Guió Anàlisi d'Algorismes I

EDA – Grup 30 – FIB

20 de setembre de 2020

Continguts:

- Cost en temps i en espai.
- Cost en cas pitjor, millor, mitjà.
- Notacions asintòtiques $\mathcal{O}, \Omega, \Theta$.

1 Cost en temps i en espai

La pregunta que sempre ens fem com a programadors és, donat un programa, ho puc fer millor? Més eficient? Més ràpid? Més senzill? Per poder respondre hem de ser capaços de comparar programes/algorismes entre ells i determinar quin és millor.

Però què vol dir que un algorisme sigui millor que un altre? Per poder respondre hem de definir que vol dir que un algorisme sigui bo/dolent, és a dir, què tan eficient és, el que només te sentit en termes relatius (per comparar) i no absoluts.

Definim la **eficiència** d'un algorisme com la quantitat de recursos que utilitza del sistema i aquests recursos es mesuren en temps (què tan ràpid és) i en espai (quina és la quantitat de mèmoria que requereix). A la eficiència en temps li diem: **cost en temps**, a la eficiència en espai: **cost en espai**.

Evidentment aquests costos es poden mesurar de moltes maneres. Una possibilitat és experimentalment, programant l'algorisme i mesurant el temps que triga per resoldre cada instància del problema que resol. Quins són els inconvenients de fer-ho d'aquesta manera? Que depenem del llenguatge de programació utilitzat, del sistema operatiu que fem servir, del tipus de màquina, etc...

Com que volem que les nostres mesures siguin independents dels factors anteriors i a més volem poder predir el comportament dels nostres algorismes, mesurarem l'eficiència dels nostres algorismes de manera formal, matemàticament.

Com mesurem el cost d'un algorisme? El cost en temps: com el nombre d'operacions elementals que es requereixen per executar-lo. El cost en espai: com la quantitat de memòria que es requereix per executar-lo.

I què són les operacions elementals? Les sumes, restes, multiplicacions, divisions, compatracions, asignacions, pas de paràmetres, lectura/escriptura d'elements bàsics, indivisibles com ara enters, reals, caracters, etc...

Exemple 1): Algorisme d'ordenació per selecció. Input: Vector de n elements. Fem tres tipus d'operacions amb els elements del vector: comparacions, intercanvis, a més dintre dels bucles fem asignacions, increments i intercanvis. Si comptem el nombre de comparacions: $n-1$ per trobar el primer element, $n-2$ per trobar el segon, etc., en total, de l'ordre de n^2. Si comptem el nombre d'intercanvis, fem de l'ordre de n intercanvis Per a n's grans, domina el nombre de comparacions i direm que el cost de l'algorisme és de l'ordre de n^2. El cost en espai és de l'ordre de n.

Llavors definirem els cost d'un algorisme com una funció T que a tota possible entrada de l'algorisme li associa un valor: el seu cost en temps/espai.

$$T : A \to \mathbb{R}^+, a \mapsto T(a)$$

Però si ho deixem així és massa general i estaríem comparant costos sobre vectors de diferents mides, el que seria poc útil. Per resoldre aquest problema es restringeix la funció T a entradas de la mateixa mida, és a dir:

$$T_n : A_n \to \mathbb{R}^+, a_n \mapsto T_n(a_n)$$

Exemple 2): Algorisme d'ordenació per inserció. Cas millor: $n-1$ comparacions i cap intercanvi. Cas pitjor: de l'ordre de n^2 comparacions i n intercanvis.

Exemple 3): Cerca dicotòmica. Cas millor: nombre constant de comparacions. Cas pitjor: de l'ordre de $\log(n)$ comparacions.

2 Cost en cas pitjor, millor, mitjà

Definim 3 tipus de costos:

- Cost en cas pitjor: $T_{pitjor}(n) = \max\{T_n(a_n) : a_n \in A_n\}$

- Cost en cas millor: $T_{millor}(n) = \min\{T_n(a_n) : a_n \in A_n\}$
- Cost en cas mitjà: $T_{mitja}(n) = \sum_{a_n \in A_n} Pr_(a_n) \times T_n(a_n)$

3 Notacions asintòtiques $\mathcal{O}, \Omega, \Theta$

A més, necessitem una manera més pràctica de referir-nos al cost quan diem *proporcional a* o *de l'ordre de, i per això es fa servir la notació asintòtica.*

$\mathcal{O}(f)$**:** *Sigui* $f : \mathbb{N} \to \mathbb{R}^+$,

$$\mathcal{O}(f) = \{g : \mathbb{N} \to \mathbb{R}^+ : \exists c > 0, \exists n_0 \in N, \forall n > n_0, g(n) <= cf(n)\}$$

$\mathcal{O}(f)$ es la classe de funcions que tenen un creixement asintòtic (per a n's grans) com a molt igual de ràpid que el de f (potser més lent). Explicació de la definició amb gràfica de n i n^2 amb $c = 1$.

Exemples: *(Explicar abús de notació)*

- $\mathcal{O}(1) = \{1/n, 57, 1024, ...\}$
- $\mathcal{O}(n) = \{n, \sqrt{n}, n + 12, 2n - 52, \log(n), 45, ...\}$
- $\mathcal{O}(n^3) = \{n, n\sqrt{n}, n^2 - 17n - 3, \log^2(n), ...\}$

Propietat important: *Fixeu-vos que si* $\lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ *llavors* $g = \mathcal{O}(f)$.

$\Omega(f)$**:** *"simètric"de* $O(f)$.

Propietat important: *Fixeu-vos que si* $\lim_{n\to\infty} \frac{g(n)}{f(n)} > 0$ *llavors* $g = \Omega(f)$.

$\Theta(f)$**:** $\Theta(f) = O(f) \cap \Omega(f)$.

Propietat important: *Fixeu-vos que si* $\lim_{n\to\infty} \frac{g(n)}{f(n)} = c > 0$ *llavors* $g = \Theta(f)$ *i* $f = \Theta(g)$.