Tema 1: Eficiencia de los algoritmos

¿Qué es la ciencia de la Computación?

El estudio de los algoritmos incluyendo sus **propiedades**, su **hardware**, sus **aspectos lingüísticos** y sus **aplicaciones**.

- Propiedades: diseñar algoritmos, estudiar si se pueden especificar con un algoritmo y estudiar su eficiencia
- Hardware: diseño y construcción de los equipos que ejecutan el algoritmo
- Aspectos lingüísticos: diseñar el código
- Aplicaciones: identificar nuevos problemas a solucionar

¿Qué es un algoritmo?

Una **secuencia finita** y **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 con **recursos limitados y en tiempo finito**.

- Definición según **Knuth**
  Un algoritmo es un método computacional que termina en un número finito de etapas $\forall x \in I$
- Diferencia entre algoritmo y programa
  Cualquier sistema de cómputo real (computador) tiene un límite sobre el tamaño de los casos que puede manejar. Sin embargo, este límite no le afecta al algoritmo.

¿Qué es un método computacional?

Un método computacional es una cuaterna (Q, I, Ω, f) en la que $I \subseteq Q$ y $\Omega \subseteq Q$ y $f: Q \to Q$ tal que $f(a) = a \quad \forall a \in \Omega$ y donde Q es el conjunto de los estados del cálculo, I es el input, Ω el output y f la regla de cálculo que se esté usando.

Cada input $x \in I$ define una sucesión computacional x_0, x_1, x_2… como sigue:

$$x_o = x \text{ y } x_{k+1} = f(x_k)$$

Características de un algoritmo

1. **Finitud**
2. **Especificidad**
3. **Input**: cero o más
4. **Output**: uno o más
5. **Efectividad**: operaciones básicas para hacerlas en un periodo finito de tiempo (papel y lápiz)

¿Qué es la algorítmica?

El estudio de los algoritmos incluyendo su construcción, expresión, validación, análisis y el test de los programas.

- Construcción: estudiar los métodos que se ha demostrado que en la práctica son más útiles
- Expresión: clara y concisa para conseguir un buen estilo

- Validación: demostrar que calcula correctamente sobre inputs legales independientemente del lenguaje y la tecnología usada
    - A partir de aquí, se escribe el programa
- Análisis: determinar cuánto tiempo de cálculo y almacenamiento requerirá y comparar el algoritmo con el resto.
- Test de los problemas: corrección de errores

Tema 2: Tiempo de ejecución

3 métodos de calcular la eficiencia

- Enfoque **empírico**: depende del agente tecnológico
- Enfoque **teórico**: no depende del agente tecnológico
- Enfoque **híbrido**: la forma de la función se determina teóricamente y los parámetros de la función se determinan empíricamente.

Principio de Invarianza

Dos implementaciones diferentes de un mismo algoritmo no difieren en eficiencia más que, a lo sumo, en una constante multiplicativa.

Dos implementaciones consumen $t_1(n)\ t_2(n)$ unidades de tiempo. Siempre existe C tal que $t_1(n) \leq C t_2(n)$ para n suficientemente grande.

Tiempo de ejecución

No depende del input sino del tamaño de este

La notación O y los límites

Se dice que f es de orden g, y se denota por $f = O(g(n))$, si existen $C,\ k > 0$ tales que

$$\forall n \geq k,\ f(n) \leq C\ g(n)$$

Lema: Si existe el límite cuando n tiende a infinito de $|f(n)/g(n)|$ entonces $f(n) = O(g(n))$

Observación: se puede pensar de O como $\leq$.

Notaciones Ω y Θ

$f(n) = \Omega(g(n)) \Leftrightarrow g(n) = O(f(n))$

"la notación Θ indica que f es de orden exacto g"

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) = \Omega(g(n))$$

Observación: se puede pensar de Ω como $\geq$, y de Θ, como $=$.

Tasa de Crecimiento

Por ejemplo, cuando el tamaño del input aumenta, el algoritmo cúbico tardará más que el cuadrático. También puede haber funciones incomparables.

Notación asintótica de Brassard y Bratley

Sea $f: N \rightarrow R^*$. Definimos

$$O(f(n)) = \{t: N \to R^{*}: \exists c > 0, \exists n_0 > 0: \forall n \geq n_0 \Rightarrow t(n) \leq cf(n)\}$$

$$\Omega(f(n)) = \{t: N \to R^{*}: \exists c > 0, \exists n_0 > 0: \forall n \geq n_0 \Rightarrow t(n) \geq cf(n)\}$$

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

- Notación asintótica condicional
  $O(f(n)/P(n)) = \{t: N \to R^{*}: \exists c > 0, \exists n_0 > 0: \forall n \geq n_0, P \Rightarrow t(n) \leq cf(n)\}$

Cálculo de la eficiencia

Supongamos que tenemos dos segmentos de programa, P_1 y P_2 cuyos tiempos son $T^1(n)$ y $T^2(n)$, y que $T^1(n)$ es $O(f(n))$ y $T^2(n)$ es $O(g(n))$. Entonces

- $T^1(n) + T^2(n) = O(max(f(n), g(n)))$
- $T^1(n) \cdot T^2(n) = O(f(n) \cdot g(n))$
- $O(cf(n)) = O(f(n))$ (las constantes no importan)
- $O(a_m n^m + \cdots + a_1 n + a_0) = O(n^m)$

Algoritmos de fuerza bruta

La fuerza bruta es un enfoque directo para resolver un problema, que se basa exclusivamente en el planteamiento del problema y en las definiciones y conceptos que intervienen en el mismo. No recurre a algoritmos, métodos, procedimientos o técnicos que no vayan incorporadas en el problema en sí mismo.

Características:

- Gran aplicabilidad, simplicidad
- Son algoritmos estándar para realizar tareas simples
- No suelen ser eficientes (algunos son muy lentos)
- No es ni constructivo, ni creativo

Algoritmos de Búsqueda Exhaustiva (Algoritmos combinatorios)

Son algoritmos de fuerza bruta para problemas en los que hay que buscar algún elemento con una propiedad específica. Frecuentes en problemas de grafos, optimización, etc.

Método:

- Se hace una lista con todas las posibles soluciones
- Se evalúan una por una las “soluciones” de la lista, desechando las infactibles, hasta que solo quede una

Características:

- Generalmente poco eficientes
- Suele haber otros algoritmos mejores (no siempre)

Problemas P y NP

Se dice que un algoritmo está en la **clase P** cuando su tiempo de ejecución es polinomial.

Hay otros algoritmos para los que la única forma de que tengan un tiempo polinomial es realizando una etapa aleatoria (incluyendo el azar de alguna manera). Estos son los algoritmos de la **clase NP**. También se define la clase NP como el conjunto de algoritmos que, dada una posible solución, se puede comprobar en un tiempo polinomial que, de hecho, es una solución.

- Clase P ⊂ clase NP
- Si $P = NP$ es un problema abierto (no se sabe, aunque parece que no)

Tema 3: Resolución de recurrencias asintóticas