Guió Anàlisi d'Algorismes II
EDA – Grup 20 – FIB

Amalia Duch

September 22, 2020

Contents

- Regles per calcular el cost d'agorismes iteratius.
- Exemples
- Regles per calcular el cost d'algorismes recursius.
- Exemples
- Fita inferior per als algorismes d'ordenació basats en comparacions.

Anàlisi d'Algorismes Iteratius

1. El cost de les operacions elementals (veure classe anterior) és $\Theta(1)$.
2. Composició seqüencial: Donats dos fragments de codi `s1`, `s2` amb cost f_1 i f_2 respectivament, el cost del fragment:

```
1 s1;
2 s2;
```

   es $f_1 + f_2$.

3. Composició Alternativa (condicional): Sigui A una expressió amb cost per avaluar-la f_a, i `s1` i `s2` dos fragments de codi amb cost f_1 i f_2 respectivament, el cost, en cas pitjor, del fragment:

```
if(A) {
 s1;
}
else {
 s2;
}
```

es $f_a + \max\{f_1, f_2\}$.

4. Composició Iterativa: Sigui A una expressió amb cost per avaluar-la a la i-ésima iteració g_i, i `s` un fragment de codi amb cost a la i-ésima iteració f_i, el cost, en cas pitjor, del fragment:

```
while (A) {
 s;
}
```

si es realitzen n iteracions es $\sum_{i=1}^{n}(f_i + g_i) = O(n(f + g))$ amb $f = \max f_i$ i $g = \max g_i$.

Exemples

Càlcul del cost de differents bucles.

Anàlisi d'Algorismes Recursius

El cost dels algorismes recursius ve descrit per una recurrència. Resolent aquestes recurrències obtenim el cost de l'algorisme en qüestió. Generalment, quan analitzem algorismes, trobem dos tipus de recurrències: sustractores i divisores, que podem resoldre mitjançant teoremes mestres.

Teorema mestre per a recurrències sustractores:

Theorem 1. *Sigui $T(n)$ el cost (en cas pitjor, millor, mitjà, ...) d'un algorisme recursiu que satisfà la recurrència:*

$$T(n) = \begin{cases} f(n) & \text{if } 0 \leq n < n_0 \\ a \cdot T(n-c) + g(n) & \text{if } n \geq n_0, \end{cases}$$

amb n_0 constant, $c \geq 1$, $f(n)$ una funció arbitrària i $g(n) = \Theta(n^k)$ per a una constant $k \geq 0$. Llavors,

$$T(n) = \begin{cases} \Theta(n^k) & \text{if } a < 1 \\ \Theta(n^{k+1}) & \text{if } a = 1 \\ \Theta(a^{n/c}) & \text{if } a > 1. \end{cases}$$

Teorema mestre per a recurrències divisores:

Theorem 2. *Sigui $T(n)$ el cost (en cas pitjor, millor, mitjà, ...) d'un algorisme recursiu que satisfà la recurrència:*

$$T(n) = \begin{cases} f(n) & \text{if } 0 \leq n < n_0 \\ a \cdot T(n/b) + g(n) & \text{if } n \geq n_0, \end{cases}$$

amb n_0 constant, $b > 1$, $f(n)$ una funció arbitrària i $g(n) = \Theta(n^k)$ per a una constant $k \geq 0$. Sigui $\alpha = \log_b a$. Llavors,

$$T(n) = \begin{cases} \Theta(n^k) & \text{if } \alpha < k \\ \Theta(n^k \log n) & \text{if } \alpha = k \\ \Theta(n^\alpha) & \text{if } \alpha > k. \end{cases}$$

Exemples

- Cerca dicotòmica: $\Theta(\log(n))$ en cas pitjor
- Exponenciació ràpida: $\Theta(\log(n))$
- Nombres de Fibonacci: $\mathcal{O}(2^n)$ i $\Omega(2^{(n/2)})$