# RECURSIVE ALGORITHM

--- THUY NGUYEN ---

CS 131

11/30/2021

# What is recursive algorithm?

- Algorithm: a sequence of steps for solving a problem.
- Recursive algorithm: an algorithm that breaks problem into smaller subproblems and applies the same algorithm to solve those subproblems.

# Recursive process of painting the wall:

Paint wall:

- Paint the left side:
  - Paint the upper left
  - Paint the lower left
- Paint the right side:
  - Paint the upper right
  - Paint the lower right

# Recursive function

- Recursive function uses an if-else statement:
  - The if branch specify value of function at the initial value (basis step)
  - The else branch has recursive calls (recursive step)
- Recursive function will call itself repeatedly, minimizes the problem size each time.
- In order to terminate, the instant of problem must eventually be reduced to the initial case for which the solution is known.

# Give recursive algorithm for computing n! with n is a non-negative integer:

Procedure factorial (n: non-negative int)

if n=0 then return 1 → 0! = 1

else return n*factorial (n-1)

{output is n!}

n! = n(n-1)!
(n-1)! = (n-1)(n-2)!
…
=> Factorial function calls itself repeatedly until reaching factorial(n-n), which is factorial(0) = 0! = 1

n! = n(n-1)!
= n(n-1)(n-2)!
= …

Before writing a recursive function, a programmer should determine:

1. Whether the problem have a recursive solution
2. Whether the recursive solution is better than a non-recursive solution

## Non-recursive function

```
long factorial(int n) {
    int fac=1;
    for (int i = n; i > 1; i--)
        fac = fac * i;
    return fac;
}
```

## Recursive function

```
long factorial(int n){
    if (n = 0)
        return 1;
    else
        return n * factorial(n - 1);
```

=> In the example of computing n!, non-recursive function is easier to understand

# Give recursive algorithm for computing $a^n$ (a is nonzero real number, n is non-negative integer):

Procedure power (a: nonzero real number, n: non-negative integer)

if n=0 then return 1 → $a^0 = 1$ (base case)

else return a*power (a, n-1) → $a^n = a.a^{(n-1)}$

{output is $a^n$}

$a^n = a.a^{(n-1)}$

$= a.a.a^{(n-2)}$

$= a.a.a....a^{(n-n)}$

$= a.a.a....a^0$

$= a.a.a....1$ (recursive step stops at $a^0 = 1$, which is the base case)

# Give recursive algorithm for computing $a^2$ (a is nonzero real number, n is non-negative integer):

- Ex: a=2, n=4

$2^4 = 2.2^3$

$= 2.2.2^2$

$= 2.2.2.2^1$

$= 2.2.2.2.2^0$

$= 16$

**Fibonacci sequence** (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc.) :

1. $a_1 = 0, a_2 = 1$
2. $a_n = a_{n-1} + a_{n-2}$

# Computing the term $a_n$ in the Fibonacci sequence:

Procedure ComputeFibonacci (a1 = 0, a2 = 1, n: non-negative integer)

if n= 1 then return a1

else if n = 2 then return a2

else an = ComputeFibonacci (a2, a1+a2, n-1)

```cpp
#include <iostream>
using namespace std;

void ComputeFibonacci(int a1, int a2, int n) {
    if (n == 1) { cout << "=> Value of the desired term is " << a1 << endl; }

    else if (n == 2) { cout << "=> Value of the desired term is " << a2 << endl; }

    else {
        cout << a1 << " + " << a2 << " = " << a1 + a2 << endl;
        ComputeFibonacci(a2, a1 + a2, n - 1);
    }
}
```

```
int main() {
    int termN;
    char repeat;
    cout << "This program finds the nth term in the Fibonacci sequence by calculating step-by-step" << endl;

    do {
        termN = 0; //assign initial value for termN to activate the while loop

        while (termN < 1) {
            cout << "\n which term do you want to find? ";
            cin >> termN;  //user need to enter the valid termN (>=1)
        }

        cout << "0" << endl << "1" << endl; //first term is 0, second term is 1
        ComputeFibonacci(0, 1, termN);


        repeat = 'o'; //assign initial value for repeat to activate the while loop

        while (toupper(repeat) != 'Y' && toupper(repeat) != 'N') {
            cout << "Do you want to find another term? (Y/N)";
            cin >> repeat;
        }
    } while (toupper(repeat) == 'Y');
    return 0;
}
```

# Thank you!

- Thuy Nguyen -