Lezione 3

3.1 Metodo master

$$\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$, con $a>0$, $b>1$ costanti

chiameremo un'equazione di ricorrenza in questa forma equazione di ricorrenza in forma master.

OSS: Potrebbe esse $b \nmid n$, $a'T\left(\lfloor \frac{n}{b} \rfloor\right) + a''T\left(\lceil \frac{n}{b} \rceil\right)$, $\exists\, a', a'' > 0 : a' + a'' = a$.

DRIVING FUNCTION: $f(n)$, WATERSHED FUNCTION: $w(n) = n^{\log_b a}$

THM MASTER Siano $a>0$, $b>1$ costanti, sia $f(n)$ funzione definita e nonnegativa su tutti i numeri reali sufficientemente grandi.

Sia
$$T(n) = aT\left(\frac{n}{b}\right) + f(n).$$

Allora:

1. $\exists\, \varepsilon > 0 : f(n) = O\left(n^{\log_b a - \varepsilon}\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$;
2. $\exists\, k \geq 0 : f(n) = \Theta\left(n^{\log_b a} \log^k n\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a} \cdot \log^{k+1} n\right)$;
3. $\exists\, \varepsilon > 0 : f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) + f(n)$ soddisfa la condizione di regolarità: $\left(\exists\, c < 1 : af\left(\frac{n}{b}\right) \leq c f(n)\right)$ $\Rightarrow T(n) = \Theta(f(n))$.

Discussione

		(TREE)
caso 1	$w(n)$ cresce in maniera asintoticamente più veloce rispetto a $f(n)$, ed esse differiscono per un polinomio $\Theta(n^{\varepsilon})$	Il costo dei livelli cresce in maniera almeno geometrica dalla radice alle foglie e il costo delle foglie domina il costo totale dei nodi interni.

caso 2	$w(n)$ cresce asintoticamente in maniera simile a $f(n)$ ed esse differiscono per un fattore polilogaritmico $\Theta(\log^k n)$	Tutti i livelli dell'albero hanno approssimativamente lo stesso costo $\Theta(n^{\log_b a} \log^k n)$ e ci sono $\Theta(\log n)$ di essi.
caso 3	$w(n)$ cresce in maniera asintoticamente più lente di $f(n)$ ed esse differiscono per un fattore polinomiale $\Theta(n^{\varepsilon})$	Il costo dei livelli decresce almeno geometricamente e il costo della radice domina quello dei nodi.

• Esempio:

$$T(n) = 4T\left(\frac{n}{2}\right) + n^{1.99}$$

$a = 4$, $b = 2$

$f(n) = n^{1.99}$, $w(n) = n^{\log_2 4} = n^2$

$f(n) = n^{1.99} = n^{2-0.01}$, $f(n) = O(n^{2-\varepsilon})$, per $0 < \varepsilon \leq 0.01$

$\rightarrow T(n) = \Theta(n^2)$.

3.2 Esempi ed esercizi:

$$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$$

⚠ Ci sono situazioni in cui non è possibile applicare il Th. Master:

- C'è un gap tra il caso 1 e il caso 2
- C'è un gap tra il caso 2 e il caso 3
- $f(n)$ e $w(n)$ potrebbero essere non asintoticamente comparabili
- $w(n)$ è dominato da $f(n)$ ma la condizione di regolarità non è soddisfatta.

ESEMPIO:

$$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$$

$a = b = 2$

$f(n) = \frac{n}{\log n}$, $w(n) = n^{\log_2 2} = n$

- $f(n)$ cresce più lentamente di $w(n)$ → NON SIAMO NEL CASO 3

- $\forall \varepsilon > 0, \ \log n = o(n^{\varepsilon}) \Rightarrow \frac{1}{\log n} = \omega(n^{-\varepsilon})$

$\Rightarrow \frac{n}{\log n} = \omega(n^{1-\varepsilon}) \Rightarrow \frac{n}{\log n} \neq O(n^{1-\varepsilon})$ → NON SIAMO NEL CASO 1

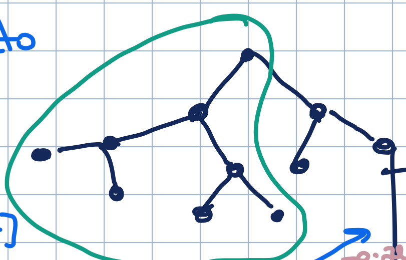
- $\frac{n}{\log n} = \Theta(n \log^k n)$ accade solo $k = -1$ ma $k \geq 0$ → NON SIAMO NEL CASO 2

3.3. ESEMPIO (IMPORTANTE) : HEAPIFY

Il tempo necessario a correggere la relazione tra $A[i]$, $A[i.left]$, $A[i.right]$ è $\Theta(1)$.

L'albero di ricorrenze potrebbe non essere completo → dim sottoalbero radicato ai figli può essere $\frac{n}{2}$ e $\frac{n}{2}$ o $\frac{n}{3}$ e $\frac{2}{3}n$ → caso peggiore

Per un albero di dimensione n radicato a un nodo i il tempo che impiega HEAPIFY è quello per correggere le relazione tra $A[i]$, $A[i.left]$ $A[i.right]$ + il tempo di eseguire HEAPIFY su un sottoalbero radicato a uno dei figli



caso peggiore

$$T(n) \leq T\left(\frac{2}{3}n\right) + \Theta(1)$$

n è la dimensione dell'istanza considerata = altezza del nodo delle strutture albero binario a cui i sottoalberi sono radicati è $\log n$.

$a = 1, \ b = \frac{3}{2}, \ f(n) = \Theta(1), \quad w(n) = n^{\log_{\frac{3}{2}} 1} = n^0 = 1$

$f(n) = \Theta(1 \log^0 n) \xrightarrow[\text{caso 2}]{} T(n) = O(\log n)$

$$\boxed{\text{→ Eseguire HEAPIFY su un sottoalbero con altezza } h \text{ impiega } O(h) : \leq ch}$$

OSS

ANALISI DI BUILD-MAX-HEAP

BUILD-MAX-HEAP
1. A.heap-size = n
2. for i $\lfloor \frac{n}{2} \rfloor$ down to 1
3. MAX-HEAPIFY

Tempo comp: $O(n \log n)$ (bound grossolano)

Heap con n nodi

$\frac{n}{2^{h+1}}$: # di nodi di altezza h

e altezza $\lfloor \log n \rfloor$

In realtà, usando oss

$$T(n) \leq \sum_{h=0}^{\lfloor \log n \rfloor} ch \left\lceil \frac{n}{2^{h+1}} \right\rceil \leq$$

$\lceil x \rceil \leq 2x$, $|x| \geq 1/2$ $\Rightarrow$ $\left\lceil \frac{n}{2^{h+1}} \right\rceil \leq \frac{n}{2^h}$

poiché $\frac{n}{2^{h+1}} \geq 1/2$ per $0 \leq h \leq \lfloor \log n \rfloor$

$$\leq \sum_{h=0}^{\lfloor \log n \rfloor} ch \frac{n}{2^h} = cn \sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h} \leq cn \sum_{h=0}^{\infty} \frac{h}{2^h} = cn \sum_{h=0}^{\infty} h \left(\frac{1}{2}\right)^h = cn \frac{1/2}{(1-1/2)^2} = O(n).$$

$$\sum_{h=0}^{\infty} h x^h = \frac{x}{(1-x)^2} \quad \text{se } |x| < 1$$