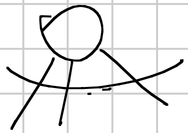
$L_{\Pi}(n) = \Omega(n \log n)$ Π: ordinamento per confronti

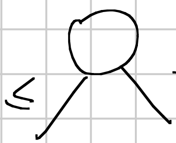
Π problema che si risolve per decisioni/confronti

$S(n) = \#$ soluzioni di Π su istanze di input di dim. n

ogni confronto ottiene t esiti

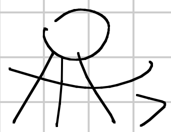


t



$\le$ $>$

$t = 2$ ordinamento



$<$ $=$ $>$

$t = 3$ ricerca

$\forall \Pi$, AdD ~~per~~ che rappresenta un algoritmo per Π

Titolo nota 09/03/2016

$$\underbrace{S(n) \leq f}_{\text{correttezza dell'algoritmo}} \underbrace{\leq t^h}_{\text{proprietà degli alberi t-ari completi}}$$

h = altezza = # decisioni

$$h \geq \log_t (S(n))$$

$$\Rightarrow \quad L_\Pi(n) = \Omega(\log S(n))$$

Titolo nota 09/03/2016

Esempi ricerca della posizione dell'elemento MAX in un array

$S(n) = n$ (array di dim n)

$L_\Pi(n) = \Omega(\log n)$ NON È SIGNIFICATIVO.

$L(n) = \Omega(n)$

(tecnica della dimensione dell'input)

Π: ricerca di una chiave k in un array a di n interi

$$S(n) = \underbrace{n}_{\substack{\text{pos. di } k \\ \text{in } a \\ (k \in a)}} + \underbrace{1}_{k \notin a}$$

$$L_\Pi(n) = \Omega\left(\log_3 (n+1)\right) = \Omega(\log n)$$

SIGNIFICATIVO per il PROBLEMA della RICERCA IN UN ARRAY ORDINATO

$\Rightarrow$ Ricerca Binaria è un algoritmo ottimo

(3) Tecnica degli EVENTI CONTABILI

\# volte in cui un evento si deve ripetere per risolvere un problema Π

→ limite inferiore

ESEMPIO

Π: generare le permutazioni di n elementi distinti

EVENTO: generazione di una permutazione

$L_{\Pi}(n) = \Omega(n!)$

Π: problema della ricerca del MAX / MIN in un array

MAX

Evento: ogni elemento NON massimo deve uscire perdente da un confronto

$L_\Pi(n) = n-1$ confronti ($n-1$ elementi NON max)

↳ sono necessari (e anche sufficienti)

↳ abbiamo un algoritmo che esegue esattamente $n-1$ confronti

Π: Problema del Torneo

n squadre
selezionare la più forte
ipotesi: "transitività della bravura"
A batte B e B batte C ⇒ A batte C
(non c'è bisogno di giocare)

LIM. INFERIORE

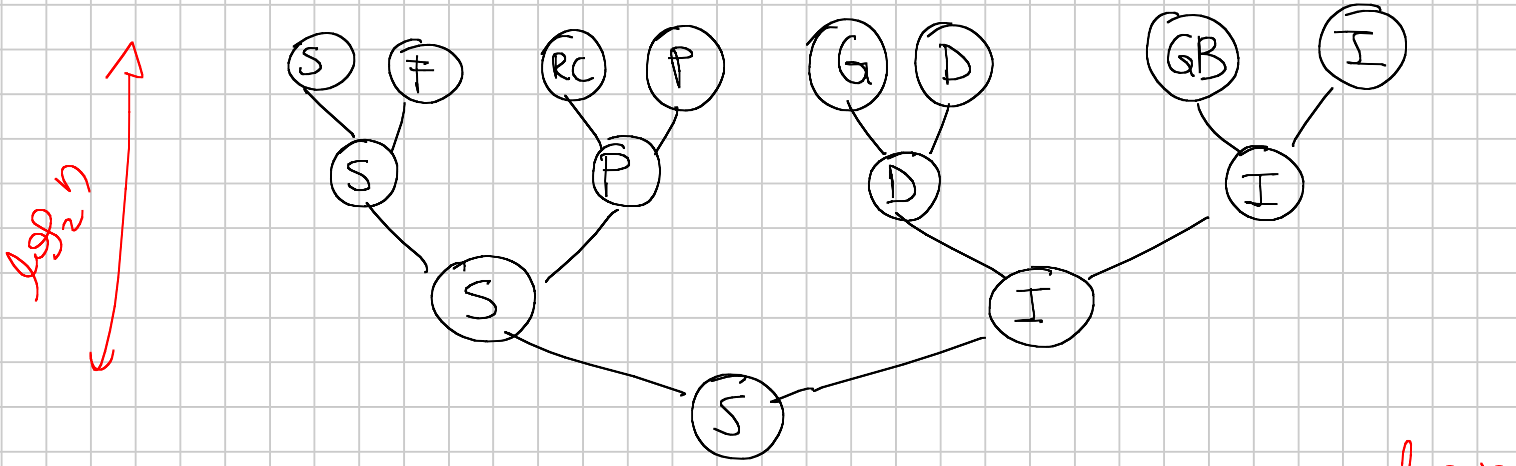
evento: le n-1 squadre che non sono la più forte
devono perdere almeno una partita

⇒ n-1 partite sono necessarie

Algoritmo del torneo a eliminazione diretta (è ottimo)

n = potenza di 2

EUROPEI 2012



foglie: squadre

nodi interni: partite

radice: finale $\downarrow$ squadra più forte

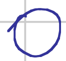
La squadra in finale hanno giocato $\log_2 n$ partite

Proprietà

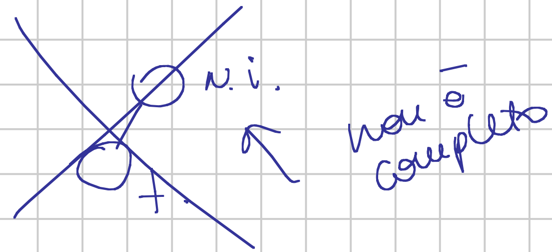
In un albero binario completo ~~di n nodi foglie~~ con n foglie ci sono esattamente $n-1$ nodi interni.

Dim (per induzione su n)

BASE $n = 1$



nodi interni $= 0 = 1 - 1$

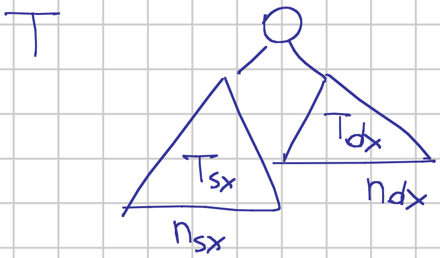


ipotesi induttiva

ogni ABC con $f < n$ foglie contiene $f - 1$ nodi interni.

passo ABC con n foglie

T



$n_{sx} = \#$ foglie del sottoalbero T_{sx} $< n$

$n_{dx} = \#$ " " " T_{dx} $< n$

$n = n_{sx} + n_{dx}$

$$\text{nodi interni di } T = \underbrace{1}_{\text{radice}} + (\text{nodi interni di } T_{sx}) + (\text{nodi interni } T_{dx})$$

$$\underset{\substack{\text{i.i.} \\ (n_{sx}<n,\ n_{dx}<n)}}{=} \cancel{1} + (n_{sx} - 1) + (n_{dx} \cancel{-1}) = n_{sx} + n_{dx} - 1 = n - 1 \qquad \square$$

Torneo a elim. diretta $\rightarrow$ algoritmo ottimo!

Problema del doppio torneo

selezionare le 2 squadre più forti.

la ~~squadra~~ seconda squadra veramente più forte è da ricercare tra tutte e sole le squadre che hanno perso una partita con la squadra più forte.

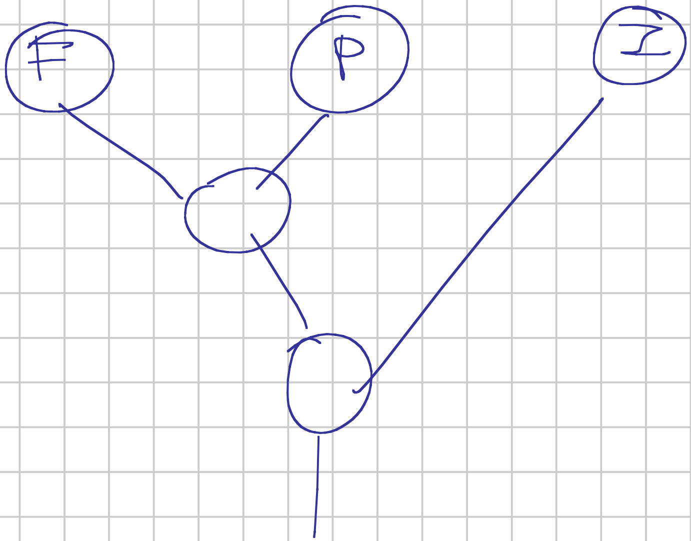
↓

sono $\log_2 n$ (la squadra più forte gioca $\log n$ partite)

↓

~~si~~ giocano un secondo torneo a eliminazione diretta tra loro

↳ chi vince è la seconda squadra veramente più forte



occorrono oltre

$\log_2 n - 1$ partite

$\#$ partite $= (n-1) + (\log n - 1)$ } sufficienti per risolvere il problema del doppio torneo

si può dimostrare che sono anche necessarie.

Ricorrenze

- Metodo di sostituzione
- Metodo iterativo
  (metodo dell'albero di ricorsione)
- Metodo dell'esperto

$$T(n) = \begin{cases} \Theta(1) & n \leq 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases} \qquad a \geq 1, \quad b > 1$$

Costo in tempo di un alg. D. & I., che divide un problema di dim n in $\underline{a}$ sottoproblemi, tutti di dimensione n/b. I sottoproblemi sono risolti ricorsivamente con la stessa tecnica, in tempo $T\left(\frac{n}{b}\right)$ ciascuno.
$f(n) \rightarrow$ costo al di fuori delle chiamate ricorsive, x dividere e ricombinare

TEOREMA dell'ESPERTO

Date le costanti $a \geq 1$ e $b > 1$, e la funzione $f(n)$ non negativa, sia $T(n)$ una funzione definita sugli interi non negativi dalla ricorrenza

$$T(n) = aT(n/b) + f(n)$$

dove $\frac{n}{b}$ rappresenta $\lfloor n/b \rfloor$ o $\lceil n/b \rceil$. Allora $T(n)$ può essere asintoticamente limitata nei seguenti modi

1) se $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ per qualche costante $\varepsilon > 0$, allora $T(n) = \Theta\left(n^{\log_b a}\right)$

2) se $f(n) = \Theta\left(n^{\log_b a}\right)$, allora $T(n) = \Theta\left(n^{\log_b a} \cdot \log n\right)$

3) se $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ per qualche costante $\varepsilon > 0$, e se vale la condizione di regolarità

$$a f\left(\frac{n}{b}\right) \leq c \cdot f(n)$$ per qualche costante $c < 1$, e $\forall n$ suff. grande,

allora $T(n) = \Theta(f(n))$

ESEMPI

Merge Sort

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

$a = 2$, $b = 2$, $f(n) = \Theta(n)$

$n^{\log_b a} = n^{\log_2 2} = n$

$f(n) = \Theta\left(n^{\log_b a}\right) \Rightarrow$ II° caso del teorema $T(n) = \Theta(n \cdot \log n)$

Ricerca Binaria

$$T(n) = T\left(n/2\right) + \Theta(1)$$

$a = 1$, $b = 2$, $f(n) = \Theta(1)$

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$.

$n^{\log_b 1} = n^0$

$\Rightarrow$ II° caso $T(n) = \Theta(1 \cdot \log n)$

$$T(n) = 9\,T\left(\frac{n}{3}\right) + n$$

$a = 9 \qquad b = 3 \qquad f(n) = n$

$n^{\log_b a} = n^{\log_3 9} = n^2$

$f(n) = n^1 = O\left(n^{\log_3 9 - \varepsilon}\right)$ vero se ad es. $\varepsilon = 1$

$1 \le \log_3 9 - \varepsilon \Rightarrow 0 < \varepsilon \le \log_3 9 - 1 = 2 - 1 = 1$

I° caso: $T(n) = \Theta\left(n^{\log_3 9}\right) = \Theta(n^2)$

$$T(n) = 3T(n/4) + n \log n$$

$a = 3$, $b = 4$, $f(n) = \cancel{\Theta}\ n \cdot \log n$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.795\ldots} \qquad \log_4 3 < 1$$

$$f(n) = n^{1} \log n = \Omega\left(n^{\log_4 3 + \varepsilon}\right) \quad \checkmark \text{ ok per } \varepsilon = 0.2$$

$1 \geq \log_4 3 + \varepsilon \qquad \varepsilon \leq 1 - \log_4 3 = 1 - 0.795$

Cond. di regolarità:

$$a f\left(\frac{n}{b}\right) = 3 f\left(\frac{n}{4}\right) = 3 \cdot \frac{n}{4} \cdot \log\left(\frac{n}{4}\right) \leq \underbrace{\frac{3}{4}}_{<1} \cdot \underbrace{n \cdot \log n}_{f(n)} = \frac{3}{4} f(n)$$

$c = \frac{3}{4} < 1$

III° caso: $T(n) = \Theta(f(n)) = \Theta(n \log n)$

Titolo nota 09/03/2016

ESEMPIO

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

$a = 2, \quad b = 2, \quad f(n) = n \log n$

$$n^{\log_b a} = n^{\log_2 2} = n$$

$f(n) > n^{\log_b a}$ ma solo per un fattore $\log n$.

$$f(n) \notin \Omega\left(n^{\log_b a + \varepsilon}\right) \quad \forall \varepsilon > 0$$

$n^1 \log n \qquad n^{1+\varepsilon} \qquad 1 \geq 1+\varepsilon$

$\varepsilon > 0$

NO $\overline{\underline{III}}^\circ$ caso

DIMOSTRAZIONE

$n = b^j \qquad j \in \mathbb{N}$ (solo per le potenze esatte di b)

$$T(n) = a\,\underbrace{T\left(\frac{n}{b}\right)} + f(n) = a\left[a\,T\left(\frac{1}{b}\cdot\frac{n}{b}\right) + f\left(\frac{n}{b}\right)\right] + f(n) =$$

$$= a^2\,\underbrace{T\left(\frac{n}{b^2}\right)} + a f\left(\frac{n}{b}\right) + f(n) = a^2\left[a\,T\left(\frac{1}{b}\cdot\frac{n}{b^2}\right) + f\left(\frac{n}{b^2}\right)\right] + a f\left(\frac{n}{b}\right) + f(n) =$$

$$= a^3\,T\left(\frac{n}{b^3}\right) + a^2 f\left(\frac{n}{b^2}\right) + a f\left(\frac{n}{b}\right) + f(n) = a^3\,T\left(\frac{n}{b^3}\right) + \sum_{j=0}^{2} a^j f\left(\frac{n}{b^j}\right)$$

$$= \cdots = a^i\,T\left(\frac{n}{b^i}\right) + \sum_{j=0}^{i-1} a^j f\left(\frac{n}{b^j}\right) \underset{i=\log_b n}{=} a^{\log_b n}\cdot T\left(\frac{n}{b^{\log_b n}}\right) + \sum_{j=0}^{\log_b n-1} a^j f\left(\frac{n}{b^j}\right)$$

$b^{\log_b n} \to n$

$$= n^{\log_b a}\cdot \underbrace{T(1)}_{\Theta(1)} + \underbrace{\sum_{j=0}^{\log_b n-1} a^j f\left(\frac{n}{b^j}\right)}_{g(n)} = \underline{\Theta\left(n^{\log_b a}\right)} + g(n) \to \text{lavoro dei passi D4 I}$$

$n^{\log_b a}$: # foglie di dim 1

$\Theta(1)$: costo della soluzione diretta

$\Theta\left(n^{\log_b a}\right)$: soluzione diretta di tutti i sottoproblemi di dim 1