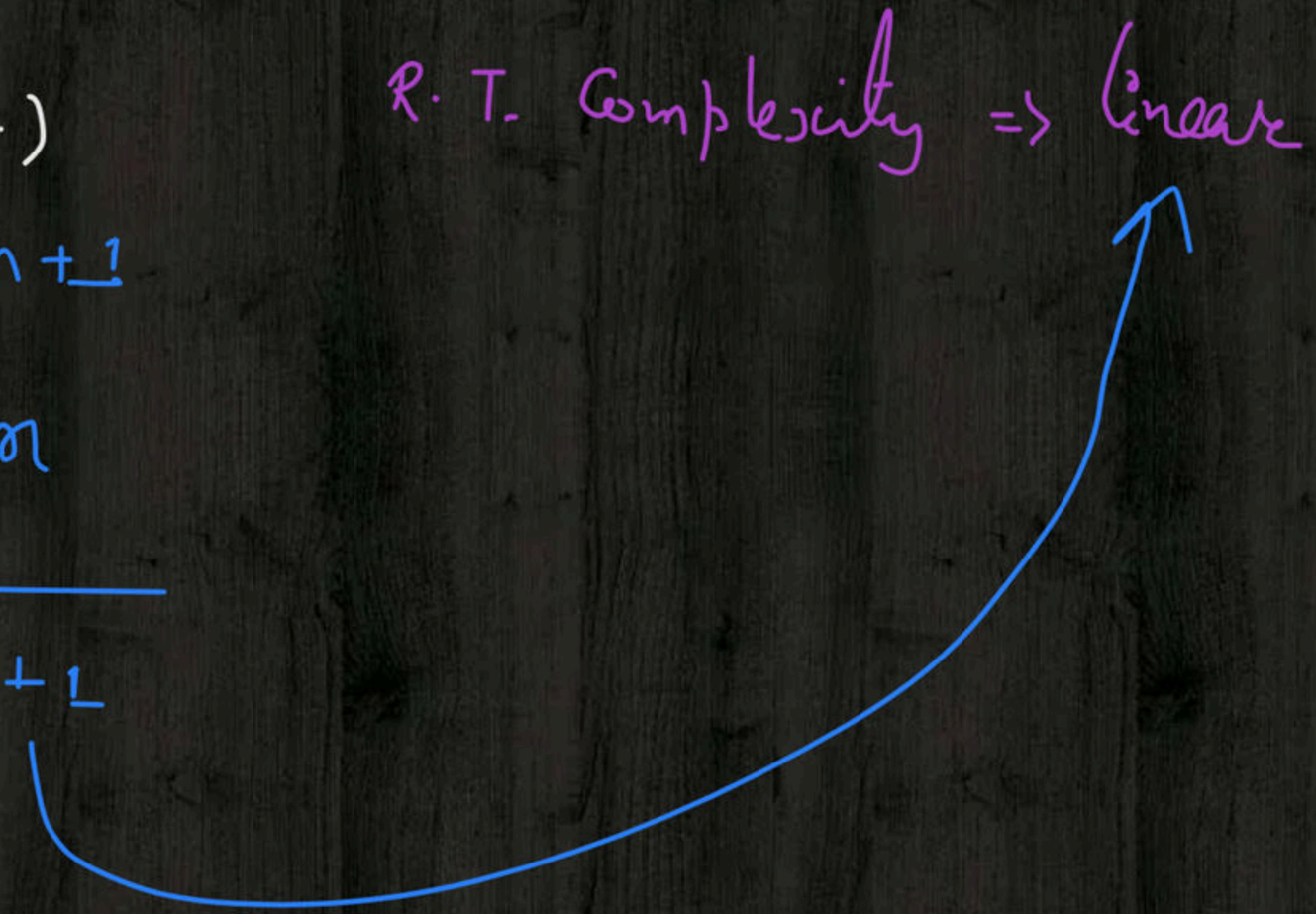
```
for( i = 1; i <= n; i++)
{
  x = A[i] + y;
}
```

$\hookrightarrow n+1$

$\longrightarrow n$

Total $\Rightarrow 2n+1$

R.T. Complexity => linear



Example: 2

```
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
    {
        x = A[i][j] + y;
    }
}
```

`for(i=0; i<n; i++)` ——— $n+1$ = $n+1$

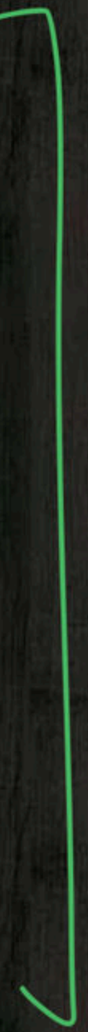
`for(j=0; j<n; j++)` ——— $n*(n+1) = n^2+n$

`x = A[i][j] + y;` ——— $n*n = n^2$

Total $\Rightarrow 2n^2+2n+1 \Rightarrow$ R.T. Complexity $\Rightarrow$ Quadratic

Example: 3

```
for(i = 0; i<n;  i++)
{
  for(j=0; j<n; j++)
  {
    for(k= 0; k<n; k++)
    {
      y =  A[i][j][k] +y;
    }
  }
}
```



R.T. Complexity $= n^3$

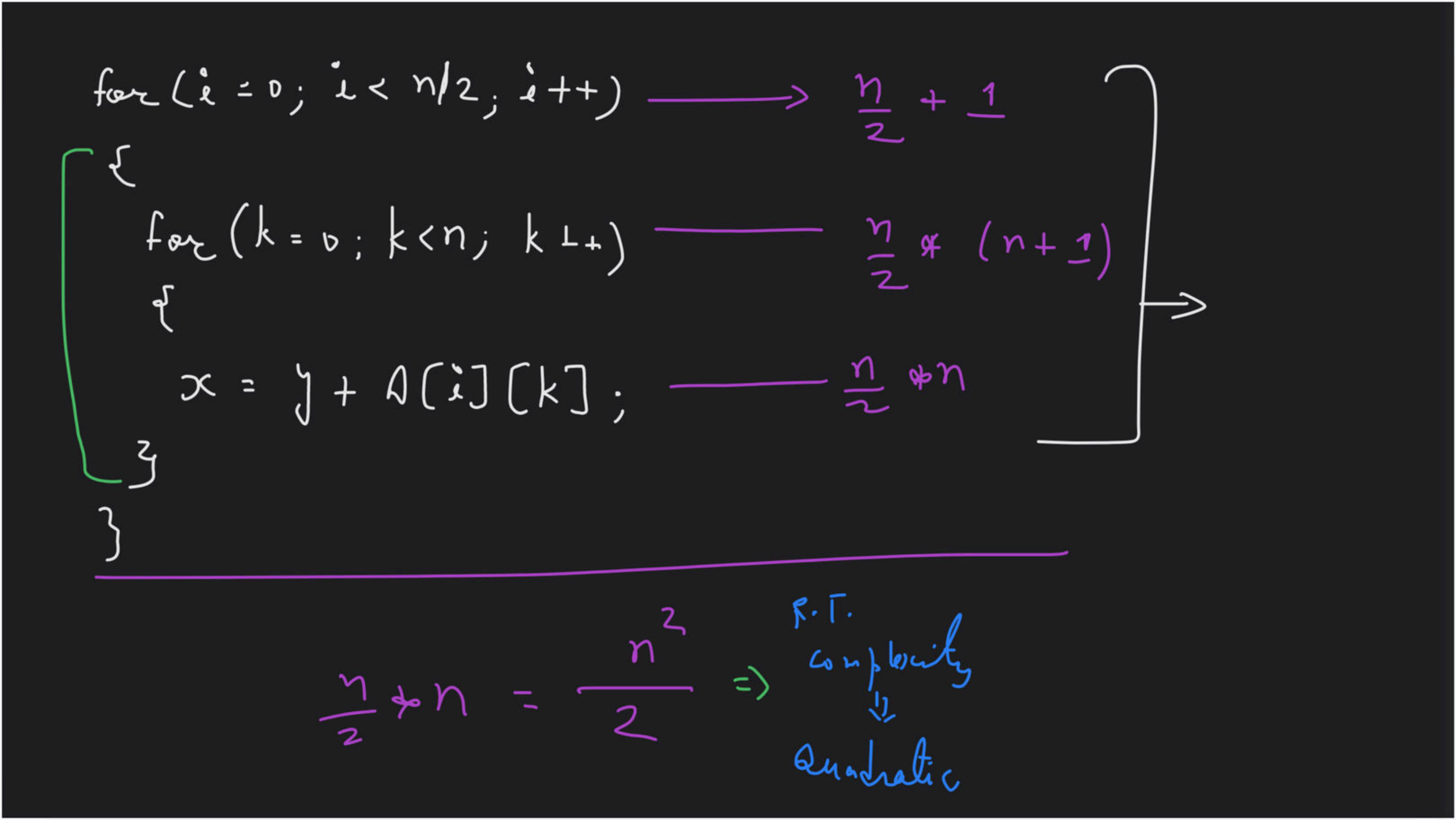
```
for (i = 0; i<n; i++)
{
  for (k=0; k<n; k++)
  {
    x = A[i][k] + y;
  }
for (j=0; j<n; j++)
{
q = A[i][j] + y;
}
}
```

n^2

n^2

$n^2 + n^2 = 2n^2$

R.T. Complexity = Quadratic

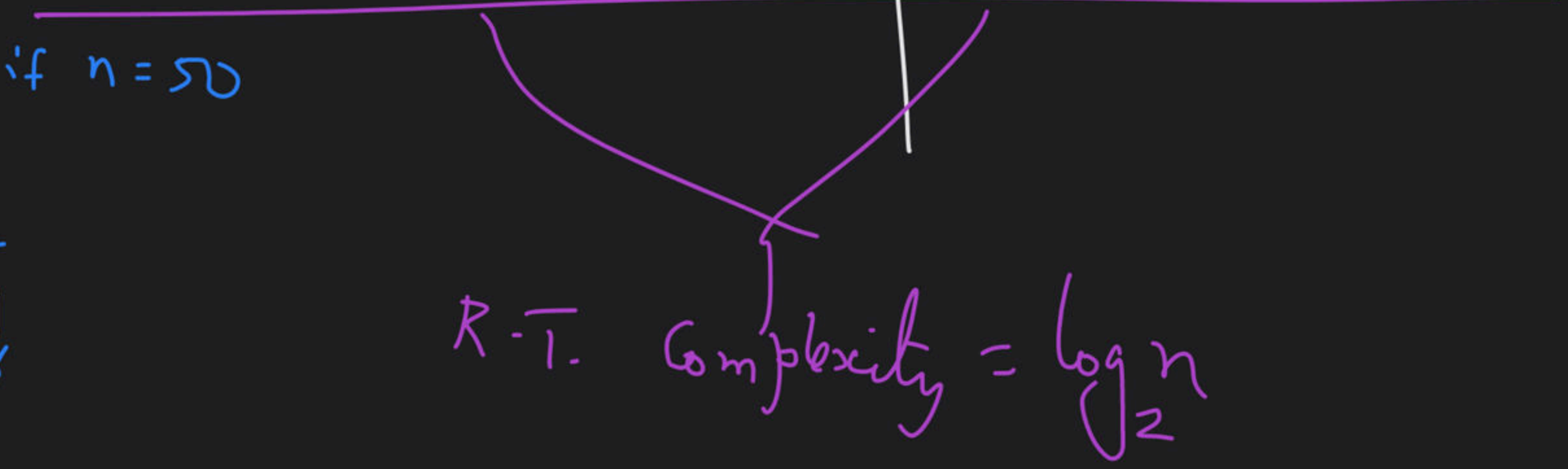

for (i = 0; i < n/2; i++) ⟶ n/2 + 1
{
for (k = 0; k < n; k++) ⟶ n/2 * (n + 1)
{
x = y + A[i][k]; ⟶ n/2 * n
}
}
n/2 * n = n^2/2 ⇒ R.T. complexity ⇓ Quadratic

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

}
```

```
for(i=n; i>=1; i=i/2)
{

}
```



i = 1
= 2
= 4
= 8
= 16
= 32

R.T. Complexity = $\log_2 n$

$$x = \log_2 n$$

$$\Downarrow$$

$$2^x = n$$

Asymptotic Notation

Asymptotic Notation

Question 1

Consider an algorithm which takes n number of inputs and performs an operation on it, which requires n-1 operations. The best possible run time complexity for the algorithm can be represented as:

(A) $O(n)$

(B) $\Theta(n)$

(C) $O(n \log_2 n)$

(D) A & B both

Question 2

Consider an algorithm which takes n number of inputs and performs an operation on it. The operation is performed by algorithm in such a way that it is not dependent on number of inputs. Which of the following can be the run time complexity for the algorithm?

(A) O(1)

(B) Θ(1)

(C) O(n)

(D) All

Happy Learning