RECURSIVE ALGORITHM

**TABLE OF CONTENT**

* **INTRODUCTION TO RECURSIVE ALGORITHM**
* *PROPERTIES OF A RECURSIVE ALGORITHM*
* **TYPES OF RECURSIVE ALGORITHM**
* **HOW TO SOLVE A PROBLEM USING RECURSIVE ALGORITHM**
* **SAMPLE PROBLEMS AND THEIR RECURSIVE SOLUTIONS**
* **IMPLEMENTATION IN ANY PROGRAMMING LANGUAGE**

Introduction to recursive algorithm

TO begin with, recursion means breaking down a problem or list to the least or base which can be easily understood and solved.

a recursive algorithm simplifies a problem by breaking it down into sub-problems of the same type. the output of one recursion becomes the input of another recursion.

generally, if a problem can be solved sing solutions to smaller versions of the same problem and the smaller versions reduced to easily solvable cases, then one can use a recursive algorithm to solve that problem.

*PROPERTIES OF RECURSION*

* it performs the same operation multiple times with different inputs
* in every step, we try smaller inputs to make the problem smaller.
* base condition is needed to stop the recursion otherwise infinite loop will occur.

TYPES OF RECURSIONS

Direct recursion; this recursion occurs when a function calls itself as part of its execution or body

* Indirect recursion; this recursion requires at least two functions to call each other during their execution resulting in a more complex structure.
* *Tail recursion;* this is a type of recursive function where the function calls itself at the end(tail) of the function in which no computation is done after the return of a recursive call.
* *Mutual recursion;* this is a situation where two or more functions call each other recursively until a based Nested recursion; this is a recursion type where a recursive method has a parameter defined in terms of itself or when it passes the parameter as a recursive call.
* Linear recursion; this is a recursive function which makes only a single call to itself each time the function runs.
* Binary recursion; this is a recursive function which calls itself twice during the calls of its execution.
* Non-tail recursion; a recursive function is said to be non-tail recursion if the recursion call is not the last thing done by the function. After returning back, there is something left to evaluate.

**HOW TO SOLVE A PROBLEM USING RECURSIVE ALGORITHM**

* Step1; Initialize the algorithm that is; know what your function should do.
* Step2; Pick a sub-problem and assume your function already works on it.
* Step3; Take the answer to your sub-problem and use it to solve the original problem.
* Step4; Redefine the answer in terms of a smaller or simpler sub-problem

**SAMPLE EXAMPLES**

*some structural examples to demonstrate recursive algorithm*

1. to better understand the definition of recursion, we will look at the structure of a direct program.

int fun(int z) {

fun(z-1); //recursive call

}

in this program, you have a method named fun that calls itself again in its function body. thus, you can say that it is direct recursive.

2. to illustrate an indirect recursion, let us look at its recursive program structure

int fun1 (int z){ int fun2(int y){

fun2(z-1); fun1(y-2)

} }

in this example, we can see that the function fun1 explicitly calls fun2, which is invoking fun1 again. hence, you can say that this is an example of indirect recursion.

. let us look at the recursive structure to illustrate tail recursion

int fun (int z)

{

print (“%d”, z)

fun(z-1)

//recursive call is last executed statement

}

if you observe this program, you can see that the last line Adi will execute for method fun is a recursive call. and because of that, there is no need to remember any previous state of the program.

4. structural program of a non-tail recursion

int fun (int z)

{

fun(z-1);

print (“%d”, z);

//recursive call is not the last executed statement

in this function, we can observe that there is another operation after the recursive call. hence, the ADI will have to memorize the previous state inside this method block. that is why this program can be considered non tail recursion.