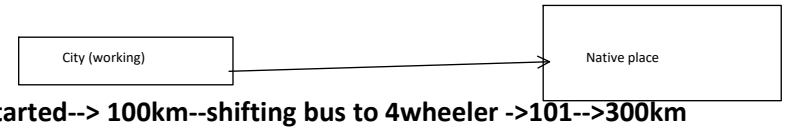
Learning the concepts:
What exactly the topic
Why we need to use
Where we need to apply
How we need to implement
Any alternatives for the topic / any +ve and -ve about concept

What is an algorithm?
"an algorithm is a step-by step method to solve a problem"
***an algorithm is not a code**

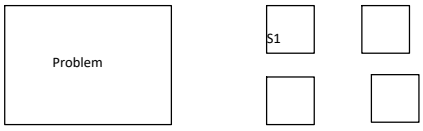
- **Not a program**
- **Instructions to reach destination**
- **It is just a set of logical steps to solve a problem** **400km**



Started--> 100km--shifting bus to 4wheeler ->101-->300km
Shift 4wheeler to train--> 301 to 400km -- End

10+20?
+ --> we need to add two variables because the + indicates the addition
30

Why do we need an algorithm?
Algorithms helps us think logically
A programmer must know how the solution works before writing any code
Algorithms are Language - independent :
Same algorithm can be implemented in python,c,c++,jave etc..
Algorithms prevents errors :
if logic is correct then coding becomes very easy
Algorithms help break big problems in to smaller steps
First divide the problem in to smaller then solve each problem separately then you will get the final solution



Find the solution for **biggest** element and store that value in "**Result**" variable

a=10 b=30
Which is the largest variable
Print("the largest value");
Largest_value;
If a>b then
Print ("a is the largest value")
Else
Print ("b is largest value")

a=10,b=30
Step 1 find biggest element
If a>b
Step2 result
then store result= a
Else store result =b

Characteristics of a good Algorithm :

1. **Input ---->** takes values or data
2. **Output** -->Produces a result
3. **Definiteness** --> every step must be clear ,unambiguous ,and specific . No confusing or vague instructions
   Example :- step: compare A variable with B and store in "Result" variable (clear step)

Bad step :do the comparison and store result

4. **Finiteness** --> must end after a definite number of steps
   -An algorithm must end after a limited number of Steps
   -It cannot run forever
   i=10 j=5
   While (i>j)
   {
   Looping
   j++
   }
5. **Effectiveness -->** steps must be simple and executable
   All steps must be simple and can be performed easily using basic operations.
   The algorithm should not ask you to do impossible tasks
   a=20 b=30 result addition of 2 numbers store in total variable :

Result =a+b
Step1- "Add two number " -->can be done
Step 2 "multiple two numbers" --> can be done
Step3 "read what mindset about raviteja right now " --> not possible (not effective)

6.Correctness: Must produce the correct result
An algorithm should always give the **correct result** for every valid input
Ex- find largest number between 4,9,2
correct output:9
If result will get 4 or 2 its not correct

Example Algorithm:
Using an ATM;
Start:
1.insert card
2.enter pin number (4digit)
3.choose withdrawal
4.Enter the amount
5.dispense cash
6.print receipt
7.end

How to write an algorithm :
Step1: identify input
Step2:identify Output
Step3: List steps clearly and in order
Step4:Make sure every step must be executable
Step5: End the algorithm

Find the square of a given number

Step 1: start
Step 2: read number N
Step 3: compute square =N*N
Step 4 : display the Square
Step 5 : stop

Convert Celsius to Fahrenheit

Step 1: start
Step 2:read temperature in Celsius(c)

Step 3: Compute Fahrenheit=(c*9/5)+32
Step 4 : display Fahrenheit value
Step 5: stop

Pseudocode:
"pseudocode means English-like code."
It is not a programing language :,
->no bracket
-> no semicolons
-> no strict syntax
->only logic in plain English
Why use pseudocode?
Before wring real code ,we must be very clear about the steps.
Pseudocode helps us think,plan,and visualize the logic without worrying about any language syntax

Pseudocode format:
Start--> start
End-->END
Read data--> INPUT
Print message-->OUTPUT/PRINT
Conditions -->IF/ELSE/ENDIF
Loops -->FOR/WHILE

Make a Tea ->

Algorithm:
1. Boil water
2. Add tea powder
3. Add milk
4. Add sugar
5. Filter and pour into the cup
6. Stop

Pseudocode :

START
Boil water
Add tea powder
Add milk
Add sugar
Stir well
Filter and pour into cup
END

-->Check even or odd

```
START
INPUT number
IF number %2==0 THEN
    PRINT "Even"
ELSE
    PRINT "Odd"
END IF
END
```

--> finding the largest of two numbers

```
START
INPUT A,B
IF A>B THEN
PRINT "A IS largest"
```

ELSE
Print "B is Largest"
Endif
End

"pseudocode is the bridge between normal English and programing
It makes you think logically.
Once your pseudocode is correct then it is easy to write or
converting in to any programing leagues very easy "
Relationship between alg and pscode
Algorithm is WHAT todo.
Pseudocode is HOW to do

Algorithm=Steps
Pseudocode=English-like code
Both help us solve problems clearly
Coding comes after understanding algorithms.

Algorithmic thinking:

1. **sequential problem solving**
   no decisions, no loops, steps happen exactly in order

Example ADD 2NUMBERS
Input : A B
Output : sum of A and B
Start
Read input A
Read input **B**
Compute Sum= A+B
Display sum
Stop

2. **Decision-based problem solving (conditional thinking)**
   Use if/else statements
   Example evn or odd
   Input : A B
   Output : even or Odd
   Start
   Read input N
   Check if N mod 2=0
   Condition is true -> print "even"
   Else print "Odd"
   Compute Sum= A+B
   Stop

3. **Repetitive problem solving(loops):**
   Repeat steps until a condition is met
   Example sum of first n natural numbers
   Input : N
   Output : Sum of numbers from 1 to N
   Start
   Read input N
   Set sum =0
   Set i=1
   Repeat while i<=n
   -add I to sum
   -increment I value
   Stop

4. **Decomposition (divide into smaller task)**
   Break large problem into small, managble sub problem

Example ATM withdrawal process:

Input : card,PIN,Amount
Output : Cash, receipt
Start
Insert card
Validate card
Enter pin
Verify pin
Display withdrawal option
Read amount
Check the balance
If sufficient balance -> dispense the cash
Print receipt
Eject card
Stop

Flowchart:

"A flow chart is a visual representation of steps in an algorithm or process"

It uses symbols to show actions,decisions,inputs,and outputs in a clear and logical flow

Shapes used to create flowchart:

1.Terminator :

Use for : start of the algorithm
End of the algorithm

2. Process shape(action/instruction):

Use for : -any action
- Any calculation
- Any assignments

Example :- a=10
Sum=a+b
Display output
Insert ATM card
Enter PIN number

3. Decision shape(yes/no conditions)

Use for:
- Conditions
- True/False check
- Loops
- Branching

Example:-
- Is PIN correct?
- Is low>high?
- Is pin correct?

Yes-> continue
NO->show invalid

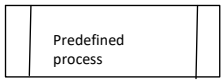
4. Input /output shape(parallelogram)

Use for:
- Taking input
- Printing/displaying output

Example: enter n value

Print "sum"

5. **Connector/Flow Lines:**
   use for:
   Indicating flow of steps
   Showing direction of algorithms
   Types:
   Straight Arrow
   Loop-back Arrow
   Branch arrow (yes/no)
6. **Loop Indicator:**
7. **Predefined process** -->use for:-when calling functions,subroutines,reusable block,



Example : call fib(n-1)
Call factorial(n)

8. **Document Shape:-**
   When the algorithm outputs a document/report/receipt
   Example - AtM print receipt

9. **Data/Database shapes(cylinder)**
   Use for file/data storage .. Database access .
10. **Annotation /note:-**
    Use for adding explanation ,adding clarifications…

Flowchart example:

-->Check even or odd

```
START
INPUT number
IF number %2==0 THEN
    PRINT "Even"
ELSE
    PRINT "Odd"
END IF
END
```

Introduction to Sorting Algorithms:-

Sorting means arranging the data in particular order:-

- Increasing (Ascending Order)
- Decreasing (descending order)
- Alphabetical order
- By date/time
- By price/rating

Why do we use sort:

1.makes searching faster
2.organizing the data nicely
3.improve efficiency of program

Name	Rank
Raju	10
Kiran	5
Santhosh	2
Mani	3

4.saves time in Data processing
5.helps in decision making sorted data-->better analysis-->better

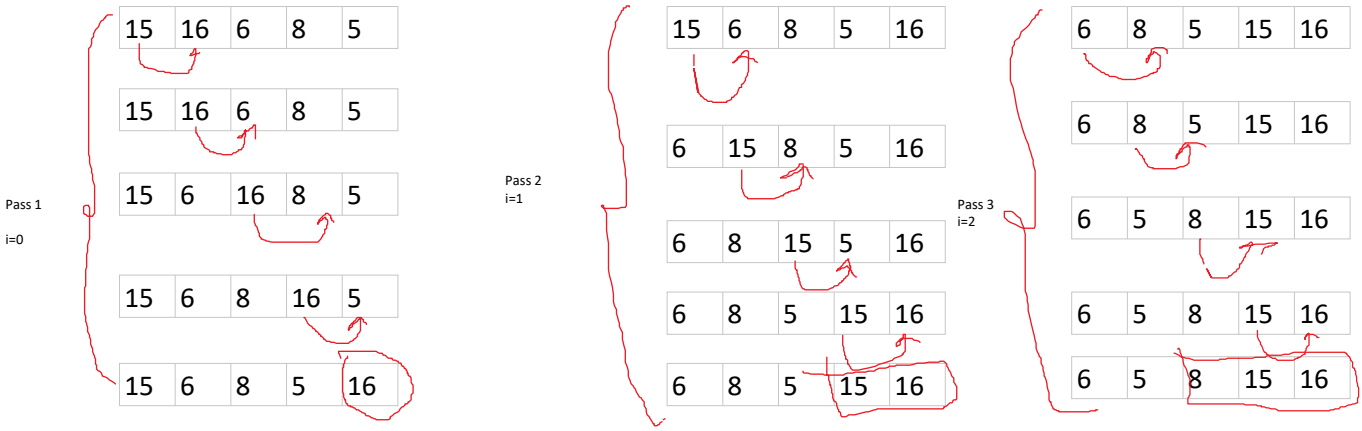
decision making

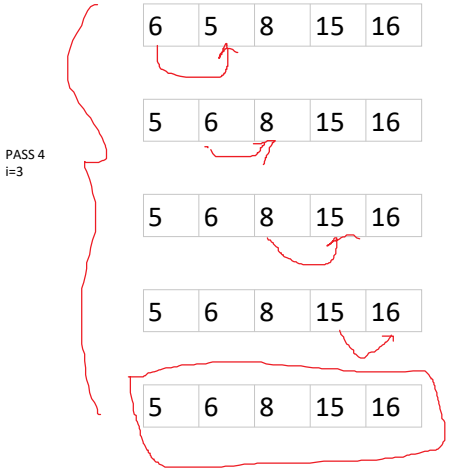
Sorting : Bubble sort and insertion sort :

--> Bubble Sort:

"bubble sort is a simple sorting algorithm that repeatedly compares adjacent elements and swap them if they are in the wrong order"

n=6
Passes =(n-1)





HOW IT WORKS:

1. Compare the first and 2nd elements
2. If the 1st is bigger then swap
3. Move the next pair
4. Keep doing this until the last element
5. Repeat the entire pass until no swaps are needed

```
A=[15,16,6,8,5]
n=4
For I in range(n-1):
    For j in range(n-i-1):
    If A[j]>A[j+1]:
        A[j],a[j+1]=A[j+1],a[j]
```

For i=0,i<=n-1 ,i++

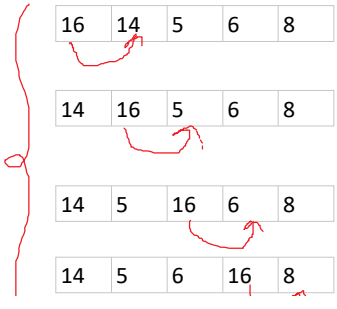
0<=4

For j=0,j<n-1,j++ 0--1

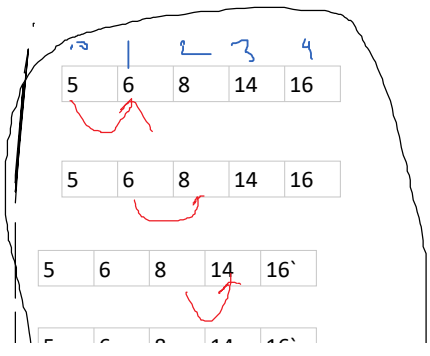
0<3
15>16

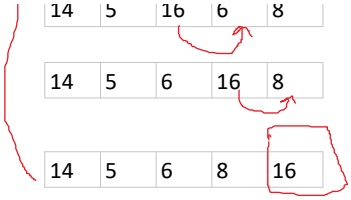
1<3
16>6
16,6=6,15
16=6
6=15

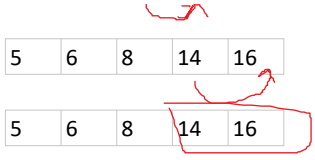
Bubble sort with optimization:

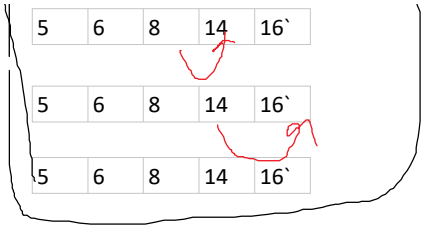


n=5










```
A=[15,16,6,8,5]
n=4
For I in range(n-1):
{
   Flag=0;
     For j in range(n-i-1):
          {
               If A[j]>A[j+1]:
                    {
                    A[j],a[j+1]=A[j+1],a[j]
                    Flag=1
                    }
          }
          If(flag==0)
          Break;
}
```

```
i=2 4<4..

5>  6
6>8
8>14
14>16
```

Algorithm for bubble sort

1. Start
2. Take the input list of numbers.
3. Find the length of the list → n.
4. Repeat the following for i from 0 to n-2:
   - For each j from 0 to n-i-2:
     - Compare element at index j with element at index j+1.
     - If the element at j is greater than the element at j+1, swap them.
5. After completing all passes, the list becomes sorted.
6. Print the sorted list.
7. Stop.

Pseudocode for bubble sort

```
START
A list of numbers
n length of A
For I 0 to n-2
     For j 0 to n-i-2
          If A[i]>A[j+1] then
               Temp  A[j]
               A[j] a[J+1]
               a[j+1] Temp
          Endif
     Endfor
     Print "Sorted array", A
STOP
```