

1. *Dare una definizione informale e formale di algoritmo*

Informalmente, un algoritmo è una procedura computazionale ben definita eseguita per risolvere un dato problema computazionale. Formalmente, un algoritmo è un procedimento di calcolo esplicito, descrivibile con un numero finito di regole che conduce al risultato dopo un numero finito di operazioni, cioè di applicazioni di regole

2. *Dare una definizione di programma*

I programmi sono formulazioni concrete di algoritmi astratti che si basano su particolari rappresentazioni dei dati, e utilizzano operazioni di manipolazione dei dati, messe a disposizione da uno specifico linguaggio di programmazione

3. *Quale è la differenza tra il concetto di algoritmo e quello di programma?*

Un algoritmo è una formulazione astratta, un programma è una formulazione concreta di un algoritmo necessaria per poter eseguire l'algoritmo su un calcolatore

4. *Quali sono i vantaggi derivanti dall'analisi di algoritmi in termini di tempo di esecuzione?*

L'analisi della complessità di un algoritmo in termini di tempo di esecuzione consente di: (1) stimare il tempo impiegato, (2) stimare il più grande input gestibile in termini ragionevoli, (3) confrontare l'efficienza di due algoritmi diversi, (4) ottimizzare le parti critiche

5. *Perché è utile l'analisi asintotica?*

Ci consente di astrarre da alcuni dettagli irrilevanti (ex: termini di ordine diverso, costanti moltiplicative) e di focalizzare la nostra attenzione solo sul tasso di crescita – analizziamo come il tempo di esecuzione cresce asintoticamente in funzione della dimensione dell'input. L'obiettivo è quello di semplificare l'analisi del tempo di esecuzione di un algoritmo prescindendo dai dettagli implementativi o di altro genere e di classificare le funzioni in base al loro comportamento asintotico

6. *Qual'è la complessità dell'**InsertionSort** nel caso peggiore, medio e ottimo?*

Caso peggiore e medio: $O(n^2)$, caso ottimo $O(n)$

7. *Descrivere informalmente il significato delle notazioni O , Ω e Θ*

Forniscono delle delimitazioni superiori, inferiori e strette alla complessità degli algoritmi

8. *Dare la definizione formale della notazione O*

Siano $f(n)$ e $g(n)$ delle funzioni non negative. Diciamo che $f(n) = O(g(n))$ se esistono delle costanti positive c, n_0 tali che $f(n) \leq cg(n)$ per ogni $n \geq n_0$

9. *Vero o falso?*

- $n^2 = \Theta(n \log_2 n)$
- $n^k = O(a^n)$ per ogni $k > 0$ e $a > 1$
- $3^n = \Theta(2^n)$

10. *Assumendo $T(1) = 1$, risolvere le seguenti ricorrenze utilizzando il metodo indicato*

- (a) $T(n) = 2T(n/2) + n^2$ (metodo iterativo)
- (b) $T(n) = 2T(n/3) + n^2$ (alberi di ricorsione)
- (c) $T(n) = 4T(n/4) + \sqrt{n}$ (teorema del master – ricordo che $\sqrt{n} = n^{1/2}$)

(a)

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + n^2 \\
&= 2[2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2] + n^2 = 4T\left(\frac{n}{4}\right) + \frac{n^2}{2} + n^2 && \text{dopo 2 passi} \\
&= 4[2T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)^2] + \frac{n^2}{2} + n^2 = 8T\left(\frac{n}{8}\right) + \frac{n^2}{4} + \frac{n^2}{2} + n^2 && \text{dopo 3 passi} \\
&= 8[2T\left(\frac{n}{16}\right) + \left(\frac{n}{8}\right)^2] + \frac{n^2}{4} + \frac{n^2}{2} + n^2 = 16T\left(\frac{n}{16}\right) + \frac{n^2}{8} + \frac{n^2}{4} + \frac{n^2}{2} + n^2 && \text{dopo 4 passi} \\
&= \dots \\
&= 2^k T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k-1} \frac{n^2}{2^i} = 2^k T\left(\frac{n}{2^k}\right) + n^2 \sum_{i=0}^{k-1} \left(\frac{1}{2}\right)^i && \text{dopo } k \text{ passi}
\end{aligned}$$

Ci fermiamo quando $\frac{n}{2^k} = 1$ e quindi quando $k = \log n$. Da cui

$$\begin{aligned}
T(n) &= 2^{\log n} + n^2 \sum_{i=0}^{\log n - 1} \left(\frac{1}{2}\right)^i = n + n^2 \frac{1 - \left(\frac{1}{2}\right)^{\log n}}{1 - \frac{1}{2}} = n + n^2 \frac{1 - \left(\frac{1}{2^{\log n}}\right)}{\frac{1}{2}} = \\
&= n + 2n^2 \left(1 - \frac{1}{n}\right) = 2n^2 - n = \Theta(n^2)
\end{aligned}$$

(b) Albero di ricorsione per $T(n)$ è dato in Figura 1. Osserviamo che:

- ogni livello i ha 2^i nodi; inoltre il costo di ogni nodo al livello i è il costo associato alla **radice** dell'albero di ricorsione per $T\left(\frac{n}{3^i}\right)$ ossia $\left(\frac{n}{3^i}\right)^2$. Il costo di ciascun livello i è pari a:

$$c_i = 2^i \cdot \left(\frac{n}{3^i}\right)^2 = 2^i \frac{n^2}{(3^i)^2} = 2^i \frac{n^2}{(3^2)^i} = 2^i \frac{n^2}{9^i} = \left(\frac{2}{9}\right)^i n^2$$

- l'ultimo livello corrisponde ad una chiamata di $T(n/3^h)$ con $\frac{n}{3^h} = 1$ e quindi $h = \log_3 n$ (qui h è pari all'altezza h dell'albero).

Il valore della ricorrenza viene calcolato sommando prima i costi dei nodi su ciascun livello; quindi:

$$T(n) = \sum_{i=0}^{\log_3 n} c_i = \sum_{i=0}^{\log_3 n} \left(\frac{2}{9}\right)^i n^2 = n^2 \sum_{i=0}^{\log_3 n} \left(\frac{2}{9}\right)^i = n^2 \frac{1 - \left(\frac{2}{9}\right)^{\log_3 n + 1}}{1 - \frac{2}{9}} = \frac{9}{7} n^2 \left(1 - \left(\frac{2}{9}\right)^{\log_3 n + 1}\right)$$

Ora

$$\left(\frac{2}{9}\right)^{\log_3 n + 1} = \frac{2}{9} \left(\frac{2}{9}\right)^{\log_3 n} = \frac{2}{9} \left(\frac{2^{\log_3 n}}{9^{\log_3 n}}\right) = \frac{2}{9} \left(\frac{n^{\log_3 2}}{n^{\log_3 9}}\right) = \frac{2}{9} \left(\frac{n^{\log_3 2}}{n^2}\right) = \frac{2}{9} \left(\frac{n^\alpha}{n^2}\right)$$

(abbiamo posto $\alpha = \log_3 2$). Allora:

$$T(n) = \frac{9}{7} n^2 \left(1 - \left(\frac{2}{9}\right)^{\log_3 n + 1}\right) = \frac{9}{7} n^2 \left(1 - \frac{2}{9} \left(\frac{n^\alpha}{n^2}\right)\right) = \frac{9}{7} n^2 + \frac{2}{7} n^\alpha$$

Inoltre, $1 < 2 < 3$ implica (pochè il logaritmo è crescente) $0 = \log_3 1 < \alpha = \log_3 2 < \log_3 3 = 1$. Questo significa che

$$T(n) = \frac{9}{7} n^2 + \frac{2}{7} n^\alpha = \Theta(n^2)$$

- (c) Poichè $a = b = 4$, $\log_b a = 1$ e $n^{\log_b a} = n$. Allora, $T(n) = n^{\frac{1}{2}} = \Omega(n^{\frac{1}{2}}) = \Omega(n^{\log_b a - \varepsilon})$ con $\varepsilon = \frac{1}{2} > 0$.
Caso 1 del teorema del master. Possiamo concludere che $T(n) = \Theta(n^{\log_b a}) = \Theta(n)$

11. Si dimostri applicando il metodo di sostituzione che la seguente ricorrenza è $O(n)$:

$$T(n) = 2T(n/2) + 1 \text{ se } n > 1, T(1) = 1 \text{ se } n = 1$$

Dimostriamo che esistono delle costanti positive c e d tali che $T(n) \leq cn - d$ per ogni $n \geq 1$. Procediamo per induzione su n .

Caso base $n = 1$: $T(1) = 1 \leq c \cdot 1 - d = c - d$ basta scegliere $c - d \geq 1$ e quindi $c \geq d + 1$.

Passo induttivo: $n > 1$:

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + 1 && \text{per ip. induttiva } T\left(\frac{n}{2}\right) \leq c\frac{n}{2} - d \\
&\leq 2\left(c\frac{n}{2} - d\right) + 1 \\
&= cn - 2d + 1 \\
&= cn - (2d - 1) \\
&\leq cn - d
\end{aligned}$$

L'ultima disuguaglianza è vera solo scegliamo d in maniera tale che $2d - 1 \geq d$ e quindi $d \geq 1$. Poichè deve essere (vedi sopra) $c \geq d + 1$ abbiamo anche che $c \geq 2$.

Infine $T(n) \leq cn - d = O(n)$ implica $T(n) = O(n)$.

12. *Si supponga che un albero binario di ricerca contenga i numeri compresi tra 1 e 200 e che si voglia ricercare il numero 67. Quali fra le seguenti sequenze non può essere esaminata e perchè*

(a) 60, 167, 144, 132, 90, 83, 64, 70, 65, 66, 67

(b) 60, 167, 144, 132, 90, 83, 64, 70, 63, 66, 67

Risposta corretta (b) (63 non è al posto giusto, perchè?).

13. *Dare una definizione formale di **max-heap***

Un max-heap binario è un albero binario quasi completo che soddisfa la proprietà del max-heap: il valore associato a ciascun nodo u deve essere maggiore del valore associato ai suoi figli

14. *Si consideri la seguente sequenza di numeri*

30 15 10 45 16 11 13 24 20 31

*Assumendo che siano memorizzati in un array secondo la notazione posizionale, discutere motivando la risposta se sequenza rappresenta un max-heap. Nel caso in cui la sequenza non sia si applichi la procedura **BuildMaxHeap** per costruire un heap contenente gli stessi elementi*

La sequenza non è banalmente un max-heap (ad esempio, perchè il nodo 15 viola la proprietà del max-heap, si veda Fig 2). La sequenza di trasformazioni che consento di ottenere un max-heap è descritto in Fig 3, ..., Fig 6

15. *Si illustri la procedura di ordinamento del seguente max-heap 80 77 67 33 20 23*

16. *Cosa si intende per requisito di uniformità semplice di una funzione hash?*

Ogni chiave deve avere la stessa probabilità di essere associata a uno qualsiasi degli m possibili slot (qui, m è la dimensione della tabella), indipendentemente dallo slot a cui sarà associata una qualsiasi altra chiave

17. *Si consideri una tabella hash di dimensione $m = 10$ inizialmente vuota. Si mostri il contenuto della tabella dopo aver inserito la seguente sequenza di valori 34 12 36 19 16 24 17 27 25 15. Si assuma che le collisioni vengano gestite mediante indirizzamento aperto utilizzando come funzione hash $h(k, i) = (h'(k) + i + i^2) \bmod m$ dove la funzione hash ordinario $h'(k) = k \bmod m$. Riuscireste ad inserire 13 come ulteriore elemento? Fornire l'elenco delle posizioni scandite durante il tentativo di inserimento di 13*

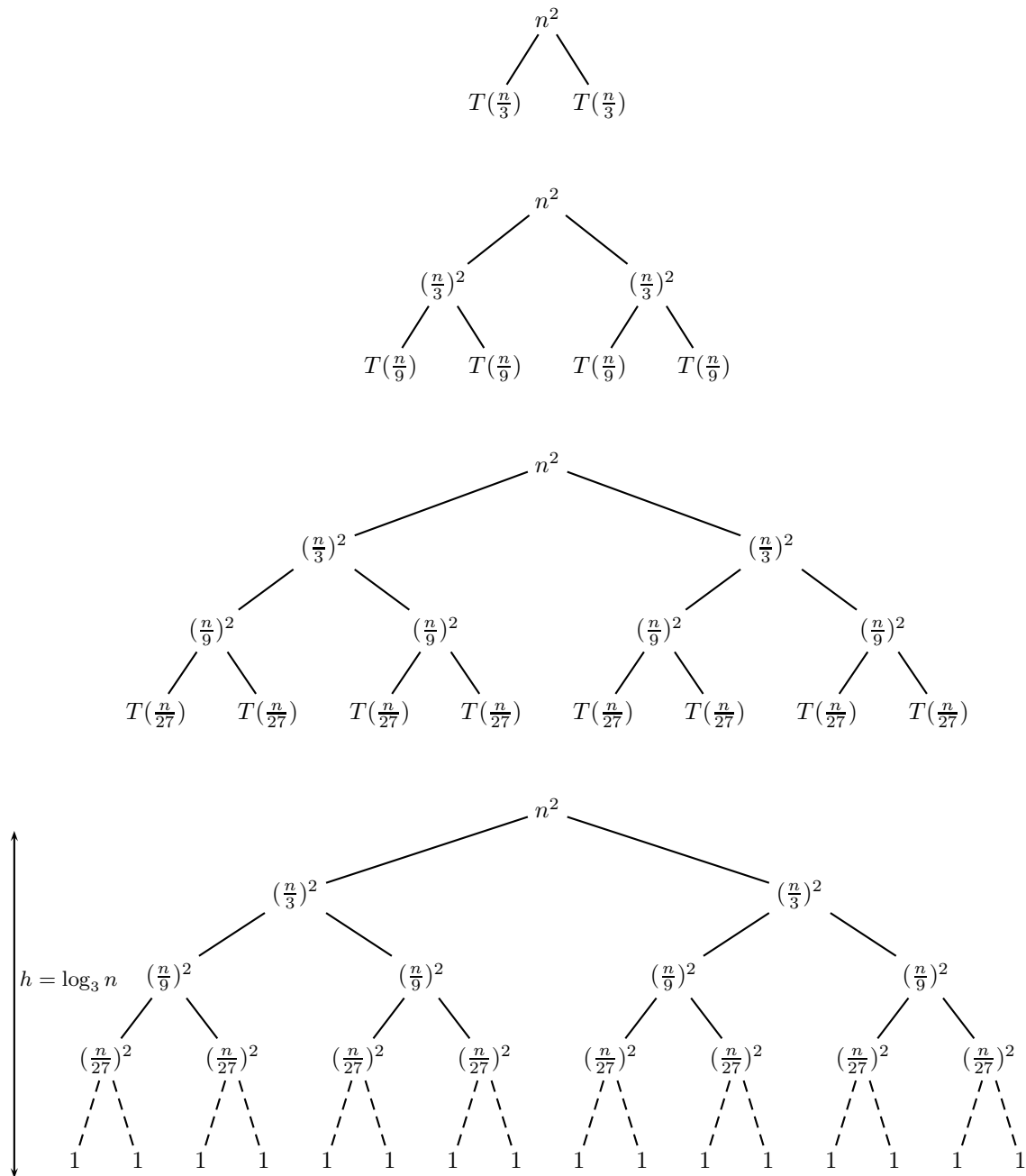


Figure 1: *Albero di ricorsione per $T(n) = n^2 + 2T(\frac{n}{3})$*

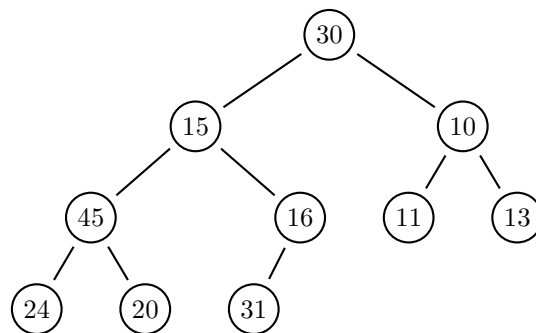


Figure 2: *una sequenza di elementi che non è un heap*

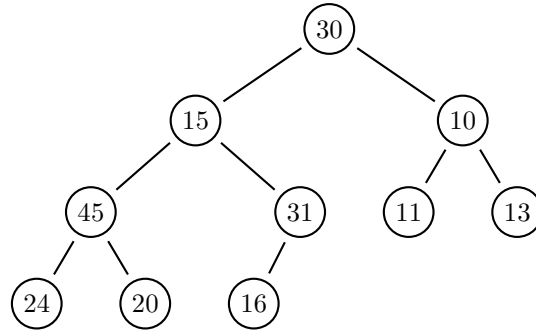


Figure 3: *Ottenuto dall'albero in Fig 2 scambiando 16 con 31*

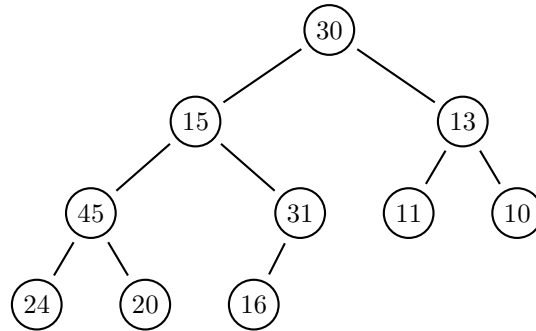


Figure 4: *Ottenuto dall'albero in Fig 3 scambiando 13 con 10*

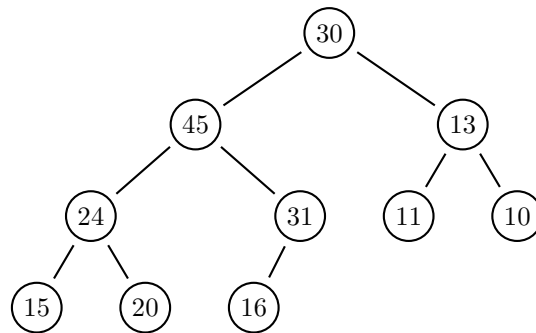


Figure 5: *Ottenuto dall'albero in Fig 4 scambiando 15 prima con 45 e poi con 24*

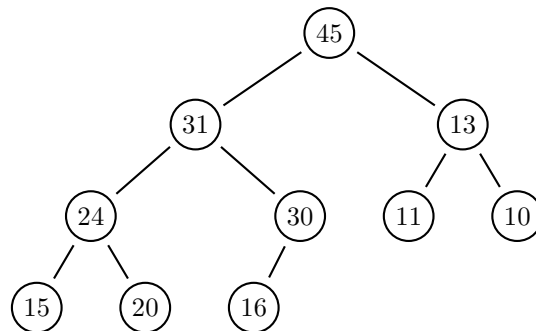


Figure 6: *Ottenuto dall'albero in Fig 5 scambiando 30 prima con 45 e poi con 31*