VIT®

BHOPAL

www.vitbhopal.ac.in

CSE1021- Problem Solving and Programming

VIT®

BHOPAL

www.vitbhopal.ac.in

Unit-3

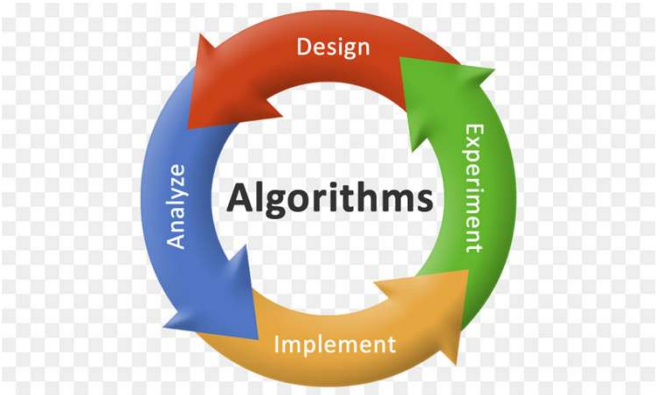
Algorithms

An algorithm is a set of rules that specify the order and kind of arithmetic operations that are used on a specified set of data.

An algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function.

An algorithm is a finite, definite, effective procedure, with some output

Systematic procedure that produces – in a finite number of steps – the answer to a question or the solution of a problem.



Basic Questions About Algorithms

For each algorithm, we should answer the following basic questions:

- does it terminate?
- is it correct?
- is the result of the algorithm determined?
- how much resources will it use in terms of
- memory? (and memory bandwidth?)
- operations?
- run-time

Swapping Two Numbers Using Variable

- In many case, programmers are required to swap values of two variables. Here, we shall learn how to swap values of two integer variables, that may lead to swapping of values of any type. Values between variables can be swapped in two ways −

With help of a third (temp) variable

Without using any temporary variable

Swapping Two Numbers Using Variable With help of a third (temp) variable

Logic:

- Suppose we take two values in a and b, a=10, b=20 respectively.
- temp=a;
- a=b;
- b=temp;
- After the operation of this statement, the value of two variable a and b will be interchanged.
- Output:
- A=20, B=10;

Algorithm:

- Using the temporary variable

Step 1: Start

Step 2: Take the value of variables a and b.

Step 3: Assign the value of variable a into the temp variable.

Step 4: Assign the value of variable b into a variable.

Step 5: Assign the value of variable temp into the b variable.

Step 6: Print the value of a and b.

Step 7: Stop

Pseudocode

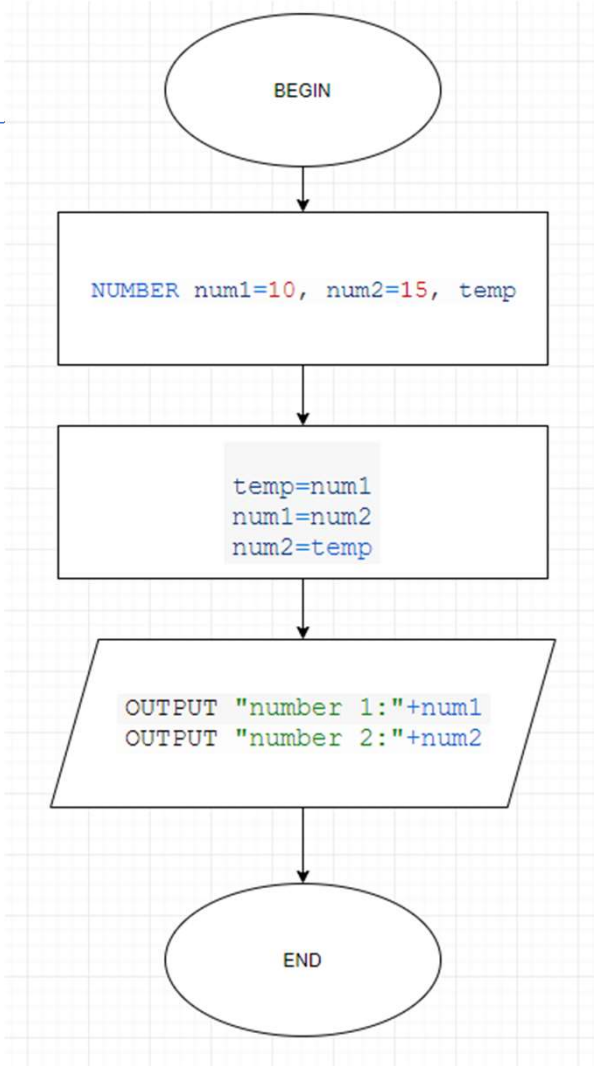
- Using the temporary variable

```
procedure swap(a, b)

   set temp to 0
   temp ← a
   a ← b       // a holds value of b
   b ← temp    // b holds value of a stored in temp

end procedure
```

Flow Chart


BEGIN
NUMBER num1=10, num2=15, temp
temp=num1
num1=num2
num2=temp
OUTPUT "number 1:"+num1
OUTPUT "number 2:"+num2
END

Python Program


```
# Python program to swap two variables

x = 5
y = 10

# To take inputs from the user
#x = input('Enter value of x: ')
#y = input('Enter value of y: ')

# create a temporary variable and swap the values
temp = x
x = y
y = temp

print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))
```

Swapping Two Numbers without Using Variable

- Here we will use arithmetic operator (+, -). We use addition and subtraction operation on these variables.
- a=a+b;
- b=a-b;
- a=a-b;

Algorithm:

- Without Using the temporary variable

Step 1: Start

Step 2: Take the value of variables a and b.

Step 3: Add the values of the a and b and assign it to the variable a.

Step 4: Subtract the values of a and b and assign it to the variable b. (Here the value of a is changed due to the step 3).

Step 5: Subtract the values of a and b and assign it to the variable a.

Step 6: Print the value of a and b.

Step 7: Stop

Pseudocode

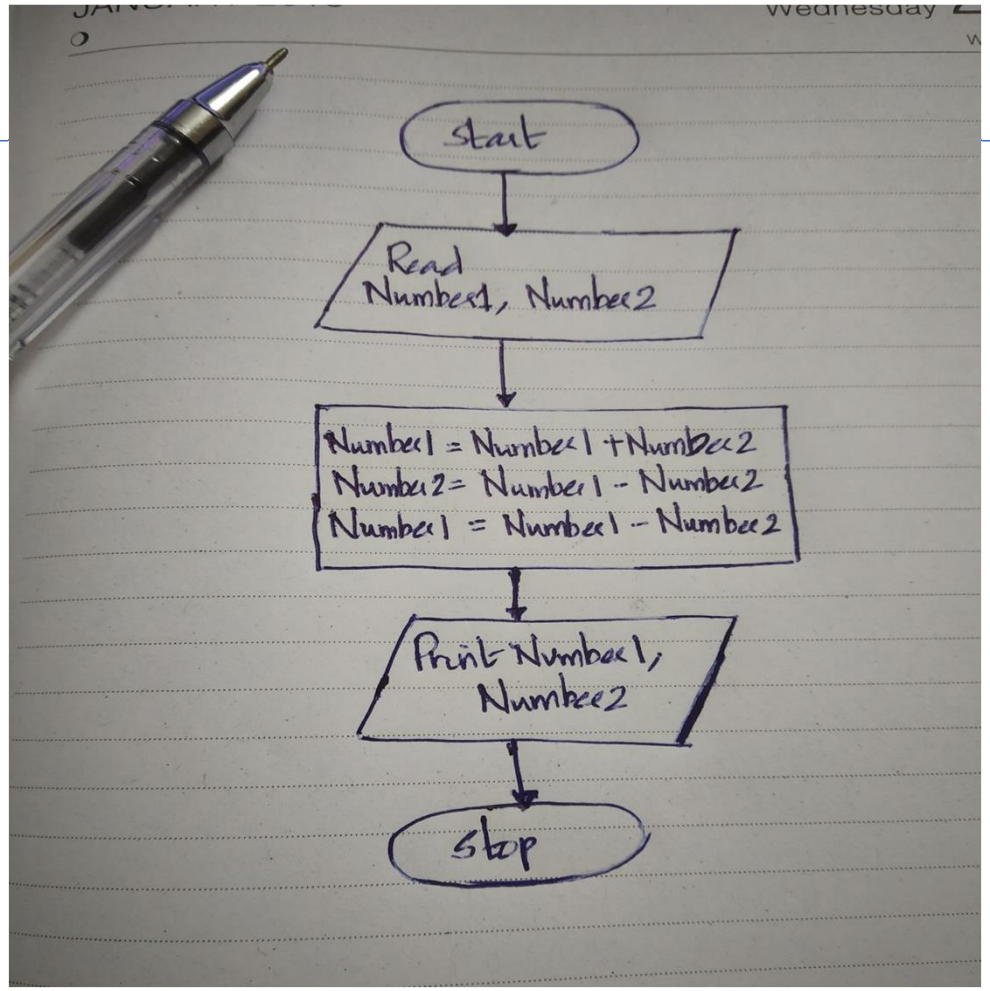
- Without Using the temporary variable

```
procedure swap(a, b)

   a ← a + b    // a holds the sum of both
   b ← a - b    // b now holds the value of a
   a ← a - b    // a now holds value of b

end procedure
```

Flow Chart


Start
Read
Number1, Number2
Number1 = Number1 + Number2
Number2 = Number1 - Number2
Number1 = Number1 - Number2
Print Number1,
Number2
Stop

Python Program


```
x = 10
y = 5

# Code to swap 'x' and 'y'

# x now becomes 15
x = x + y

# y becomes 10
y = x - y

# x becomes 5
x = x - y
print("After Swapping: x =", x, " y =", y)
```

Without using third variable and without using arithmetic operator

- Here we will use the Exclusive OR bit-wise operator.
- Logic:
- a=a^b;
- b=a^b;
- a=a^b;

Algorithm:

- Without Using the temporary variable

Step 1: Start

Step 2: Take the value of variables a and b.

Step 3: Operate Exclusive OR operation on variable a and b and assign the exclusive OR output in a variable. (Here value of a is changed)

Step 4: Operate Exclusive OR operation on variable a and b and assign the exclusive OR output in b variable. (Here value of b is changed)

Step 5: Operate Exclusive OR operation on variable a and b and assign the exclusive OR output in a variable. (Here value of a is again changed)

Step 6: Print the value of a and b.

Step 7: Stop

Python Program


```
# Python 3 code to swap using XOR

x = 10
y = 5

# Code to swap 'x' and 'y'
x = x ^ y; # x now becomes 15 (1111)
y = x ^ y; # y becomes 10 (1010)
x = x ^ y; # x becomes 5 (0101)

print ("After Swapping: x = ", x, " y =", y)
```

Program to count digits in an integer

- The integer entered by the user is stored in variable n. Then the while loop is iterated until the test expression n != 0 is evaluated to 0 (false).
- After first iteration, the value of n will be 345 and the count is incremented to 1.
- After second iteration, the value of n will be 34 and the count is incremented to 2.
- After third iteration, the value of n will be 3 and the count is incremented to 3.
- At the start of fourth iteration, the value of n will be 0 and the loop is terminated.

Program to count digits in an integer

- Method 1

```
# Iterative Python program to count
# number of digits in a number


def countDigit(n):
    count = 0
    while n != 0:
        n //= 10
        count += 1
    return count



n = 345289467
print("Number of digits : % d" % (countDigit(n)))
```

Program to count digits in an integer

- Method

```
# Python3 implementation of the approach
def count_digits(n):
    n = str(n)
    return len(n)


n = 456533457776
print(count_digits(n))
```

Algorithm: calculate the sum and average of first n natural numbers

- Allows a user to enter the number (n) he wishes to calculate the sum and average. The program accepts user input using the input function.
- Next, run loop till the entered number using the for loop and range() function.
- Next, calculate the sum using a `sum = sum + current number` formula.
- At last, after the loop ends, calculate the average using `average = sum / n`. `n` is a number entered by the user.

Python Program to calculate the Sum


```
n = 10
sum = 0
for num in range(0, n+1, 1):
    sum = sum+num
print("SUM of first ", n, "numbers is: ", sum )
```

Python Program to calculate average


```
print ("calculate an average of first n natural numbers")
n = 10
average = 0
sum = 0
for num in range(0,n+1,1):
    sum = sum+num;
average = sum / n
print("Average of first ", n, "natural number is: ", average)
```

Calculate the sum and average of first n natural numbers using formula

In the above program, we calculated the sum and average using loop and range function. Let's see how to **calculate the sum and average directly using a mathematical formula**.

The sum of the first n natural number = `n * (n+1) / 2`, for n a natural number.

the average of first n natural number = `( n * (n+1) / 2) / n`

Calculate the sum and average of first n natural numbers using formula


```
n = 10
sum = n * (n+1) / 2
average  = ( n * (n+1) / 2) / n
print("Sum of fthe irst ", n, "natural numbers using formula is: ", sum )
print("Average of the first ", n, "natural numbers using formula is: ", average )
```

Calculate sum using the built-in sum function in Python

Sum of numbers in the list is required everywhere. Python provide an inbuilt function sum() which sums up the numbers in the list

Syntax:

```
sum(iterable, start)
iterable : iterable can be anything list , tuples or dictionaries ,
 but most importantly it should be numbers.
start : this start is added to the sum of
numbers in the iterable.
If start is not given in the syntax , it is assumed to be 0.
```

Possible two syntaxes:

```
sum(a)
a is the list , it adds up all the numbers in the
list a and takes start to be 0, so returning
only the sum of the numbers in the list.
sum(a, start)
this returns the sum of the list + start
```

sum() function in Python


```
# Python code to demonstrate the working of
# sum()

numbers = [1,2,3,4,5,1,4,5]

# start parameter is not provided
Sum = sum(numbers)
print(Sum)

# start = 10
Sum = sum(numbers, 10)
print(Sum)
```

sum() function in Python

Practical Application: Problems where we require sum to be calculated to do further operations such as finding out the average of numbers.

```
# Python code to demonstrate the practical application
# of sum()

numbers = [1,2,3,4,5,1,4,5]

# start = 10
Sum = sum(numbers)
average= Sum/len(numbers)
print average
```

Factorial Computation

Calculate factorial value of a number

- Factorial of a non-negative integer, is multiplication of all integers smaller than or equal to n.
- For example factorial of 6 is 6*5*4*3*2*1 which is 720.
- **Recursive Solution:**
  Factorial can be calculated using following recursive formula.

n! = n * (n-1)! n! = 1 if n = 0 or n = 1

Algorithm to calculate factorial value of a number

- step 1. Start
  step 2. Read the number n
  step 3. [Initialize]
  i=1, fact=1
  step 4. Repeat step 4 through 6 until i=n
  step 5. fact=fact*i
  step 6. i=i+1
  step 7. Print fact
  step 8. Stop

Prog. calculate factorial value of a number

File Edit Format Run Options Window Help

```
def factorial(n):

    if n == 0:
        return 1

    return n * factorial(n-1)


num = 5;
print("Factorial of", num, "is",
factorial(num))
```

calculate factorial value of a number

- **Iterative solution:**
  Factorial can also be calculated iteratively as recursion can be costly for large numbers. Here we have shown the iterative approach using both for and while loop.

```
def factorial(n):

        res = 1

        for i in range(2, n+1):
                res *= i
        return res


num = 5;
print("Factorial of", num, "is",
factorial(num))
```

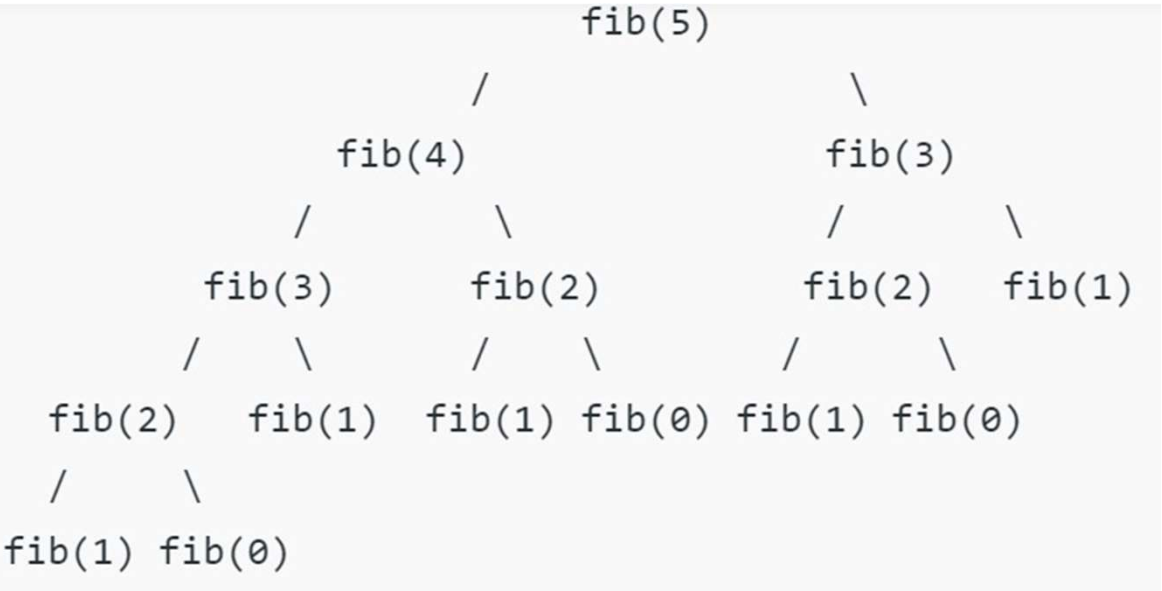
Fibonacci Sequence

- The Fibonacci numbers are the numbers in the following integer sequence.
  0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..
- In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the recurrence relation

$$F_n = F_{n-1} + F_{n-2}$$

$F_0 = 0$ and $F_1 = 1$.

Time Complexity: T(n) = T(n-1) + T(n-2) which is exponential.


fib(5)
fib(4)
fib(3)
fib(3)
fib(2)
fib(2)
fib(1)
fib(2)
fib(1)
fib(1)
fib(0)
fib(1)
fib(0)
fib(1)
fib(0)

Algorithm to print Fibonacci series up to given number N

- Step 1: Start
- Step 2: Declare variables N, N1, N2, N3, i.
- Step 3: Read value of N from user.
- Step 4: if N < 2, display message “Enter number > 2” and go to step 9.
- Step 5: Initialize variables N1 = 0, N2 = 1, i = 0
- Step 6: Display N1, N2
- Step 7: Compute N3 = N1 + N2
- Step 8: Repeat following statements until i < N - 2

  Display N3

  N1 = N2

  N2 = N3

  N3 = N1 + N2

  i = i + 1
- Step 9: Stop

Prog. to print Fibonacci series up to given number N


```
# Function for nth Fibonacci number

def Fibonacci(n):
        if n<0:
                print("Incorrect input")
        # First Fibonacci number is 0
        elif n==0:
                return 0
        # Second Fibonacci number is 1
        elif n==1:
                return 1
        else:
                return Fibonacci(n-1)+Fibonacci(n-2)


print(Fibonacci(5))
```

Reverse a Number

Algorithm to Reverse a Number

- Step 1:Start
- Step 2:Intilize reverse=0
- Step 3:Read digit
- Step 4:check whether digit>0 then go to step 5 else go to step 9
- Step 5:reverse=reverse*10
- Step 6:reverse=reverse+digit%10
- Step 7:digit=digit/10
- Step 8: Go to step 4
- Step 9:Print reverse
- Step 10: Stop

Base Conversion

- Decimal to Any Base
  - Decimal to Octal
  - Decimal to Binary
  - Decimal to Hexadecimal
- Binary to Any Base
- Octal to any base
- Hexadecimal to any base

The Basic Idea

- The basic algorithm for this conversion is to repeatedly divide (integer division) the given decimal number by the target base value until the decimal number becomes 0.
- The remainder in each case forms the digits of the number in the new base system, however, in the reverse order.

An Upper Limit

- The algorithm, in general, will work for any base, but we need digits/characters to represent that many numbers of a base system.
- For simplicity, we limit ourselves to base 36 i.e 10 numbers + 26 alphabets.
  - (We can differentiate lower and upper case, but we intend to focus on the algorithm here!).
- It will be clear after the implementation that the method has to work for any base.

Decimal to Base$_x$

1. Divide the decimal number by the radix of the target base
2. The remainder from step 1 becomes the value for the current column.
3. Use the quotient (answer) from step 1 as the decimal value to calculate the next column.
4. Return to step 1 and repeat until the quotient is zero.

Let's return to the number 237 and convert it to a binary number:

Example – Decimal to Binary

decimal:		237		
radix:		2		
Decimal		**Radix**	**Quotient**	**Remainder**
237	/	2	118	1 (2^0)
118	/	2	59	0 (2^1)
59	/	2	29	1 (2^2)
29	/	2	14	1 (2^3)
14	/	2	7	0 (2^4)
7	/	2	3	1 (2^5)
3	/	2	1	1 (2^6)
1	/	2	0	1 (2^7)
binary:		11101101		

Example – Decimal to Hexadecimal

decimal:		3,134,243,038		
radix:		16		
Decimal		**Radix**	**Quotient**	**Remainder**
3,134,243,038	/	16	195,890,189	E (14) (16^0)
195,890,189	/	16	12,243,136	D (13) (16^1)
12,243,136	/	16	765,196	0 (0) (16^2)
765,196	/	16	47,824	C (12) (16^3)
47,824	/	16	2,989	0 (0) (16^4)
2,989	/	16	186	D (13) (16^5)
186	/	16	11	A (10) (16^6)
11	/	16	0	B (11) (16^7)
hexadecimal:		xBAD0C0DE		

Decimal to Any Base – Python Implementation

```
def dec_to_base(num,base):  #Maximum base - 36
    base_num = ""
    while num>0:
        dig = int(num%base)
        if dig<10:
            base_num += str(dig)
        else:
            base_num += chr(ord('A')+dig-10)  #Using uppercase letters
        num //= base
    base_num = base_num[::-1]  #To reverse the string
    return base_num
```

Base$_x$ to Decimal

- Actually, I have already demonstrated how to convert a number from a base different than ten, into decimal. Once again, here is the complete formula, where c_x represents the coefficients at each place-column.

$$c_n \cdot b^n + \ldots + c_2 \cdot b^2 + c_1 \cdot b^1 + c_0 \cdot b^0$$

As an example, let's convert the binary answer back into decimal:

$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
$1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$
$128 + 64 + 32 + 8 + 4 + 1$
237

$Base_x$ to $Base_y$

- Is it possible to convert a number from $Base_x$ directly to $Base_y$ without converting to decimal (base-10) first?
- Yes, however, you will need to perform all of your math operations in either $Base_x$ or $Base_y$. The algorithms that I have presented are performed with base-10 since that is the number system most people are familiar with.