

# Taller de Algoritmos y C++

## Estructuras de Datos

Luis Garreta

10 de agosto de 2017

Ingeniería de Sistemas y Computación  
Pontificia Universidad Javeriana – Cali

## 1. Introducción

Los algoritmos de Fuerza Bruta utilizan un enfoque "directo" para resolver problemas de una forma "fácil" y "simple", aunque puede ser ineficiente. Solo se busca resolver el problema considerando su declaración y definición sin hacer gran esfuerzo de hacer un algoritmo "inteligente"

## 2. Objetivos

- Analizar la complejidad de los siguientes algoritmos de fuerza bruta a través de conteo para determinar el peor caso (O grande).
- Aplicar técnicas de buen estilo de programación y documentación de software.
- Implementar los algoritmos en el lenguaje C++ siguiendo el esquema de división de archivos de librerías, es decir
  - Un archivo de encabezados con las declaraciones de las funciones (prototipos ) que implementan los algoritmos (de extensión .h)
  - Un archivo de implementación de con las funciones declaradas en archivo de encabezados (de extensión .cpp)
  - Un programa que use y llame cada una de esas funciones. El programa debe mostrar un menú con las opciones para probar cada uno de los algoritmos. Utilice una estructura *switch-case* donde en cada caso llame a una función que ejecuta la prueba del algoritmo:

```
#include <iostream>
using namespace std;
#include "algoritmos.h"

void probarPotencia ();
int main () {
    int opcion = -1;
    while (opcion != 0) {
        cout << endl << "Seleccione 0:Salir, 1:Potencia, 2:Busqueda. Cual opcion: ";
        cin >> opcion;
        switch (opcion) {
            case 0: return 0;
            case 1: probarPotencia (); break;
        }
    }
    return 0;
}
```

```

void probarPotencia () {
    int base, exponente, resultado;
    cout << "Ingrese base: ";
    cin >> base;
    cout << "Ingrese exponente: ";
    cin >> exponente;
    resultado = potencia (base, exponente);
    cout << "El resultado fue: " << resultado << endl;
}

```

### 3. Condiciones

- Grupos máximo de 2 personas, pero la evaluación se hace individual.
- Fecha de entrega: Martes 15 de Agosto
- Reporte de Análisis de los Algoritmos: puede ser en computador o a mano (bien escrito, legible y ordenado).
- Implementación: deben crear un repositorio en github (u otro) y tenerlos allí listos para revisión:
  - En cualquier momento se realizará una evaluación y usted simplemente los descarga sus archivos y los modifica de acuerdo a la evaluación.

### 4. Algoritmos a Implementar

#### 4.1. Exponente por fuerza bruta ( $a^n$ )

- Un algoritmo de fuerza bruta sería:

$$a^n = a * a * a * ... a$$

```

Algoritmo exponenciacion (base, exponente)
    resultado = 1
    for i=1 to exponente do
        resultado *= base

    return resultado

```

#### 4.2. Búsqueda exhaustiva por fuerza bruta

El algoritmo simplemente compara sucesivamente los elementos de un arreglo dado con una clave dada hasta encontrarla o llegar al final:

- Si la encuentra devuelve la posición, sino devuelve -1.
- El arreglo puede o no estar ordenado

```

Algoritmo BusquedaSecuencial (A[0..n], K)
    i = 0
    While i < n do
        if A [i] == K do
            return i
        i++

    return -1

```

### 4.3. Ordenamiento por Burbuja

- La operación básica de este algoritmo es la comparación  $A[j+1] < A[j]$  y el intercambio  $\text{swap } A[j] \text{ and } A[j+1]$ .

```
Algoritmo ordenamientoBurbuja (A[0..n-1])
for i=0 to n-2 do
  for j=0 to n-2-i do
    if A[j+1] < A[j]
      swap A[j] and A[j+1]
```

### 4.4. Ordenamiento por Selección

- Sobre el arreglo completo de  $n$  elementos se busca el elemento mínimo y se lo intercambia con el primer elemento del arreglo. Así el menor elemento ya ubicado. Entonces, otra vez se busca el menor elemento pero ahora desde la posición siguiente hasta la final (la parte no ordenada,  $n-1$  elementos) y se intercambia con el elemento a partir del cual se arrancó la búsqueda. Así el segundo elemento queda ya ordenada y se continua con el resto de elementos de la misma forma hasta  $n-1$ , ya que el ultimo elemento al final quedará organizado en su posición.

```
Algoritmo ordenamientoSeleccion (A[0..n-1])
for i=0 to n-2 do
  min=i
  for j=i + 1 to n-1 do
    if A[j] < A[min]
      min=j
  swap A[i] and A[min]
```

### 4.5. Emparejamiento de Cadenas

- El problema tiene dos entradas a considerarse:
  - Un texto largo de  $n$  caracteres y
    - Un patrón o cadena de  $m$  caracteres para ser buscado en el texto largo
  - El algoritmo:
    - Comienza alineando el patrón al inicio del texto.
    - Entonces cada carácter de el patrón se compara al correspondiente segmento del texto.
    - Si empareja entonces retorna la posición, se mueve al siguiente carácter hasta el final.

```
Algoritmo emparejamientoCadenas (T [0..n-1], P [0..m-1])
for i:=0..n-m do
  j:=0;
  while j < m AND P [j] == T [i + j] do
    j:=j + 1;
  end
  if j == m then
    return i;
  end
end
return -1;
```