Algorítmica

Tema 1: La Eficiencia de los Algoritmos

- Planteamiento general
  - Ciencia de la Computación
  - El concepto de algoritmo
  - Elección de un algoritmo
  - Problemas y casos
  - Distintos tipos de casos

Lo que no es la Ciencia de la Computación

La Ciencia de la Computación es el estudio de los computadores

.

La Ciencia de la Computación es a los computadores lo mismo que la Astronomia es a los telescopios, la Biologia a los microscopios o la Quimica a las pipetas.

La Ciencia no trata sobre sus herramientas, sino sobre como usarlas y sobre lo que descubrimos cuando las usamos

La Ciencia de la Computación es el estudio de cómo escribir programas de computador.

La programación es una parte muy importante de la Ciencia de la Computación. Pero es una herramienta para implementar ideas y soluciones. Un programa es un medio para lograr un objetivo, pero no un objetivo.

La Ciencia de la Computación estudia los usos y aplicaciones de computadores y software.

El estudio de cómo usar un computador/software es una parte de la Ciencia de la Computación igual que el manejo de un movil es una parte de la Ingenieria de Telecomunicaciones

¿Entonces qué es la Ciencia de la Computación?

Las definiciones anteriores:

- Aunque no necesariamente (completamente) equivocadas, son incompletas, y por tanto equívocas

Una definición más precisa:

La Ciencia de la Computación es el estudio de los Algoritmos, incluyendo sus propiedades, su hardware y consideraciones lingüísticas, y sus aplicaciones.

¿Qué es un algoritmo?

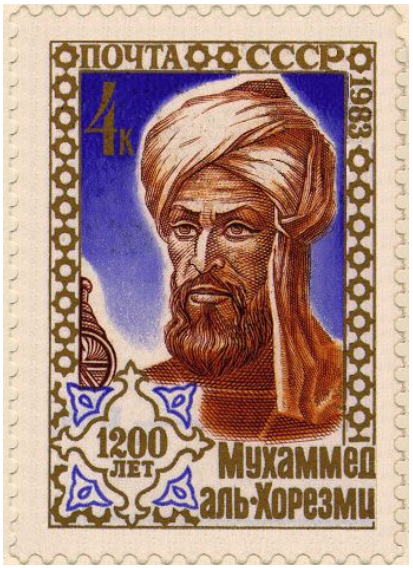
Definición formal:

- Una secuencia ordenada de pasos, exentos de ambigüedad, tal que al llevarse a cabo con fidelidad, dará como resultado que se realice la tarea para la que se ha diseñado, (se obtenga la solución del problema planteado) en un tiempo finito.

O, más informalmente:

Un método por etapas para realizar alguna tarea.

¿De dónde provienen?



Etimología.

- "algos" = pena, sufrimiento en griego.
- "algor" = frio en latín.
- “arithmos” = número en griego
- degeneración de “logaritmo”

Muhammad ibn Musa (Al-Khuwarizmi) fue un matemático persa (siglo 9 adc)

Su libro "Al-Jabr wa-al-Muqabilah" ("Algoritmi de Numero Indorum”) es la clave

“Algoritmos” en la vida diaria

Los algoritmos no están limitados a realizar tareas matemáticas.

Usamos los “algoritmos” todos los días, por ejemplo:

- Cocinando recetas de cocina.
- Siguiendo instrucciones para montar aparatos.
- Indicaciones para llegar a sitios.
- Al realizar tareas rutinarias.

Un algoritmo no es una receta

Además de ser un conjunto finito de reglas que definen una secuencia de operaciones para resolver un tipo dado de problema, tiene cinco características primordiales

- a) Finitud: terminar después de un número finito de etapas.
- b) Especificidad: cada etapa debe estar precisamente definida; las acciones que hay que llevar a cabo deben estar rigurosamente especificadas para cada caso.
- c) Input: Un algoritmo tiene cero o más inputs.
- d) Output: uno o más outputs.
- e) Efectividad: todas las operaciones que hay que realizar deben ser tan básicas como para que se puedan hacer exactamente y en un periodo finito de tiempo usando solo lápiz y papel (tecnología?)

Tecnología

Aquí entendemos por *Agente Tecnológico* cualquier ente capaz de realizar las etapas que describe el algoritmo, es decir, capaz de ejecutar el algoritmo.

Por tanto, puede ser:

- Persona
- DNA (!)
- Computador

En nuestro caso, generalmente, será un computador.

La Ciencia de la Computación trata sobre

- 1) Propiedades formales y Matemáticas

  Cómo diseñar algoritmos para resolver una gran variedad de problemas.

  Cómo determinar si los problemas son (eficientemente) computables, es decir, ¡si se pueden especificar por un algoritmo!

  Estudiar la conducta de los algoritmos para decidir si trabajan correctamente y con cuanta eficiencia.

- 2) Hardware

  El diseño y construcción de equipos capaces de ejecutar algoritmos.

  Los incesantes avances tecnologicos:

  Computadores cada vez mas rápidos, redes, …

  Computación paralela

  “Computación cuántica, molecular...”?

La Ciencia de la Computación trata sobre

- 3) Reglas lingüísticas
  - El diseño de lenguajes de programación y de la traducción a estos lenguajes de los algoritmos para que el hardware disponible pueda ejecutarlos.
  - Programación funcional, Programación orientada a objetos, Programación visual, …
- 4) Aplicaciones cada vez más novedosas
  - Identificar nuevos problemas importantes para los computadores y del diseño del software que los resuelva.
  - Los primeros computadores se usaron sobre todo para cálculos numéricos y tratamiento masivo de datos.
  - Ahora, … se usan en negocios, grafismo, multimedia, domótica, Internet, WWW, … ¿qué será lo siguiente?

¿Cómo se estudia todo esto?

Algorítmica

El estudio de los algoritmos incluye el de diferentes e importantes áreas de investigacion y docencia, y se suele llamar Algorítmica

La Algoritmica considera:

- La construcción
- La expresión
- La validación
- El análisis, y
- El test de los programas

1. La construcción de algoritmos

El acto de crear un algoritmo es un arte que nunca debiera automatizarse.

Pero, indudablemente, no se puede dominar si no se conocen a la perfección las técnicas de diseño de los algoritmos.

El área de la construcción de algoritmos engloba el estudio de los métodos que, facilitando esa tarea, se han demostrado en la práctica más útiles.

2. La expresión de algoritmos

Los algoritmos han de tener una expresión lo más clara y concisa posible.

Como este es un tema clave del estudio de los algoritmos, requerirá que se le dedique tanto esfuerzo como sea necesario, a fin de conseguir lo que se suele denominar un buen estilo.

3. Validación de algoritmos

Cuando se ha construido el algoritmo, se hace necesario demostrar que calcula correctamente sobre inputs legales.

Este proceso se conoce con el nombre de validación del algoritmo.

La validación persigue asegurar que el algoritmo trabajará correctamente independientemente del lenguaje empleado, o la tecnología usada.

Cuando se ha demostrado la validez del método, es cuando puede escribirse el programa, comenzando una segunda fase del trabajo (la verificación del programa).

4. Análisis de algoritmos

El análisis de algoritmos se refiere al proceso de determinar cuánto tiempo de cálculo y cuánto almacenamiento requerirá un algoritmo.

Nos permite hacer juicios cuantitativos sobre el valor de un algoritmo sobre otros.

A menudo tendremos varios algoritmos para un mismo problema. Habrá que decidir cual es el mejor o cual es el que tenemos que escoger, según algún criterio prefijado, para resolverlo.

Puede permitirnos predecir si nuestro software necesitará algún requisito especial.

… y, 5. Test de los programas

El test de un programa es la última fase que se lleva a cabo.

Básicamente supone

- la corrección de los errores que se detectan, y
- la comprobación del tiempo y espacio que son necesarios para su ejecución.

Elección de un algoritmo

Supongamos que ante un cierto problema tenemos varios algoritmos para emplear.

¿Qué algoritmo elegir?.

Queremos buenos algoritmos en algún sentido propio de cada usuario.

El problema central que se quiere resolver es el siguiente:

Dado un algoritmo, determinar ciertas características que sirven para evaluar su rendimiento.

Ejemplo

- Se tienen *n* bolas de igual tamaño, todas ellas de igual peso salvo una más pesada. Como único medio para identificar esta bola singular se dispone de una balanza romana clásica (sólo indica si algo pesa más, menos o igual que otra cosa).
- Diseñar un algoritmo que permita determinar cuál es la bola más pesada.

Ejemplo: Elementos duplicados

- Sea A[1..n] un vector de enteros.
- Determinar si existen dos índices distintos i y j, tales que A[i] = A[j]
- Diseñar un algoritmo para solucionar el problema.

Elección de un algoritmo

Multiplicación de enteros

Algoritmo clásico

Algoritmo de multiplicación a la rusa

Escribir multiplicador y multiplicando en dos columnas

Hasta que el número bajo el multiplicador sea 1

- REPETIR:

  Dividir el número bajo el multiplicador por 2, ignorando los decimales

  Doblar el número bajo el mutiplicando sumándolo a sí mismo

  Rayar cada fila en la que el número bajo el multiplicador sea par, o añadir los números que queden en la columna bajo el multiplicando.

Elección de un algoritmo

- 1) Escribir multiplicador y multiplicando en dos columnas
- 2) Repetir las siguientes operaciones hasta que el número bajo el multiplicador sea 1:
- a) Dividir el número bajo el multiplicador por 2, ignorando los decimales
- b) Doblar el número bajo el mutiplicando sumándolo a sí mismo
- c) Rayar cada fila en la que el número bajo el multiplicador sea par, o añadir los números que queden en la columna bajo el multiplicando.

Multiplicador	Multiplicando	Resultado	Resultado acumulado
45	19	19	19
22	38	--	19
11	76	76	95
5	152	152	247
2	304	---	247
1	608	608	**855**

Elección de un algoritmo

Posibles criterios para la elección son:

La adaptabilidad del algoritmo a los computadores

Su simplicidad y elegancia

El costo económico que su confección y puesta a punto puede acarrear.

La duración del tiempo consumido para llevar a cabo el algoritmo (esto puede expresarse en términos del número de veces que se ejecuta cada etapa).

Problemas y casos

(19,45) es un Caso del Problema de multiplicar dos enteros positivos

Los problemas más interesantes incluyen una colección infinita de casos (¿ajedrez?).

Un algoritmo debe trabajar correctamente en cualquier caso del problema en cuestión.

Para demostrar que un algoritmo es incorrecto, solo necesitamos encontrar un caso del problema que no produzca la respuesta correcta.

La diferencia que existe entre algoritmo y programa es que cualquier sistema de cómputo real tiene un límite sobre el tamaño de los casos que puede manejar. Sin embargo, este límite no puede atribuirse al algoritmo que escojamos para usar.

Problemas y casos

Formalmente:

El tamaño de un caso x, notado por $|x|$, es el número de bits necesarios para representar el caso en un computador usando un código precisamente definido y razonablemente compacto.

Informalmente:

El tamaño de un caso es cualquier entero que, de algún modo, mida el número de componentes del caso.

Ejemplos:

Ordenación: longitud del array, Matrices: Número de filas y columnas, Grafos: Número de vértices y arcos

Diferentes tipos de casos

El tiempo consumido por un algoritmo puede variar mucho entre dos casos diferentes del mismo tamaño.

Consideremos dos algoritmos de ordenación elementales: Insercion y Seleccion.

Procedimiento Insercion

```
(T[1..n])
for i := 2 to n do
    x:= T[i]; j := i-1
    while j > 0 and x < T[j] do
                    T[j+1] := T[j]
                     j := j-1
    T[j+1] := x
```

Procedimiento Seleccion

```
(T[1..n])
for i:= 1 to n-1 do
    minj := i; minx := T[i]
    for j := i+1 to n do
       if T[j] < minx then minj := j
                           minx := T[j]
    T[minj] := T[i];
    T[i] := minx
```

Diferentes tipos de casos

U y V dos arrays de n elementos: U ordenado en orden ascendente y V en orden descendente.

Comportamiento de Seleccion:

Indiferente de U o V, el tiempo requerido para ordenar con Seleccion no varía en mas de un 15%.

Comportamiento de Insercion:

Insercion(U) consume menos de 1/5 de segundo si U es un array de 5.000 elementos

Insercion(V) consume tres minutos y medio si V es un array de 5.000 elementos.

Si pueden darse esas diferencias, ¿Podremos hablar del tiempo consumido por un algoritmo solo en función del tamaño del caso?

Diferentes tipos de casos

El tiempo que se obtiene para V da el máximo tiempo que se puede emplear para resolver el problema: V describe el *peor caso*.

El peor caso es útil cuando necesitamos una garantía total acerca de lo que durará la ejecución de un programa. El tiempo del peor caso, para un n dado, se calcula a partir del caso de tamaño n que tarda más.

Si el problema se ha de resolver en muchos casos distintos, es más informativo el tiempo del *caso promedio*: la media de los tiempos de todos los posibles casos del mismo tamaño (¡hay que conocer la distribución de probabilidad asociada a los casos!)

U, sin embargo, describe el mejor caso de este problema, es decir, el caso sobre el que se tarda menos tiempo.

Diferentes tipos de casos

Consideramos el Tiempo de Ejecución,

Tiempo del Peor Caso:

- La función definida por el *máximo* número de etapas que se realizan en cualquier caso de tamaño n

Tiempo del Mejor Caso:

- La función definida por el *mínimo* número de etapas que se realizan en cualquier caso de tamaño n

Tiempo del Caso promedio:

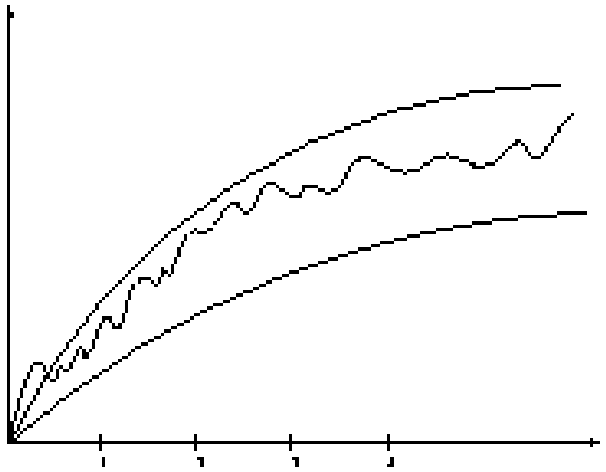
- La función definida por el número *medio* de etapas que se realizan en cualquier caso de tamaño n

Diferentes tipos de casos

Es difícil estimar exactamente el tiempo de ejecución

- El mejor caso depende del input
- El caso promedio es difícil de calcular
- Por tanto generalmente nos referiremos al Peor Caso
  - Es fácil de calcular
  - Suele aproximarse al tiempo de ejecución real

La Estrategia a seguir: intentar encontrar cotas inferiores y superiores para el tiempo de ejecución del peor caso.



Cota Superior

Tiempo Real

Cota Inferior