ALGORITHM COMPLEXITY

MARTZEL P. BASTE

LEARNING ACTIVATOR

easy to understand
and debug

nt use of the
s resources

INTENDED LEARNING OUTCOMES

- Explain the importance of program efficiency
- Discuss the underlying concepts about space and time complexity in evaluating program performance

input	Declaration	Process	Output
• Take two integers	• Create ***sum*** variable	• store the result to ***sum*** variables	• return the "*sum*" variable

```java
public int add(int num1, int num2) {
    int sum; // creating the sum variable
    sum = num1 + num2; // storing the sum of num1 and num2
    return sum; // returning the sum variable
}
```

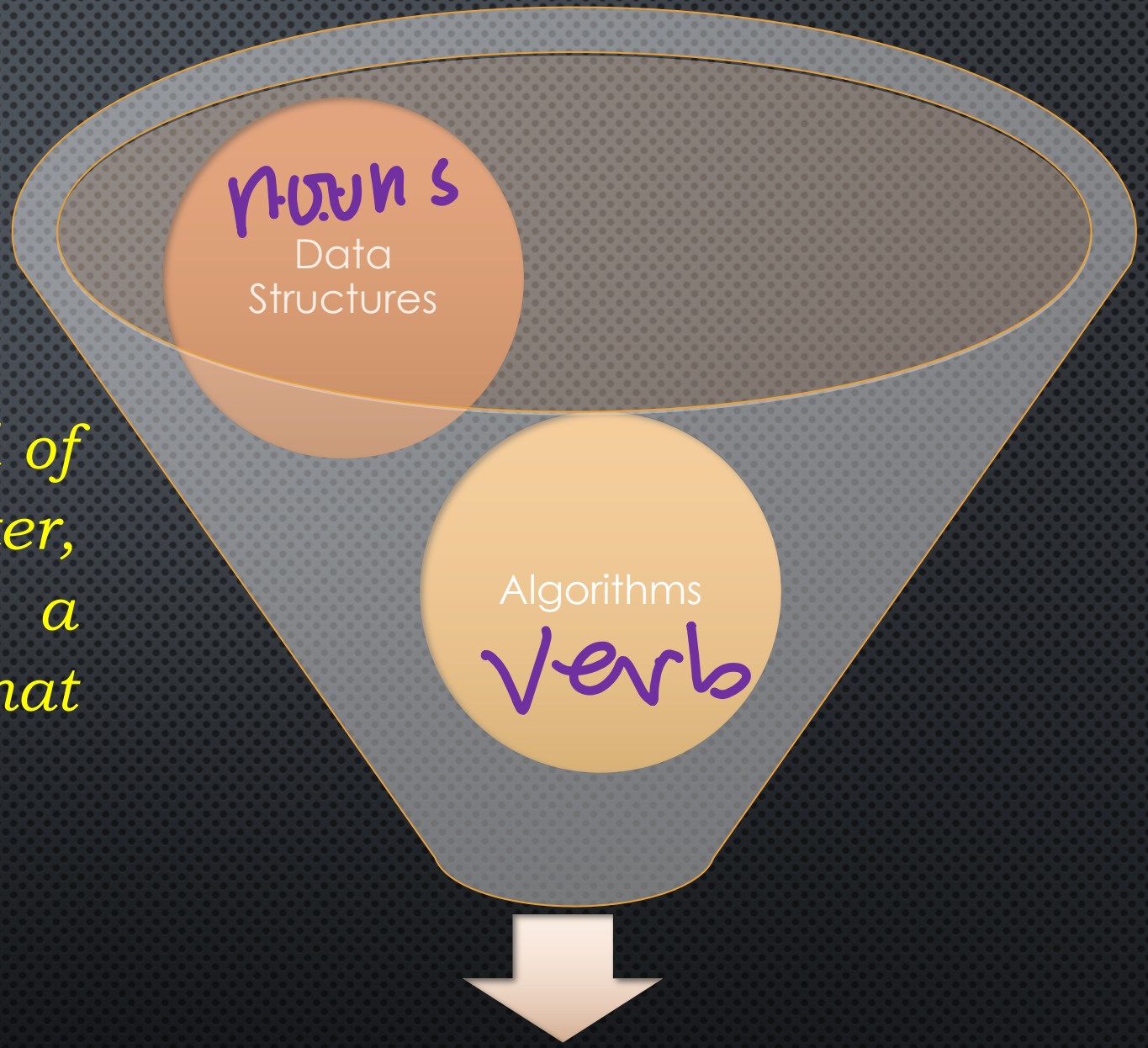
Input

Algorithm

Output

An Algorithm is just a method of doing something on a computer, while a Data Structure is a layout for memory that represents some sort of data.

~ Om Singh



THE PERFORMANCE OF AN ALGORITHM IS MEASURED BASED ON THE FOLLOWING PROPERTIES :

Time Complexity

Space Complexity

```
int max=new Scanner(System.in).nextInt();
int arr[]=new int[max];
for(int i = 0; i <max; i++) {
    arr[i]=new Scanner(System.in).nextInt();
}
```

256MB -> 10^8 for normal array declaration

4MB -> 10^6 for array in a function

SPACE COMPLEXITY

is the amount of memory used by the algorithm (including the input values to the algorithm) to execute and produce the result.

FOR ANY ALGORITHM MEMORY MAY BE USED FOR THE FOLLOWING:

SPACE COMPLEXITY = AUXILIARY SPACE + INPUT SPACE

```
public void process() {
    input();
}

public void input() {

}
```

~~INSTRUCTION SPACE~~ ~~ENVIRONMENTAL STACK~~ DATA SPACE

Data Type	Size
byte, boolean	1 byte
short, char	2 bytes
int	4 bytes
long, double	8 bytes

CALCULATING THE SPACE COMPLEXITY

Data Type	Size
byte, boolean	1 byte
short, char	2 bytes
int	4 bytes
long, double	8 bytes

```java
public int add(int num1, int num2) {
    int sum;
    sum = num1 + num2;
    return sum;
}
```

Constant space complexity

num1
num2 } int → 4 bytes
sum

4(3) + 4 = 16

return value

Data Type	Size
byte, boolean	1 byte
short, char	2 bytes
int	4 bytes
long, double	8 bytes

```java
public int add(int arr[], int max) {
    int sum=0;
    for(int i = 0; i <max; i++) {
        sum=sum+arr[i];
    }
    return sum;
}
```

4*max (bytes) → arr[]

4 bytes for max, i, and sum

4+1+2+4

Linear space complexity

TIME COMPLEXITY

Input Size: *is defined as total number of elements present in the input.*

For a given problem we characterize the input size ***N*** *appropriately. For example:*

Sorting problem	*Graph Problem*	*Numerical Problem*
Total number of item to be sorted	*Total number of vertices and edges*	*Total number of bits needed to represent a number*

TIME COMPLEXITY

1 statement is 1 second
N statements is N seconds

run all the three algorithms on three **different** computers

run all the three algorithms on the **same** computer

TIME COMPLEXITY

Sum of first **N** natural numbers

```
int max=new Scanner(System.in).nextInt();
int sum=0;
for(int i =1; i <=max; i++) {
    sum=sum+i;
}
System.out.println("The sum 1 to "+max+" is "+sum);
```

```
20
The sum is: 210
The sum is: 210
```

```
int max=new Scanner(System.in).nextInt();
int sum=0;
sum=max*(max+1)/2;
System.out.println("The sum 1 to "+max+" is "+sum);
```

loop N times
1+2+3+...+max

natural number
n*(n+1)/2

- In the first solution, the time complexity will be **N** atleast, there is an increase of time taken as the value of **N** increases.
- While in the second, time complexity is **constant**, it will always give the result in 1 step because it is NOT dependent of **N**,
- And since the algorithm's performance may vary with different types of input data, hence for an algorithm we usually use the **worst-case Time complexity** of an algorithm because that is the maximum time taken for any input size.

SUMMARY

1. The efficiency of an algorithm is mainly defined by two factor; the Time Complexity and Space Complexity
2. While executing, algorithm uses memory space for three reasons:
    1. **Instruction Space-** It's the amount of memory used to save the compiled version of instructions.
    2. **Environmental Stack -**Sometimes an algorithm(or a function) may be called inside another algorithm.
    3. **Data Space -**Amount of space used by the variables and constants.
3. The time complexity is the number of operations an algorithm performs to complete its task with respect to **input size**.
4. The algorithm that performs the task in the smallest number of operations is considered the most efficient one.
5. ***Input Size:*** *Input size is defined as total number of elements present in the input.*

Thank you!

Martel Baste