Lezione 2

2.1 EQUAZIONI DI RICORRENZA

Approcci "divide and conquer" $\begin{cases} \text{divide} \\ \text{conquer} \\ \text{combine} \end{cases}$

Recursion
Bottom : i sotto pr. sono istanze del caso base che possono essere risolte direttamente, in tempo costante.

Le eq di ricorrenza sono lo strumento naturale per analizzare la complessità degli algoritmi ricorsivi.

Un'equazione di ricorrenza è una formula che descrive il valore di una funzione in corrispondenza di un argomento x generico in termini del valore della funzione stessa assume su altri argomenti, in genere di dimensione inferiore.

Supponiamo che l'algoritmo ricorsivo considerato divida un problema di dim n in a sottoproblemi di dimensione $\frac{n}{b}$.

$$T(n) = \begin{cases} D(n) + a\,T\left(\frac{n}{b}\right) + C(n) & \forall n \geq n \\ \Theta(1) & \forall n < n_0 \end{cases}$$

divide → $D(n)$
combine → $C(n)$

In alcuni casi la formula di ricorrenza è una disuguaglianza

$\leq$ → upper bound → big-O

$\geq$ → lower bound → big-Ω

Analizzare la complessità di un algoritmo ricorsivo significa risolvere l'eq. di ricorrenza ad esso associata.

Metodi per risolvere un'eq. di ricorrenza:

(ne esistono almeno 4)

- METODO DI SOSTITUZIONE
- METODO DELL'ALBERO DI RICORSIONE
- METODO MASTER
- METODO AKRA-BAZZI (non lo vedremo).

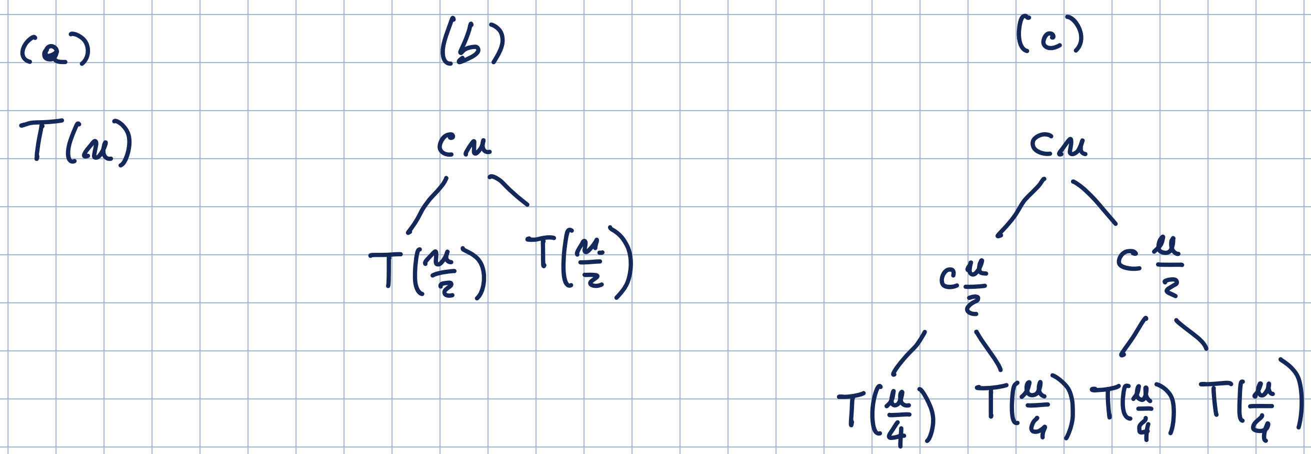
2.2 ESEMPIO: MERGE SORT

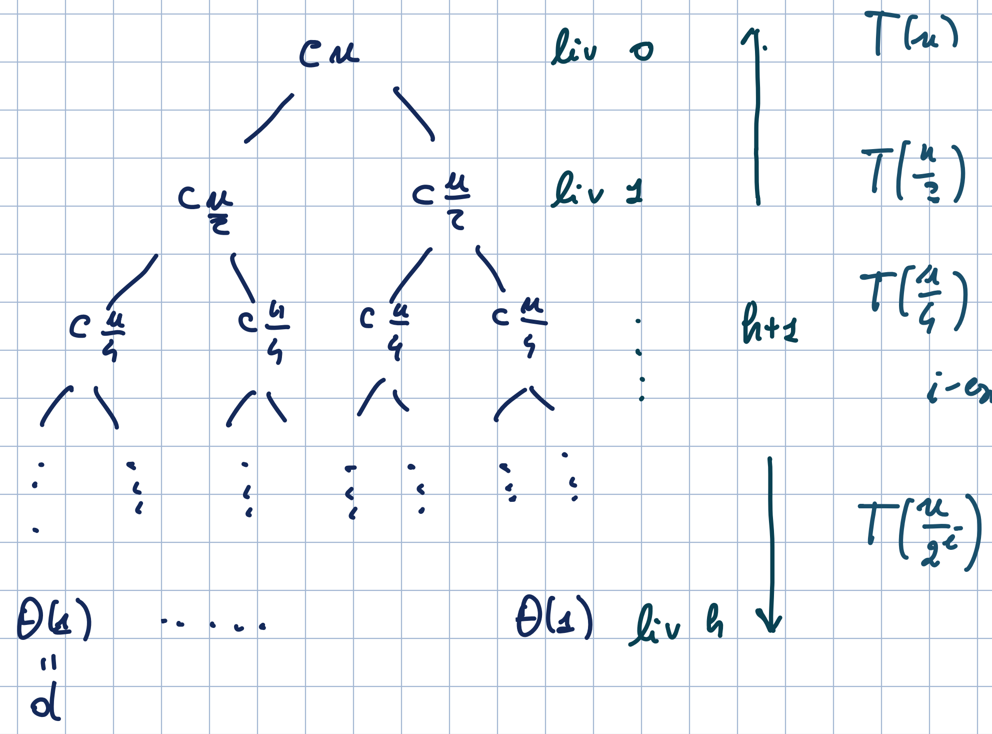
- Divide: calcola la metà dell'array $(\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)$: $\rightarrow D(n) = \Theta(1)$.
- Conquer: risolve $2 = a$ problemi di dimensione $\frac{n}{2}$ $(b = 2)$.
- Combine: Merge, $C(n) = \Theta(n)$

$$T(n) = \Theta(1) + 2T\left(\frac{n}{2}\right) + \Theta(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \qquad \forall\, n \geq n_0$$

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + \Theta(n) & \forall\, n \geq 2 \\ \Theta(1) & n = 1 \end{cases}$$

$$T(n) \leq 2T\left(\frac{n}{2}\right) + \underbrace{\Theta(n)}_{cn}$$





$$h: \; T\left(\frac{n}{2^h}\right) = T(1) \;\rightarrow\; \frac{n}{2^h} = 1 \;\rightarrow\; n = 2^h \;\rightarrow\; h = \log n$$

i-esimo livello:

- costo nodo: $c\frac{n}{2^i}$
- # nodi: 2^i

$\Big\}$ costo i-esimo liv $= 2^i c \frac{n}{2^i} = cn$

$$\#\text{ foglie} = 2^h = 2^{\log n} = n.$$

$$T(n) = \Big(\sum_{i=0}^{h-1} \underbrace{cn}_{\text{costo i-esimo livello}}\Big) + d \cdot n = h \cdot cn + dn = cn \log n + dn = O(n \log n)$$

$$\Theta(n \log n)$$

2.3. METODO DELL'ALBERO DI RICORSIONE

Ogni sottoalbero corrisponde a un sottoproblema.

Si sommano i costi dei nodi a ogni livello, e poi i costi di tutti i livelli e quelli delle foglie.

EX: $T(n) = 3T\left(\frac{n}{4}\right) + \Theta(n^2)$

HINT: $\sum_{n \in \mathbb{N}} q^i = \frac{1}{1-q}$, $-1<q<1$

$\sum_{i=0}^{k} q^i = \frac{1-q^{k+1}}{1-q}$, $q \neq 1$

2.4 METODO DI SOSTITUZIONE (metodo più generale)

- GUESS (supporre la forma della soluzione usando costanti simboliche)
- INDUZIONE MATEMATICA per verificare la soluzione proposta e trovare le costanti.

Bisogna sempre esplicitare le costanti.

ESEMPIO (a): $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \underbrace{\Theta(n)}_{\to\, dn}$

$\exists c>0,\ \exists n_0 : \forall n \geq n_0$

guess: $T(n) = O(n \log n) \to T(n) \leq cn\log n$

caso base: $n < n_0$ $\quad T(n) = \Theta(1)$, ci torniamo dopo

Ipotesi induttiva: assumiamo che $T\left(\left\lfloor \frac{n}{2}\right\rfloor\right) \leq c\left\lfloor \frac{n}{2}\right\rfloor \log\left(\left\lfloor \frac{n}{2}\right\rfloor\right)$

$n \geq n_0$, $\frac{n}{2} \geq n_0$

SOSTITUIAMO:

$$T(n) = 2T\left(\left\lfloor \frac{n}{2}\right\rfloor\right) + dn \leq$$

$$2\left(c\left\lfloor \frac{n}{2}\right\rfloor \log\left\lfloor \frac{n}{2}\right\rfloor\right) + dn \leq$$

$$\cancel{2}c\frac{n}{\cancel{2}}\log\frac{n}{2} + dn = cn(\log n - \log 2) + dn =$$

$$= cn\log n - cn + dn \overset{?}{\leq} cn\log n$$

per $c \geq d$

Caso base: $n_0 = 1$ $\quad c\, n_0 \log n_0$ NO $\rightarrow$ caso base

$n_0 = 2$

(base dell'induzione) : $T(n) \leq cn\log n$ $\quad \left.\begin{array}{l} n \geq n_0 \\ \frac{n}{2} < n_0 \end{array}\right\} \rightarrow \boxed{n_0 \leq n < 2n_0}$

$n_0 = 2 \quad \rightarrow \quad 2 \leq n < 4 \quad \rightarrow \quad n = 2, n = 3$

$c = \max\{T(2), T(3)\}$ $\quad$ $c = \max\{T(2), T(3)\}$

$T(2) \leq c \leq c \cdot 2\log 2$ ✓ $\quad$ $0 < d \leq c$

$T(3) \leq c \leq c \cdot 3 \log 3$ ✓

EX (b) $\quad T(n) = 2T\left(\frac{n}{2} + 17\right) + \Theta(n)$

guess: $O(n\log n)$ $\quad n \geq n_0 = 17$

TRICK OF THE TRADE

Es (c): $\quad T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$

guess: $T(n) = O(n) \rightarrow T(n) \leq cn - d$

sost: $T(n) \leq 2T\left(\frac{n}{2}\right) + d \leq 2c\frac{n}{2} + d = cn + \underset{0}{\underset{\vee}{d}} \overset{?}{\leq} cn$

$T(n) \leq 2T\left(\frac{n}{2}\right) + d \leq 2\left(c\frac{n}{2} - d\right) + d = cn - 2d + d =$

$= cn - d \leq cn$

ES (d): $\quad T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n)$

guess: $T(n) = O(n) \rightarrow T(n) \leq cn$

sbagliato

sostit: $T(n) \leq 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) \leq 2\,O\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) = O(n) + \Theta(n) = O(n)$

sost: $T(n) \leq 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) \leq 2c\left\lfloor \frac{n}{2} \right\rfloor + \Theta(n) \leq cn + \Theta(n) = O(n)$

$\checkmark$

$0 \qquad \overset{?}{\leq} cn$