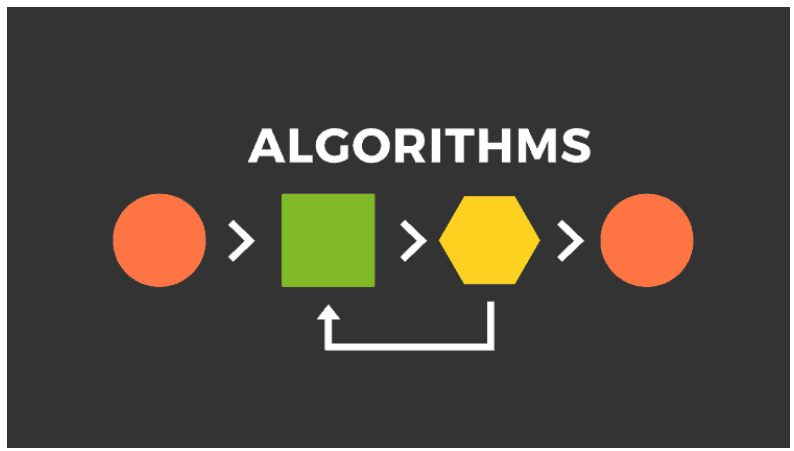
Algoritmos y Estructuras de Datos
Tema 1: Introducción al Desarrollo de Algoritmos

Grado Imat. Escuela ICAI

Juan C. Aguí García

January 2024



Qué es un algoritmo ? Una receta ?

How to Make a Chocolate Cake

Recipe

4 oz. chocolate
1 cup butter
2 cups sugar
3 eggs
1 tspn. vanilla
1 cup flour

Melt chocolate and butter. Stir into melted chocolate. Stir eggs and vanilla. Mix in flour. Spread mix in greased pan. Bake at 350º for 40 minutes or until inserted fork comes out almost clean. Cool in pan before eating.

Program code

Declare variables.
chocolate
butter
sugar
eggs
vanilla
flour
mix

mix = melted ((4*chocolate) + butter)
mix = stir (mix + (2*sugar))
mix = stir (mix `(3*eggs) + vanilla)
mix = mix + flour
spread (mix)

Algoritmo: su definición y características

In mathematics and computer science, an algorithm is a **finite sequence of rigorous instructions**, typically used to solve a class of specific problems or to perform a computation[1]

Un algoritmo debe ser :

Finito El algoritmo debe tener un inicio y un fin. Aunque la secuencia pueda ser infinita, debe tener criterios claves de terminación

[1] Wikipedia

Algoritmo: su definición y características

In mathematics and computer science, an algorithm is a **finite sequence of rigorous instructions**, typically used to solve a class of specific problems or to perform a computation[1]

Un algoritmo debe ser :

Finito El algoritmo debe tener un inicio y un fin. Aunque la secuencia pueda ser infinita, debe tener criterios claves de terminación

Preciso Debe ser lo suficientemente sencillo y debe carecer de ambiguedades que le permitan su ejecución de forma precisa

[1] Wikipedia

Algoritmo: su definición y características

In mathematics and computer science, an algorithm is a **finite sequence of rigorous instructions**, typically used to solve a class of specific problems or to perform a computation[1]

Un algoritmo debe ser :

Finito El algoritmo debe tener un inicio y un fin. Aunque la secuencia pueda ser infinita, debe tener criterios claves de terminación

Preciso Debe ser lo suficientemente sencillo y debe carecer de ambiguedades que le permitan su ejecución de forma precisa

Claras I/O El algoritmo debe describir con claridad los datos de entrada y los resultados, así como las condiciones o rangos de los datos de entrada han de cumplir.

[1] Wikipedia

Algoritmo: su definición y características

In mathematics and computer science, an algorithm is a **finite sequence of rigorous instructions**, typically used to solve a class of specific problems or to perform a computation[1]

Un algoritmo debe ser :

Finito El algoritmo debe tener un inicio y un fin. Aunque la secuencia pueda ser infinita, debe tener criterios claves de terminación

Preciso Debe ser lo suficientemente sencillo y debe carecer de ambiguedades que le permitan su ejecución de forma precisa

Claras I/O El algoritmo debe describir con claridad los datos de entrada y los resultados, así como las condiciones o rangos de los datos de entrada han de cumplir.

Implementable Debe ser posible su implementación en un tiempo razonable, y con las tecnologías disponibles

[1] Wikipedia

Algoritmo: su definición y características

In mathematics and computer science, an algorithm is a **finite sequence of rigorous instructions**, typically used to solve a class of specific problems or to perform a computation[1]

Un algoritmo debe ser :

Finito El algoritmo debe tener un inicio y un fin. Aunque la secuencia pueda ser infinita, debe tener criterios claves de terminación

Preciso Debe ser lo suficientemente sencillo y debe carecer de ambiguedades que le permitan su ejecución de forma precisa

Claras I/O El algoritmo debe describir con claridad los datos de entrada y los resultados, así como las condiciones o rangos de los datos de entrada han de cumplir.

Implementable Debe ser posible su implementación en un tiempo razonable, y con las tecnologías disponibles

Abstracto Independiente de los lenguages de programación. Debe ser posible implementarlo en cualquier lenguage (Java, C, Python, etc). Usamos pseudocódigo para su especificación

[1] Wikipedia

Specificación de Algoritmos: Pseudocódigo

La especificación de un algoritmo en pseudocódigo debe:

- Especificar los inputs y sus condicionantes
- Especificar los outputs
- Especificar la secuencia lógica de operaciones, utilizando sentencias de control aprobadas
- Añadiendo los comentarios pertinentes para el entendimiento del mismo
- Seguir un estándard aceptado[a]

[a] Ver Section 2 de Intro to Algorithms, 4th edition)

Ejemplo: Pseudocódigo para el Algoritmo de Herón

function HERON-SQRT(f,precision)
  In: f: float (greater than 0)
  Out: x_1:float
  if $f < 0$ **then**
    RaiseError InvalidRange
  end if
  $x_0 \leftarrow f/2$
  $e \leftarrow +\text{inf}$
  while $e >$ *precision* **do** ▷ Precision Test
    $x_1 \leftarrow 0.5 * (x_0 + f/x_0)$
    $e \leftarrow |x_1 - x_0|$
    $x_0 \leftarrow x_1$
  end while
  return x_1
end function

Con el mismo objetivo, y los mismos datos: Son todos los algoritmos iguales ??

NO! Algoritmos muy diferentes pueden, a partir de los mismos datos, obtener idénticos resultados

$\Rightarrow$ Pero su eficicencia en términos de Tiempo y Memoria (a.k.a **"Computational and Memory Complexity"**) pueden variar significativamente

Veamos un ejemplo sencillo: **La multiplicación de dos Números Enteros**. Consideremos:

1. Usar el operador * del python
2. Método Russo
3. Como lo hacíamos en el colegio, desplazando multiplicaciones parciales y sumando en columnas
4. Algoritmo básico, que acumula el multiplicando tantas veces como indica el multiplicador

Cuál es el mejor ?? $\rightarrow$ ver notebook

Algoritmos de Multiplicación: Resumen

Python Built-In Implementado en HW, (aún así es posible que haya un algoritmo básico detrás)
$\Rightarrow$ Muy rápido

Método Ruso El ciclo de cálculo itera proporcional al logaritmo (Divide Operation) del multiplicado
$\Rightarrow$ Muy rápido

Método Colegio Itera como el número de dígitos del multiplicando.
$\Rightarrow$ Equivalente al Método Russo)

Algoritmo Básico Itera como el valor del multiplicando.
$\Rightarrow$ Muy Lento

Otro ejemplo: Ordenación Ingenua

Adelantaremos parte del tema del curso, la Ordenación, a.k.a. **Sorting**
Consideraremos de entrada, el peor y uno de los mejores algoritmos:

1. **Naive Insertion Sort**. Para todos los elementos, busca en los siguientes uno mayor, y si lo encuentra, los intercambia. Así hasta el final
2. Llamar a la función `sort()` **de Python**. Detrás hay una librería en C, de uso general, pero con muchos años de experiencia a las espaldas[2]

$\Rightarrow$ Ver notebook

[2] Ver el Timsort algorithm, ver `https://en.wikipedia.org/wiki/Timsort`

Otro ejemplo: Ordenación Ingenua

Adelantaremos parte del tema del curso, la Ordenación, a.k.a. **Sorting**
Consideraremos de entrada, el peor y uno de los mejores algoritmos:

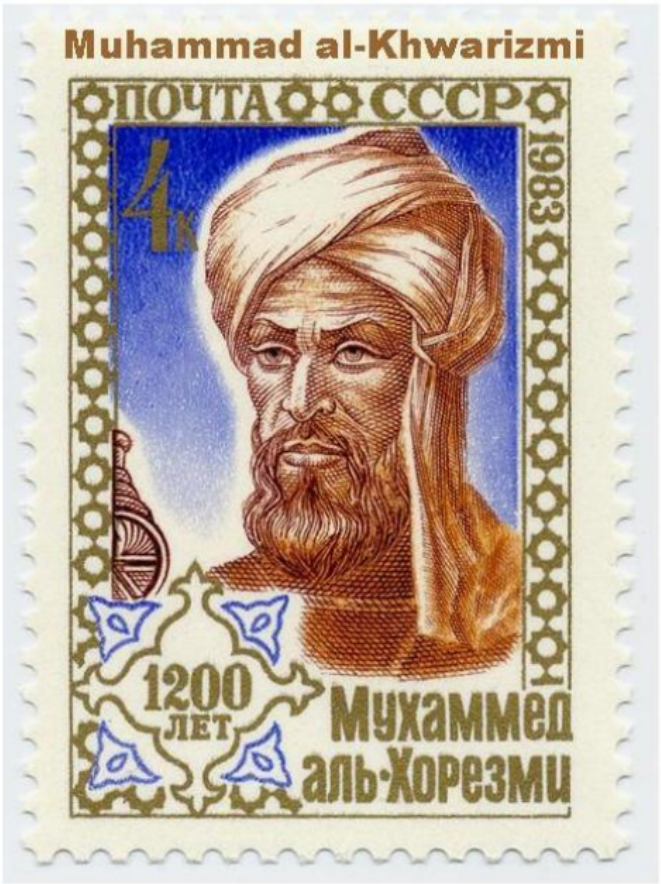
1. **Naive Insertion Sort**. Para todos los elementos, busca en los siguientes uno mayor, y si lo encuentra, los intercambia. Así hasta el final
2. Llamar a la función `sort()` **de Python**. Detrás hay una librería en C, de uso general, pero con muchos años de experiencia a las espaldas[2]

$\Rightarrow$ Ver notebook

Python sort es mejor que cualquier implementación "casera" por varios órdenes de magnitud. Primero: aprender el qué y como de los algoritmos básicos, como Sort, luego usar las librerías disponibles, y sólo en casos especiales, implementar la tuya !

[2] Ver el Timsort algorithm, ver `https://en.wikipedia.org/wiki/Timsort`

Eof Tema 1: Gracias !



"May the Algorithm's Force be with you."

COMILLAS
ICAI