OTAZIONE ASINTOTICA

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = k$

$n^{\log_b a}$ per il TH. MASTER

$\Omega(g(n)) = k > 0$

$\Theta(g(n)) = 0 < k < \infty$

$O(g(n)) = k < \infty$ $\quad O \cap \Omega = \Theta$

$o(g(n)) = k = 0$ $\quad o \subseteq O$

$\omega(g(n)) = k = \infty$ $\quad \omega \subseteq \Omega$

$o \cap \omega = \emptyset$

EOREMA MASTER

$T(n) = a\,T(n/b) + f(n)$

$a \geq 1 \quad b > 1 \quad n > 1$

$\Theta(1)$ se $n = 1$

1) $T(n) = \Theta(n^{\log_b a})$ se $f(n) = O(n^{\log_b a - \varepsilon})$ per $\varepsilon > 0$.

2) $T(n) = \Theta(n^{\log_b a} \log n)$ se $f(n) = \Theta(n^{\log_b a})$

3) $T(n) = \Theta(f(n))$ se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per $\varepsilon > 0$ e ...

- Condiz. di regolarità: $a\,f(n/b) \leq c\,f(n)$ per $c < 1$ e n. suff. grande

EFINIZIONI:

albero d-ario: tutti i nodi interni hanno al più d figli

albero strettamente binario: tutti i nodi interni hanno ESATTAMENTE 2 figli, il suo numero di nodi interni è = al numero di foglie − 1

albero d-ario completo: tutti nodi interni hanno esattamente d figli e TUTTE le foglie sono sullo STESSO LIVELLO

albero quasi completo: completo tranne all'ultimo livello $\Rightarrow$ foglie stanno sull'ultimo o sul penultimo livello, un albero quasi completo d-ario di N elementi ha ALTEZZA $\Theta(\log_d n)$

- Un albero binario completo di N nodi ha esattamente $\lceil N/2 \rceil$ foglie
- Algoritmo OTTIMO: se dato un problema P con $\Omega(g(n))$ (parte sup.) l'algoritmo ha costo $O(g(n)) \Rightarrow$ complessità asintotica la migliore possibile

ARRAY/LISTA NON ORDINATO:

FINDMIN	INSERT	DELETE	DELETEMIN
$\Theta(n)$	$O(1)$	$O(1)$	$\Theta(n)$

ARRAY/LISTA ORDINATO:

FINDMIN	INSERT	DELETE	DELETEMIN
$O(1)$	$O(n)$	$O(n)_A$ $O(1)_L$	$O(1)$

ALGORITMI e STRUTTURE DATI

FIBONACCI 1

FORMULA DI BINET:

RETURN $\frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$

INCORRETTO!

FIBONACCI 2

$T(n) = 2 + T(n-1) + T(n-2) \quad n \geq 3$

$T(1) = T(2) = 1 \quad n = 1, 2$

$\Theta(n) \Rightarrow$ input rappresentato a bit $\Rightarrow \Theta(\phi^{2^k})$

FIBONACCI 3

Usa un array [n] per memorizzare i risultati

FIBONACCI 4

usa variabili di alloggio per memorizzare gli ultimi 2 ris per il

FIBONACCI 5

potenze di matrici fino ad $n-1$, in alto a sx il ris.

FIBONACCI 6

divide et impera sulle matrici

$\Theta(\log n)$

RICERCA SEQUENZIALE ottimo per caso

$T_{best}(n) = \Theta(1)$

$T_{worst}(n) = \Theta(n)$

$T_{avg}(n) = T_{worst}(n)$

$T(n) = O(n)$

RICERCA BINARIA

$T_{best}(n) = \Theta(1)$

$T_{worst}(n) = \Theta(\log n)$

$T_{avg}(n) = T_{worst}(n)$

$T(n) = O(\log n)$

SELECTIONSORT

$T(n)$

$T_{worst}(n) = T_{best}(n) = T_{avg}(n) = \Theta(n^2)$

INSERTIONSORT 1

$T(n)$

$T_{worst}(n) = T_{best}(n) = T_{avg}(n) = \Theta(n^2)$

INSERTIONSORT 2

$T_{best}(n) = \Theta(n)$

$T_{worst}(n) = \Theta(n^2)$

$T_{avg}(n) = \Theta(n^2)$

$T(n) = O(n^2)$

INSERTIONSORT 3

$T_{best}(n) = \Theta(n \log n)$

$T_{worst}(n) = \Theta(n^2)$

$T_{avg}(n) = \Theta(n^2)$

$T(n) = O(n^2)$

MERGESORT e HEAPSORT ottimo

$T(n) = \Theta(n \log n)$

QUICKSORT ottimo nella pratica

$T_{best}(n) = \Theta(n \log n)$

$T_{worst}(n) = \Theta(n^2)$

$T_{avg}(n) = \Theta(n \log n)$

INTEGERSORT ottimo per val. $O(n)$ max int.

$T(n) = \Theta(n)$ se $K = O(n)$

BUCKETSORT

$T(n) = \Theta(n)$ se chiavi int. in $[0, K = O(n)]$

RADIX SORT ottimo per val. tra 1 e n^c, $c > 1$

$T(n) = \Theta(n)$ se n int con val. $[0, K = O(n^c)]$