



POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA,
INFORMAZIONE E BIOINGEGNERIA

API

ALGORITMI E PRINCIPI DELL'INFORMATICA

BLOCCO APPUNTI

**SECONDA PARTE
ALGORITMI E STRUTTURE DATI**

Corso del prof. Dino Mandrioli • Assistente esercitatore Achille Frigeri

Leonardo Turchi

A.A. 2014/2015

Esercitazioni

ESERCITAZIONE

martedì 2 dicembre 2014 - 08:38

NOTA: QUESTI APPUNTI SONO DA
CONSIDERARSI "COME SONO".
L'AUTORE DECLINA OGNI RESPONSABILITÀ CIRCA
EVENTUALI ERRORI O MANCANZE CHE ESSI
POTREBBERO CONTENERE.
AA 2014/2015

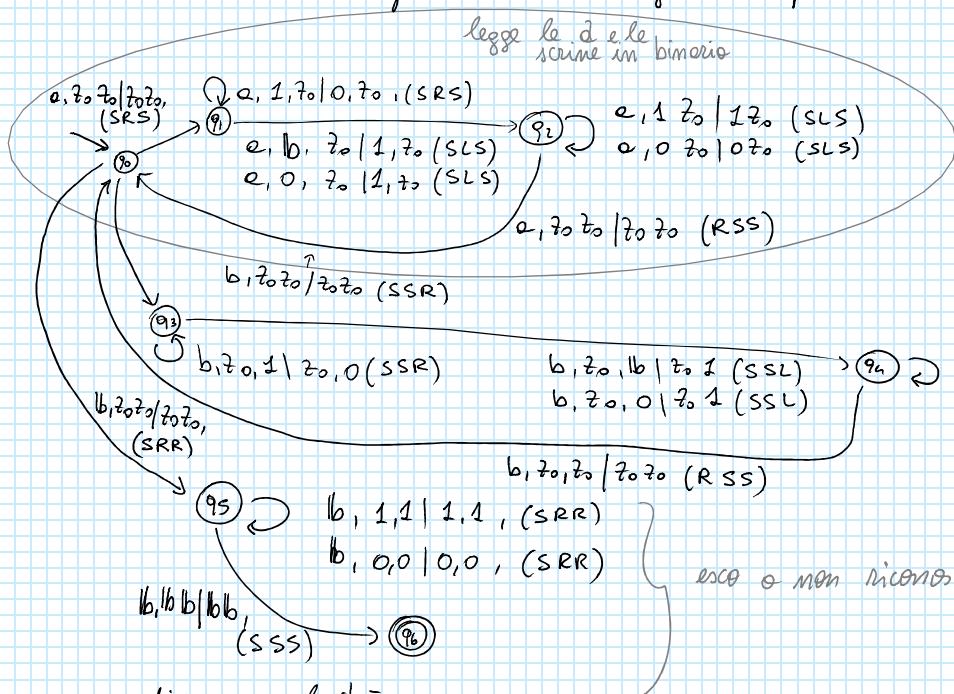
Dallo teoria alla pratica.

COMPLESSITÀ ASTRATTA

1) Definire una MT DET che riconosce $L = \{w \in \{a, b\}^* \mid \#a_w = \#b_w\}$

con complessità spaziale $\Theta(\log n)$

dove $2m$ è la lunghezza delle stringhe in input.



andiamo a calcolare la complessità:

SPAZIALE $S(m)$

se $|w| = 2m$ quando entro in q_5 i nostri binari hanno lunghezza al più $\lceil \log_2 2m \rceil + 1$ (caso $w = a^{2m}$)

quindi $S(m) = \Theta(\log_2^2 m + \log_2 m + 1) = \Theta \log_2 m$

TEMPORALE $T(m)$

Sono in un momento "generico" della computazione.
Sul nostro c'è la codifica binaria di $i \in \mathbb{N}$, leggo in input a e sono in q_0 .

- ① passo da q_0 a q_1 (ogni passaggio dura un'unità di tempo)
- ② caso pessimo: su I nostro ci sono tutti 1 ($i = 2^j - 1$)
ripeto il ciclo su q_1
 $\lceil \log_2(i-1) \rceil$ volte.
- ③ passo da q_1 a q_2
- ④ riavvio il nostro, ciclo su q_2 $\lceil \log_2(i) \rceil$
- ⑤ da q_2 a q_0
- ⑥ IDEM da 1-5 se leggo b

COMPLESSITÀ SPAZIALE
MASSIMA OCCUPAZIONE DI MEMORIA
 $a^m \cdot a^{2m} \Rightarrow A \propto a^m$
 $S(m) \propto m$
 $S(m) = \Theta(m)$

Θ grande

$$f(m) = O(g(m)) \text{ se}$$

$$\lim_{m \rightarrow \infty} \frac{f(m)}{g(m)} = l \text{ con } l \neq 0$$

faccio lo stesso per le b

④ Se ho terminato la lettura

- de q_0 e q_5
- ripetere ciclo su q_5 al più $\lceil \log_2 m \rceil$ (così penimo)
- de q_5 e q_6

$$\begin{aligned}
 T(m) &= 2 \cdot \underbrace{\sum_{i=1}^m (3 + \lceil \log_2(i-1) \rceil + \lceil \log_2 i \rceil)}_{\text{modo ordinante}} + 2 + \lceil \log_2 m \rceil \\
 &= \Theta(m \log_2 m) ? \\
 &\quad \xrightarrow{\text{con}} \sum_{i=1}^m (3 + \log(i-1) + \log(i)) \sim \\
 &\quad \sim 3 \sum_{i=1}^m 1 + 2 \sum_{i=1}^m \log_2 i = \\
 &\quad = 3m + 2 \log\left(\frac{m}{1!} \prod_{i=1}^m i\right) = \\
 &\quad = 3m + 2 \log(m!)
 \end{aligned}$$

FORMULA DI STIRLING

$$\begin{aligned}
 m! &\sim \sqrt{2\pi m} \cdot \left(\frac{m}{e}\right)^m \\
 \log_2(m!) &\sim \log_2 \left(\sqrt{2\pi m} \cdot \left(\frac{m}{e}\right)^m \right) = \\
 &= \frac{1}{2} \log_2(2\pi m) + m \log_2 \frac{m}{e} = \frac{1}{2} \log_2 2 + \frac{1}{2} \log_2 \pi + \frac{1}{2} \log_2 m + m \log_2 m - m \log_2 e \sim \\
 &\sim \frac{1}{2} \log_2 m + m \log m - m \log_2 e \sim m \log m
 \end{aligned}$$

im generale, usabile per tutti gli esercizi

$$\log m! = m \cdot \log m$$

- note gerarchia dell'infinito

Sia $\varepsilon > 0$

$$m^{1+\varepsilon} + m \log m \sim m^{1+\varepsilon} \quad \Leftrightarrow m^{\frac{3}{2}} + m \log m \sim m^{\frac{3}{2}}$$

tra i due, \log cresce sempre più lentamente (per qualsiasi esponente di m)

- note sulle basi

$$\log_c m = (\log_c n)(\log_c c)$$

poiché si può sempre cambiare base a un log. ai fini delle complessità non cambia niente se in base 2 o m penico

RIEPILOGO e CONFRONTO

MT con codifica binaria

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

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

AP

$$S(n) = \Theta(n)$$

NOTA: in AP la $S(n)$ è proporzionale a $T(n)$

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

AF

$$S(n) = \Theta(n) \quad (\text{poco di memoria, ma compl. spaziale})$$

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

CONFRONTO TRA COMPLESSITÀ:

AP vs MT

$$S(n) > S(n)$$

$$T(n) < T(n)$$

AF vs MT

$$\lambda = \alpha \sum^* \begin{cases} \text{AF : } & S(n) = 0 \\ & T(n) = \Theta(n) \\ \text{MT : } & S(n) = 0 \\ & T(n) = \Theta(1) \end{cases}$$

(quindi MT può risolvere alcuni problemi con tempo maggiore di AF, ma altri con tempo minore (costante))

2)

Sia A automo det o 2 pile

1) se A ha compl. temp. $T(n)$

quale è la compl. (minima) di una MT
o nastro singolo che simula A?

2) se la MT o m.s. ha p. temp. $T'(n)$

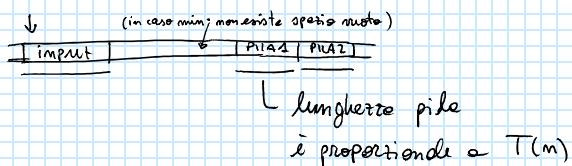
quale è la comp. di una A2P che simula la MT?

3) cosa cambia con una MT o K. mostri? ↴ il che non può dire che siano equivalenti

① MT A NASTRO SINGOLO

NOTA PER N.S.

Comp. spaziale: sono le nuove celle che uso oltre all'input che è già inserito!



$$T_{MT}(n) = T_{2P}(n) \cdot \Theta(T_{2P}(n))$$

- ↗ out. a 2 pile

$$= \Theta(T^2(m))$$

Simulare una mossa di MT con A2P A
 $\uparrow T^1(m)$

- inizializza le pile di A

MT A2P

INPUT

1b | STRINGA1 | STRINGA2 | 1b

A2P

S | T | R | I | N | G | A

S | T | R | I | N | G | A

- 2) $\left[\begin{array}{l} \text{SIMULARE una MT con un A2P non cambia la complessità} \\ \text{SIMULARE una A2P con una MT la cambia (aumenta)} \end{array} \right] \text{verificare}$

- 3) con più mostri?

- da A2P a MT 2-mostri

STESSA COMPLESSITÀ

- da MT 2-mostri a A2P

sia $T(m)$ complessità della MT e 2 mostri,

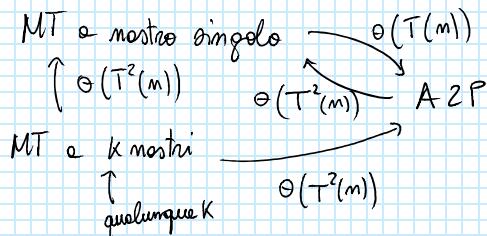
è simulata da una MT a mostro singolo

con complessità $\Theta(T^2(m))$

la quale è simulata da un A2P

con la stessa compl.

quindi la compl. dell' A2P è $\Theta(T^2(m))$



3)

$G_1 : S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow bBc \mid bc$

$G_2 : S \rightarrow KX \mid aY$

$X \rightarrow a \mid aS \mid KXX$

$Y \rightarrow K \mid KS \mid aYY$

$K \rightarrow b \mid c$

$$L(G_1) = \{ a^h b^{h+k} c^k \mid h \geq 1, k \geq 1 \}$$

→ AP det

$$|W| = m \quad T(m) = \Theta(m)$$

$$S(m) = \Theta(m)$$

$$L(G_2) = \{ w \in \{abc\}^+ \mid \#_a w = \#_b w + \#_c w \}^*$$

→ AP det

quindi

hanno lo stesso complesso

4) $L = \{a^m b^{2m}\}$ compl.?

2 modi

1. impilo 2 e e spilo 1 b per volta $\Rightarrow S(m) = 2m$ (caso pessimo)

CASO ACCETTATO $a^{\frac{m}{2}} b^{\frac{2m}{2}}$

2. impilo 1 simbolo per a e spilo 1 ogni 2b $\Rightarrow S(m) = m$

Quindi è migliore il 2

ESERCITAZIONE

martedì 9 dicembre 2014 08:34

OSS $f(m) = \Theta(g(m)) \quad (f(m), g(m) > 0, \forall m \in \mathbb{N})$

$$\exists c_1, c_2 \in \mathbb{R}^+ \quad \exists m_0 \in \mathbb{N} \text{ t.c. } \forall m \geq m_0 \quad c_1 g(m) \leq f(m) \leq c_2 g(m)$$

1) se $f(m) = \Theta(g(m))$ allora $g(m) = \Theta(f(m))$

$$g(m) \leq \frac{1}{c_1} f(m), \quad g(m) \geq \frac{1}{c_2} f(m) \rightarrow \frac{1}{c_2} f(m) \leq g(m) \leq \frac{1}{c_1} f(m)$$

2) $\boxed{\max(f(m), g(m)) = \Theta((f+g)(m))}$

quindi date 2 procedure sequenziali, la loro complessità complessiva sarà la somma delle loro complessità. Quindi come se fossero paralleli.
(combinano i fattori moltiplicativi c_1 e c_2)

$$\max(f, g) \leq f + g \leq 2 \max(f, g)$$

$$f+g = \Theta(\max(f, g))$$

3) $(m+\alpha)^b = \Theta(m^b) \quad \forall \alpha, b > 0 \text{ cost.}$

Se $m_0 > 2|\alpha|$

$$\frac{m_0}{2} \leq m_0 + \alpha \leq 2m_0$$

Se $b < 1$

$$(2m_0)^b \leq (m_0 + \alpha)^b \leq \left(\frac{m_0}{2}\right)^b$$

$$\underbrace{(2^b)}_{c_1} \cdot m_0^b \leq (m_0 + \alpha)^b \leq \underbrace{\left(\frac{1}{2}\right)^b}_{c_2} (m_0)^b$$

Se $b > 1$

$$\underbrace{\left(\frac{1}{2}\right)^b}_{c_1} (m_0)^b \leq (m_0 + \alpha)^b \leq \underbrace{2^b}_{c_2} (m_0)^b$$

CONFRONTARE

1) $m!$ con e^m

OSS $m! \geq \left(\frac{m}{2}\right)^{\frac{m}{2}}$

$$\lim_{m \rightarrow +\infty} \frac{m!}{e^m} \geq \lim_{m \rightarrow +\infty} \frac{\left(\frac{m}{2}\right)^{\frac{m}{2}}}{e^m} =$$

$$\underbrace{1 \cdot 2 \cdot 3 \cdot \dots \cdot \left(\frac{m}{2}\right)}_{\frac{m}{2}} \underbrace{\left(\frac{m}{2} + 1\right) \cdot \dots \cdot (m-1) \cdot m}_{\frac{m}{2}}$$

$$= \lim_{m \rightarrow +\infty} e^{\frac{m}{2} \log \frac{m}{2} - m} = \lim_{m \rightarrow +\infty} e^{\frac{m}{2} \underset{+\infty}{\downarrow} (\log \frac{m}{2} - 2)} = +\infty$$

quindi

$$\boxed{m! = \Omega\left(e^{\frac{m}{2} \log \frac{m}{2}}\right) = \Omega(e^m)}$$

ma $m! \neq \Theta(e^m)$

ricordando

$$m = \Omega(\log m)$$

$m = \mathcal{O}(\log \log m) \leftarrow$ cresce più lentamente

2) e^m con $m 2^m$

$$\lim_{m \rightarrow +\infty} \frac{e^m}{m 2^m} = \lim_{m \rightarrow +\infty} \frac{2^m (\log_2 e)}{2^m (\log_2 m + 1)} = \lim_{m \rightarrow +\infty} 2^{m \log_2 e - m - \log_2 m} \cdot 2^{\log_2 m} \cdot 2^m =$$

$$= \lim_{m \rightarrow +\infty} 2^{m(\log_2 e - 1)} - \log m = +\infty$$

quindi

$$e^m = \mathcal{O}(m 2^m) \text{ ma } e^m \neq \Theta(m 2^m)$$

3) $m \log_2 m$ con $\log_2 m!$

$$\left(\frac{m}{2}\right)^{\frac{m}{2}} \leq m! \leq m^m$$

$$\log_2 \left(\left(\frac{m}{2}\right)^{\frac{m}{2}} \right) \leq \log m! \leq \log_2 (m^m)$$

$$\frac{m}{2} \log \left(\frac{m}{2}\right) \leq \log(m!) \leq m \log(m)$$

$$\frac{1}{2}m((\log_2 m) - 1) \leq \log(m!) \leq m \log m \text{ per } m \geq 6 \text{ è } \geq \frac{1}{6}m \log(m)$$

$$\log(m!) = \Theta(m \log m)$$

4) Sia L_1 e L_2 linguaggi.

L_1 riconoscibile da MT a K nastri M_1 con comp. $\Theta(f_1(m))$

L_2 " " " " " M_2 " " $\Theta(f_2(m))$ note: non dire che termina in f_i , gli m poss.

e la compleltà di riconoscimento di L_1 è $\mathcal{O}(f_1(m))$

" " " " " " L_2 è $\mathcal{O}(f_2(m))$

ricordando il thr di acceleraz. lineare:

$$\forall c > 0 \text{ ric. } L_1 \text{ con } \Theta(\max(\underbrace{c \cdot f_1(m)}, m+1)) \text{ con } K+1 \text{ nastri}$$

DOMANDE:

DATO: $L_1 \cap L_2$ è riconoscibile con MT a K nastri con

A) comp. $\Theta((f_1 + f_2)(m))$? (nota: non sempre equivalente alle somme, ma anche maggiore)

S1 (crea una nuova MT che finisce la computazione di M_1 , prosegue sequenzialmente con M_2 ; se accette entrambi è 1 e le compleltà si sommano)

B) con comp. $\Theta(\max(f_1, f_2))$?

S1 (pone fare come prima ma simulare in parallelo sia M_1 che M_2 - ad es. mettendoli nelle transizioni pari M_1 e dispari M_2)

c) non è possibile riconoscere $L_1 \cap L_2$ con compl.

$$f_3(m) < \Theta((f_1 + f_2)(m))$$

FALSO

basta che sia $L_1 \cap L_2 = \emptyset \leftarrow$ così non c'è più limite inferiore
e lo si riconosce subito.
oppure regolare e f_1 e f_2 super lineari

d) le risposte cambiano se al posto di MT o k-matrici
uso MT o matrice singola oppure macchine RAM?

NO

5)

Calcolare le complessità temporali / spaziali (minime)
sul modello di calcolo meno potente
necessario a riconoscere i linguaggi

a) $L_1 = \{a^m b^{\lfloor \frac{m}{2} \rfloor} c^{\lceil \frac{m}{2} \rceil} \mid m \geq 1\}$

MT o 2 matrici

I modo

con codifica unaria di m

$$T(m) = \Theta(m)$$

$$S(m) = \Theta(m)$$

II modo

con cod. binaria di m
(ogni volta che leggo una a scrivo il
numero di a letti in binario (con +1 nei vari dispon),
poi faccio per le b (-1 per il reporta),
poi scrivo le c)

$$S(m) = \Theta(\log_2 m)$$

$$T'(m) = \Theta(m \log_2 m)$$

(caso peggiore, tutti 1)

b) $L_2 = \{a^m b^m \mid m \geq 1\} \cup \{a^m c^m \mid m \geq 1\}$

che è riconosciuto da un AP DET

quindi:

$$S(m) = \Theta(m)$$

$$T(m) = \Theta(m)$$

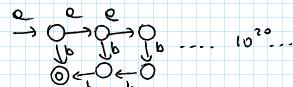
c) $L_3 = \{a^m b^m \mid 1 \leq m \leq 10^{20}\}$

linguaggio regolare (FINITO) \Rightarrow AF

quindi

$$T(m) = \Theta(m)$$

$$S(m) = \Theta(m)$$



d) $L_4 = \{a^m b^m c^m \mid m \geq 1\}$

MT o 2 matrici

cod. unaria: $S(m) = T(m) = \Theta(m)$

cod. binaria: $T(m) = m \log m$
 $S(m) = \log m$

Nota: $m \rightarrow \infty \quad m < n \log n$
quindi l'unario è più veloce nel
caso temporale.

e) Esistono famiglie di macchine M_1 e M_2 tali che la classe dei linguaggi riconosciuti da M_2 sia strettamente contenuta in quella di linguaggi riconosciuti da M_1 ma esiste un linguaggio riconosciuto da M_1 e M_2 con complessità temporale minore di quella richiesta su ogni M_2 e M_1 che lo riconosce?

↓

Penso escludere subito AF e MT e i nostri.

Potrebbero essere: AP o MT o nastro singolo (che simula la pila di un AP)

es.

$$L = \{ w \in \{a, b\}^* \mid w \in \{a, b\}^* \}^* \rightarrow \begin{array}{l} \text{su AP DET } T(m) = \Theta(m) = S(m) \\ \text{su MT A.N.S. } T(m) = \Theta(m^2) \end{array}$$

§)

$$L = \{ a^n b^n c^n \mid n > 1 \}$$

Descrivere in dettaglio una MT A NASTRO SINGOLO che riconosce L con minima complessità temporale (e spaziale)

I IDEA (unario)

b a ... a | b ... b | c ... c

posso cancellare via via le lettere finali e sostituirle con un altro simbolo.

quindi

$$T(m) = \Theta(m^2)$$

$$S(m) = \emptyset$$

II IDEA (binario)

alla I-erima posso cancellare $\frac{m}{2}$ simboli

PRIMO
GIRO

a ... a b ... b c ... c

* a * a * a \$ b ...

↑ l'ultima a la sostituisce con un altro simbolo

praticamente è già più veloce del I perché nello stesso momento che trova uno o più oranti, nel momento che scatta, sostituisce la a precedente con *

ho fatto praticamente $7:2=3 \frac{r=1}{(2^0)}$

SECONDO
GIRO

* a * a * a \$ * b * b * b \$ * c ... faccio quindi

$$3:2=1 \frac{r=1}{(2^1)}$$

↓

$\Theta(m) + 2^1$

TERZO
GIRO

continuo praticamente a dividere per 2 contando il resto

* * * a * \$ * * * b \$ \$ \$ * c ...

se sono rimasti solo \$ e * accetti
se ci sono anche a o b o c rifiuti

→ QUINDI DEVO REITERARE FINCHE' NON RIMANGONO SOLO * e \$
LA COMPLESSITA' SARÀ:

NOTA: nel binario risultante gli * sono il numero di

0, i \$ il numero di 1

es. 101101101

... 2^5 \$ 2^4 \$ 2^3 \$ 2^2 \$ 2^1 \$

i va da 0 fino a $\log m$

$$\text{totale } \sum_{i=1}^{\log m} \log(m) \cdot (\Theta(m) + 2^i) =$$

$$= m \log m + \log m \left(\sum_{i=1}^{\log m} 2^i \right) = \Theta(m \log m)$$

$$\frac{1 - 2^{\log m}}{1 - 2} = \frac{1 - m}{-1} = m - 1$$

6) ES. PSEUDOCODICE

```
int *a; int *b;  
int i = 2;  
while (i <= m) {  
    b[i] = a[i];  
    i++;  
    i = i * i;  
}
```

- valore di i per ogni ciclo

$$\begin{array}{lll} i = & 2, & 2^2, & (2^2)^2 ((2^2)^2)^2 \dots 2^{(2^K)} \\ \text{ciclo} & 0 & 1 & 2 & 3 & \dots & K \end{array}$$

- quindi il while è eseguito finché

$$2^{2^K} \leq m$$

$$K \leq \log_2(\log_2 m) \quad K = \lceil \log(\log m) \rceil$$

7)

```
int s=0, j=0, k=0;  
for (int i=0; i <= m; i++) {
```

$$j = 1;$$

```
while (j < m) {
```

$$\begin{array}{llllll} j & 1 & 2 & 4 & 8 & \dots & 2^K \\ \text{ciclo} & 0 & 1 & 2 & 3 & \dots & K \end{array}$$

$$k = 1;$$

```
while (k < m) {
```

$$s++$$

$$k = 3 * k;$$

```
}
```

$$j = 2 * j;$$

```
} }
```

complexità

$$\Theta(m \log_2^2 m) = \Theta(m \log_2 m \cdot \log_3 m)$$

perché $\log_a b = \log_c b \log_c a$

ESERCITAZIONE

martedì 16 dicembre 2014 08:37

1) COMPLESSITÀ A COSTO COSTANTE:

```

int a=0, j=1, k=0, h=0;
while (j < N) {
    k++;
    for (int i=0; i <= K; i++) {
        h = 2;
        while (h < 2^N) {
            a++;
            h = h * h;
        } // WHILE
    } // FOR
    j = 2 * j
} // WHILE

```

ANALISI:

- ciclo esterno eseguito:

$$\begin{array}{l}
 \text{j : } 1 \ 2 \ 4 \ 8 \ \dots \ 2^m \\
 \text{ripetizioni: } 1 \ 2 \ 3 \ 4 \ \dots \ m \\
 \text{finché } j = 2^m < N \rightarrow m < \log_2 N
 \end{array}
 \quad \xrightarrow{\hspace{1cm}} \text{cresce } \log N \text{ volte.}$$

- K assume (tutti) i valori da 1 fino a $\log N$
- il ciclo for viene eseguito $\log N$ volte (segue K, ogni volta)
 - ogni volta la parte interna del for viene eseguita K volte
$$\sum_{i=1}^{\log N} i = \frac{\log N (\log N + 1)}{2} = \Theta(\log^2 N)$$

- ciclo while interno:

$$\begin{array}{l}
 \text{h: } 2 \ 4 = 2^2 \ 2^2 \cdot 2^2 = 2^4 \ 2^4 \cdot 2^4 = 2^8 \ \dots \ 2^{2^m} \\
 \text{rip: } 0 \ 1 \ 2 \ 3 \ \dots \ m \\
 \text{finché } h = 2^{2^m} < 2^N \rightarrow 2^m < N \rightarrow m < \log N
 \end{array}
 \quad \xrightarrow{\hspace{1cm}} \text{eseguito } \log N \text{ volte.}$$

- COMPLESSIVAMENTE

while : $\log N$ volte

for: K volte, per K da 1 a $\log N$

while : $\log N$ volte

quindi

$$\Theta(\log N \cdot (\log^2 N \cdot (\log N))) = \Theta(\log^4 N)$$

- Determinare con criterio di costo logaritmico le complessità di:

```

1. int a = 1;
2. for (int j = 1; j <= N; j++) {
3.     a = a * j;
}

```

• con criterio di costo costante: $T(N) = \Theta(N)$

• con criterio di costo logaritmico:

- memorizzazione 1 o 1000 ha un diverso costo, quindi deve verificare la tendenza della memoria usata.

- 1. ha costo sempre fisso $\Rightarrow l(1)$

note: la funzione $l(n)$

2. $a : 1 \ 2 \ 6 \ 24 \dots n!$

restituisce il n^{o}
di bit che n occupa.

$j : 1 \ 2 \ 3 \ 4 \dots n$

(calcola il fattoriale)

3. dato da a e da $a * j$

$l(a) + l(a * j)$

$\log((j-1)!) + \log(j!)$

- IN TOTALE

$$\begin{aligned} l(1) + \sum_{j=1}^N l((j-1)!) + l(j!) &= \\ &= \Theta\left(\sum_{j=1}^N \log(j!)\right) = \Theta\left(\sum_{j=1}^N j \log j\right) \end{aligned}$$

one

$$\sum_{j=1}^N j \log j \sim \int_1^N x \log x \, dx = \left[\frac{x^2 \log x}{2} - \frac{x^2}{4} \right]_1^N$$

per N grande
la tendenza è data da
questo termine.

quindi:

$$\Theta\left(\sum_{j=1}^N j \log j\right) = \Theta(N^2 \log N)$$

3) Valutare la complessità dell'algoritmo con criterio di costo uniforme:

```

1. int alg (int n) {
2.     int fett = 0;
3.     if (n == 0) {
4.         fett = 1;
5.     } else {
6.         int m = n - 1;
7.         ...
}

```

```

5.     else {
6.         int m = m-1;
7.         fett = alg(m) * m;
8.     }
9.     return fett;
}

```

NOTA: algoritmo ricorsivo.

$$\text{SE } m=0 \rightarrow T(m) = 5 \quad \text{costante}$$

$$\text{SE } m>0 \rightarrow T(m) =$$

1 lines 1

1 lines 2

1 lines 3

1 lines 5

1 lines 6

T(m-1) lines 7

1 lines 8

$$\text{TOT} \quad 6 + T(m-1)$$

$$T(m) = \begin{cases} 5 & m=0 \\ 6 + T(m-1) & m>0 \end{cases}$$

↑
RICORSIONE

Cerco di togliere la ricorsione:

$$\begin{aligned}
 \text{modo 1)} \quad T(m) &= 6 + T(m-1) \\
 &= 6 + (6 + T(m-2)) = \\
 &= 6 + (6 + (6 + T(m-3))) \\
 &\underbrace{6 + (6 + (6 + (\dots (6 + 5))))}_{m \text{ volte}} = 6m + 5 = \Theta(m)
 \end{aligned}$$

modo 2)

$$\frac{T(m) - T(m-1)}{1} = 6 \underset{h \rightarrow 0}{\sim} \lim_{h \rightarrow 0} \frac{T(m) - T(m-h)}{h} = 6$$

$(\text{con } m \in \mathbb{R})$

$$T'(m) = 6$$

MASTER THEOREM

① Lavoro di combinazione indipendente da m

a) relazione lineare di ordine h

$$\begin{cases} f(1) = c_1, \dots, f(h) = c_h \\ f(m) = \alpha_1 f(m-1) + \alpha_2 f(m-2) + \dots + \alpha_h f(m-h) + b \quad (m > h) \end{cases}$$

Allora $f(m) = O(\alpha^m)$ per qualche $\alpha > 1$

b)

$$\begin{cases} f(1) = c_1, \dots, f(h) = c_h \\ f(m) = f(m-h) + b \quad (m > h) \end{cases}$$

Allora $f(m) = \Theta(m)$

c) partizione dei dati

$$\begin{cases} f(1) = c \\ f(m) = \alpha f\left(\frac{m}{k}\right) + b \quad (m > 1) \end{cases}$$

allora

$$\begin{aligned} \textcircled{I} \text{ se } \alpha = 1 &\rightarrow f(m) = \Theta(\log m) \\ \textcircled{II} \text{ se } \alpha > 1 &\rightarrow f(m) = \Theta(m^{\log_k \alpha}) \end{aligned}$$

(2)

Lavoro di combinazione proporzionale m

a) rel lineare di ordine m

$$\begin{cases} f(1) = c, \dots, f(h) = ch \\ f(m) = f(m-h) + bm + d \quad (m > h) \end{cases}$$

allora $f(m) = \Theta(m^2)$

b) partizione dei dati

$$\begin{cases} f(1) = c \\ f(m) = \alpha f\left(\frac{m}{k}\right) + bm + d \quad (m > 1) \end{cases}$$

allora

$$\begin{aligned} \textcircled{I} \text{ se } \alpha < k &\rightarrow f(m) = \Theta(m) \\ \textcircled{II} \text{ se } \alpha = k &\rightarrow f(m) = \Theta(m \log m) \\ \textcircled{III} \text{ se } \alpha > k &\rightarrow f(m) = \Theta(m^{\log_k \alpha}) \end{aligned}$$

c) Analizzare la complessità dell'algoritmo di ricerca binaria

```

int bimSrc(ITEM[] A, ITEM v, int i, int j) {
    if (i > j)
        return 0;
    else {
        int m = (i + j) % 2;
        if (A[m] == v)
            return m;
        else if (A[m] < v)
            return bimSrc(A, v, m + 1, j);
        else
            return bimSrc(A, v, i, m - 1);
    }
}

```

↑
tipi di dato per cui esiste
un ordinamento totale ≤
Array ordinato

COSTI

- c₁ if ($i > j$)
- c₂ return 0;
- c₃ else {
- c₄ int $m = (i + j) \% 2$;
- c₅ if ($A[m] = v$)
- c₆ return m ;
- c₇ else if ($A[m] < v$) {
- c₈ $\frac{1}{2} T\left(\left[\frac{N-1}{2}\right]\right)$ return bimSrc(A, v, $m + 1, j$);
può dividere
array in 2
 }
- c₉ else
 $\frac{1}{2} T\left(\left[\frac{N}{2}\right]\right)$ return bimSrc(A, v, $i, m - 1$);

- per semplicità posso supporre che $N = 2^k$
 - caso perimmo: $n \notin A$
e nella divisione prendo sempre la porzione dx dell'array che è lunga $\left\lfloor \frac{N}{2^k} \right\rfloor$
 - $i > j$ ($m \leq 0$) $\rightarrow T(m) = c_1 + c_2 = c$
 - $i \leq j$ ($m > 0$) $\rightarrow T(m) = \underbrace{c_1 + c_3 + c_5 + c_7 + c_9 + c_{10}}_b + T\left(\frac{m}{2}\right)$
- $$\Rightarrow T(m) = \begin{cases} c & m=0 \\ T\left(\frac{m}{2}\right) + b & m>0 \end{cases}$$

MODO 1)

CASO 1 del M.THR. ($c \neq 0$)

$$\Rightarrow T(m) = \Theta(\log m)$$

MODO 2)

espongo la ricorrenza

$$(m = 2^k)$$

$$\begin{aligned} T(m) &= T\left(\frac{m}{2}\right) + b = T\left(\frac{m}{2} + b\right) + b = \\ &= \left(\left(T\left(\frac{m}{4}\right) + b \right) + b \right) + b = \dots = \\ &= \left(\dots \left(\underbrace{T\left(\frac{m}{2^k}\right) + b}_{\substack{\text{1} \\ \text{K volte}}} \dots \right) + b \right) = \\ &= \left(\left((b+c) + b \right) \underbrace{\dots + b}_{\substack{\text{K volte}}} \right) = (k+1)b + c = \\ &= b(\log m) + b + c = \Theta(\log m) \end{aligned}$$

5) Analizzare il costo delle moltiplicazione tra interi in base 2.

1) MODO SEMPLICE

dati m e n

si può sommare m per n volte per n volte.

$\rightarrow \Theta(n^2)$ operazioni elementari su bit

$$\left[\begin{array}{l} \text{costo } A_{10} \cdot B_{10} \rightarrow \Theta(\log^2 A) \quad \text{es. in base 10} \\ A \sim B \end{array} \right]$$

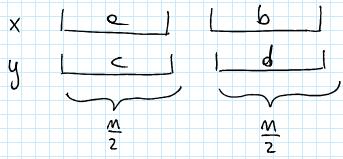
2) MODO ALGORITMICO NOTO

DIVIDE ET IMPERA

input $x, y \leftarrow$ stesse lunghezze
codifica binaria di due interi

sperro x e y in 2 sottastrinque

spostare x e y in 2 sottastrutture



$$x = a \cdot 2^{\frac{m}{2}} + b$$

$$y = c \cdot 2^{\frac{m}{2}} + d$$

vuol dire spostare gli uno a sinistra
(il resto è trascurabile)

$$\begin{bmatrix} A_2 \cdot 2^{\frac{m}{2}} & \text{shift a sx} \\ A_2 / 2^{\frac{m}{2}} & \text{shift a dx} \end{bmatrix}$$

$$x \cdot y = a \cdot c \cdot 2^m + (ad + bc) 2^{\frac{m}{2}} + bd$$

cost1 bool[] dei (bool[] x, bool[] y, int m)

```
c1      if (m == 1)
c2      return x[1] * y[1];
c3..m   else {
        "sposta x in a e b"
        "y in c e d"
    }
```

$$c_4 \quad \text{return } \text{dei}(a, c, \frac{m}{2}) \cdot 2^m + \\ 4 \cdot T\left(\frac{m}{2}\right) + c_4 \cdot m + c_5 \quad \left(\text{dei}(a, d, \frac{m}{2}) + \text{dei}(b, c, \frac{m}{2}) \right) \cdot 2^m + \\ \text{dei}(b, d, \frac{m}{2});$$

quindi:

$$T(m) = \begin{cases} c & (m=1) \\ c_3 + c_4 + T\left(\frac{m}{2}\right) + b_m + d & (m>1) \end{cases}$$

per M.THR

$$(2.B.III) \rightarrow T(m) = \Theta(m^{\log_2 4}) = \Theta(m^2)$$

3) MODO PRODOTTO GAUSSIANO

$$\text{input : } \begin{array}{c} x \quad \underline{a} \quad \underline{b} \\ y \quad \underline{c} \quad \underline{d} \end{array}$$

calcolo: $A_1 = ac \leftarrow$ prodotto tra str. di lunghezza $\frac{m}{2}$

$A_3 = bd \leftarrow$ " " " " "

$m = (a+b)(c+d) = ac + ad + bc + bd \leftarrow$ prod. fra str. di lunghezza $\frac{m}{2} + 2$ somme

$A_2 = m - (A_1 + A_3) = ad + bc \leftarrow 2$ somme

$$x \cdot y = A_1 \cdot 2^m + A_2 \cdot 2^{\frac{m}{2}} + A_3 = ac \cdot 2^m + (ad + bc) 2^{\frac{m}{2}} + bd$$

bool[] gauss (bool[] x, bool[] y, int m)

```

c1 if (m == 1)
c2     return x[1] y[1];
c3 else {
    perito..
}

```

$T\left(\frac{m}{2}\right)$ bool [] $A_1 = \text{gours}(a, c, \frac{m}{2})$

$T\left(\frac{m}{2}\right)$ bool [] $A_2 = \text{gours}(b, d, \frac{m}{2})$

$T\left(\frac{m}{2}\right) + c_4 \cdot m$ bool [] $m = \text{gours}(a+b, c+d, \frac{m}{2})$

$c_5 \cdot m$ $A_2 = m \cdot (A_1 + A_3)$

$c_7 \cdot m$ return $A_1 \cdot 2^m + A_2 \cdot 2^{\frac{m}{2}} + A_3$

quindi:

$$T(m) = \begin{cases} c & m=1 \\ 3 T\left(\frac{m}{2}\right) + b m + d & m>1 \end{cases}$$

$$T(m) = \Theta\left(m^{\log_2 3}\right) \approx m^{1.58..}$$

NOTA

MIGLIOR ALGORITMO AD OGGI NOTO PER CALCOLO DI MOLT. TRA 2 NUMERI

$$\text{Furer (2007)} = \Theta(m \log m \cdot 2^{O(\log^* m)})$$

$$\text{con } \log^* m = \begin{cases} 0 & m \leq 1 \\ 1 + \log^*(\log m) & m > 1 \end{cases}$$

ESERCITAZIONE

martedì 23 dicembre 2014 08:30

1) Macchina RAM

(CRITERIO DI COSTO LOGARITMICO)

READ 0	loop	LOAD 1
STORE 1		MULT 3
STORE 2		STORE 3
LOAD = 1		LOAD 2
STORE 3	M-rotate	SUB = 1
		STORE 2
		JGZ loop
		WRITE 3
		HALT

0	1	2	3
$m-1$	m	$m-3$	m^3
...	cicli
0	m	0	m^m

Complessità:

SPAZIALE: la massima occupazione del registro principale (σ) si ha alla fine, quando c'è m^m

$$S(m) = \Theta(\ell(m^m)) = \Theta(m \log m)$$

TEMPORALE:

costo costante: $T_c(m) = s + (m \cdot 6) + 2 = \Theta(m)$

costo logaritmico:

$$\begin{aligned}
 T_l(m) &= \underbrace{\ell(0) + \ell(m) + \ell(1) + \ell(m) + \ell(2) + \ell(m)}_{\substack{\text{ISTRUZIONI} \\ 1, 2, 3, \dots}} + \underbrace{\sum_{i=0}^{m-1} \ell(m) + \ell(3) + \ell(m^{i-1}) + \ell(m^i) + \ell(3) + \ell(2) + \ell(m-1) +}_{\substack{\text{ISTRUZIONI} \\ 1, 2, 3, \dots}} \\
 &\quad \underbrace{\ell(1) + \ell(m-i-1) + \ell(2) + \ell(4)}_{\substack{\text{ISTRUZIONE FINALE} \\ HALT}} \\
 &+ \underbrace{\ell(3) + \ell(m^m)}_{\substack{\text{loop}}} + \underbrace{\ell(1)}_{\substack{\text{loop} \\ HALT}} \\
 &= c + 3 \log(m) + m \log(m) + \\
 &\quad \sum_{i=1}^m (c' + \log m + 2 \log(m-i) + 2 \log(m^i)) \\
 &= c + 3 \log(m) + c'm + m \log m + 2 \left(\sum_{i=1}^m \log(i) + i \log(m) \right) \\
 &= \Theta(m \log m + m^2 \log m) = \Theta(m^2 \log m)
 \end{aligned}$$

RICORDANDO CHE

$$\sum_{i=1}^m \log(i) = \log \prod_{i=1}^m i = \log m! = \Theta(m \log m)$$

2)

Determinare la migliore comp. temporale ottenibile per implementare

la funzione $f: \mathbb{N} \rightarrow \mathbb{N}$, $f(n) = 2n$ (funzione "doppia")

Su macchina RAM (con costo logaritmico)

e su MT o NASTRO SINGOLO con codifica BINARIA dell'input.

RAM

1. READ 0
2. MULT = 2
3. WRITE 0
4. HALT

$$T(n) = \frac{\ell(0) + \ell(m) + \ell(m) + \ell(2) + \ell(2m) + \ell(0) + \ell(1)}{\text{ISTRUZ.}} =$$
$$= \ell(2m) + 2\ell(m) + 3\ell(1) = \Theta(\log m)$$

MT NS

b	m	0
MSB		LSB

dovrò leggere tutto m

· scrivere 0 nella prima cella libera

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

3) STRUTTURA DATI

è un modo per organizzare delle informazioni

sono ad esempio: array, liste, heap, alberi, tabelle hash, grafici...

Descrivere un algoritmo che ordini in modo crescente gli elementi di una matrice $m \times m$ (M)

ciclo: $\forall 1 \leq i \leq n, \forall 1 \leq j \leq m, M(i, j) \leq M(i, j+1)$ e

$\forall 1 \leq i \leq n, M(i, m) \leq M(i+1, 1)$

esempio:

$$\begin{bmatrix} 5 & 1 \\ 9 & 13 \\ 2 & 6 \end{bmatrix} \xrightarrow{\text{IN}} \begin{bmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 13 \end{bmatrix} \xrightarrow{\text{OUT}}$$

ALGORITMO:

ordine-mat (M, m, m)

1. crea A array di lunghezza $m \times m$

$$T(n) = m \cdot m$$

2. for ($i = 1, m, i++$)

3. for ($j = 1, m, j++$)

$$A[(i-1) * m + j] = M[i][j]$$

$$T(n) = m \cdot m$$

4. merge-sort (A)

solve la matrice in array

5. for ($i = 1, m, i++$)

6. for ($j = 1, m, j++$)

$$M[7][7] - A[7-1 * m + 7]$$

$$T(n) = m \cdot m$$

7. $\text{for } (j=1, m, j++)$

8. $M[i][j] = A[(i-1)*m + j]$

$$T(m) = m \cdot m$$

9. return

$$\begin{aligned} T(m) &= \Theta(m \cdot m + m \cdot m \log(m \cdot m)) = (m \cdot m \cdot (\log m + \log m)) = \\ &= \Theta(m \cdot m \cdot \max(\log(m), \log(m))) \end{aligned}$$

NOTA: SE M è quadrato $[m=m \wedge m=\Theta(m)]$ allora

$$T(m) = \Theta(m^2 \log(m))$$

- 4) Scrivere in pseudocodice un algoritmo che dato un array A di m elementi, determini se A contiene almeno uno terzo pitagorico e quindi se

$$\exists 1 \leq i, j, k \leq m \mid A^2[i] + A^2[j] = A^2[k]$$

pitagoro(A, m)

1. $\text{for } (i=1, m, i++)$

2. $\text{for } (j=1, m, j++)$

3. $\text{for } (k=1, m, k++)$

4. $\text{if } (A[i]*A[i] + A[j]*A[j] == A[k]*A[k])$

5. return true

6. return false

$$T(m) = \Theta(m^3) \quad A^2[i] + A^2[j] = A^2[k]$$

ordinando l'algoritmo si può ottenere una versione migliore

pit2(A, m)

cost | $\text{if}(m < 3) \text{return false}$

merge-sort($A, 1, m$)

cost | int $i = 1$

| while($i \leq m$)

| | if ($\text{somme}(A, 1, i-1, A[i])$)

| | | return true

| | else

| | | $i = i + 1$

| | return false

somme(A, m, x)

m
VOLTE

```

i = m
j = 1
while ( j < i )
    if ( A2[i] + A2[j] = x2 )
        return true
    else if ( A2[i] + A2[j] < x2 )
        j = j + 1
    else
        i = i - 1
return false

```

in questo caso $T(m) = \Theta(m^2)$

- 5) Quale è la complessità di un MERGE-SORT-G che funziona come merge sort ma spezza l'array in 4 parti?

EQUAZIONE ALLE RICORRENZE:

$$\begin{aligned}
T(m) &= \begin{cases} \text{cost.} & m=0 \\ 4T(m/4) + \Theta(m) + b m + d & \end{cases} \\
&\quad \Downarrow \qquad \leftarrow \text{CASO 2 B DEL MASTER THEOREM} \\
&= \Theta(m^{\log_4(4)} \cdot \log(m)) = \Theta(m \log m)
\end{aligned}$$

- 6) Descrivere un algoritmo per il calcolo del MAX dell'array A (del tipo divide-et-impara)

$\max(A, p, r)$

1. if ($r < p$)
2. return $-\infty$
3. $m = \left\lfloor \frac{p+r}{2} \right\rfloor$
4. $msx = \max(A, p, m)$
5. $mdx = \max(A, m+1, r)$
6. return $(\max(msx, mdx))$

QUINDI

$$T(m) = \begin{cases} \text{cost} & m=0 \\ 2T\left(\frac{m}{2}\right) + b = \Theta(m) & \end{cases}$$

$$\begin{aligned}
T(m) &= \\
&= 2T\left(\frac{m}{2}\right) + b \\
&= 2\left(2T\left(\frac{m}{4}\right) + b\right) + b \\
&= 2^{\log_2 m} + b \log m = \Theta(m)
\end{aligned}$$

- 7) Descrivere un algoritmo che dato in ingresso un array A di numeri

lungo m , ritorni True se è possibile partizionare A in coppie,
in modo che le somme di ogni coppia siano costante

es:

$$\begin{array}{cccccc} 1 & 3 & 5 & 3 & 4 & 2 \\ \cancel{V} & \cancel{V} & & V & & \\ 6 & 6 & & 6 & & \end{array}$$

A)

1 scelta	$m-1$ modi	ma completezza troppo elevata non è il modo migliore
2 scelte	$m-3$ modi	
...		

$m/2$ scelte 1 modo

B)

se A fore ordinato 1 2 3 3 4 5
potrei procedere
accoppiando via via gli estratti.



partizione (A, m)

```
if (m dispari)
    return FALSE
```

$$\begin{aligned} T(m) &= \text{cost} \\ &= \text{cost} \\ &= m \log(m) \\ &= \text{cost} \\ &= m\text{-volte} \\ &= \text{cost} \end{aligned}$$

```
merge sort (A, 1, m)
int i = 2, j = m-1
```

```
while (i < j)
```

```
    if (A[i] + A[j] ≠ A[i-1] + A[j+1])
```

```
        return false
```

```
        i = i + 1
```

```
        j = j - 1
```

```
return true
```

Allora $T(m)$ si può ridurre alla completezza del merge-sort

$$\Rightarrow T(m) = \Theta(m \log m)$$

- 8) Dato un array A di m elementi voglio stabilire se si può partizionare A in TERNE e somme costante
[sapendo che se si può fare, si può farlo in modo unico]

↓

- Prima di tutto, m deve essere multiplo di 3, e

$\sum_{i=1}^m A[i]$ deve essere multiplo di 3.

$$3 \cdot \sum_{i=1}^m A[i]$$

- Devono risultare $\frac{m}{3}$ terne, ognuna con somme $T =$

$$\text{con } T(m) = \Theta(m)$$

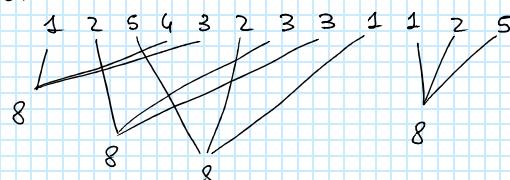
- Devono risultare $\frac{m}{3}$ terne, ognuna con somma $T = \sum_{i=1}^{i=1} M$
con $T(m) = \Theta(m)$
- Totale $\sum_{i=1}^m A[i] = T(m/3)$

ORDINO A, inizio da $i=1$ e cerco una coppia $j, k \mid A[i] + A[j] + A[k] = T$

SE NON LA TROVO, restituisco false

SE LA TROVO, cancello gli elementi in posizione i, j, k e incremento i

ESEMPIO:



partitione (A, m)

```
if ( $m$  non è multiplo di 3)
    return false
```

merge sort ($A, 1, m$)

$$T = \left(3 \cdot \sum_{i=1}^m A[i] \right) / m$$

```
if ( $T$  non è intero)
```

return false

int $i = 1$

while ($i \leq m$)

trovato = false

$j = i + 1$

$K = m$

while ($j < K$ ee trovato == false)

if ($A[j] \neq \$$ ee $A[K] \neq \$$ ee $A[i] + A[j] + A[K] = T$)

$A[K] = \$$

$A[j] = \$$

else if ($A[j] = \$$ || $A[j] + A[K] < T - A[i]$)

$j = j + 1$

else

VOLTE
 $m-i+1$

$m/3$ VOLT | $m-i$ | $\circ \quad \circ \quad -$

else

$K = K - 1$

if (trovato == false)

return false

else $i = i + 1$

while ($i \leq m$ ee $A[i] = \$$)

$i = i + 1$

return true.

QINDI

$$T(m) = \Theta(m^2)$$

ESERCITAZIONE

martedì 13 gennaio 2015 08:33

1) Selezione del k -esimo elemento di un array

Dato A array di m elementi confrontabili, il k -esimo più piccolo è quel $x \in A$ tale che al più $k-1$ elementi di A sono $< x$ e almeno k sono $\leq x$

Esempio: $A = [8, 8, 0, 5]$

8 è il terzo e quarto elemento più piccolo

Estrazione del k -esimo più piccolo si può fare in al più $\Theta(m \log m)$ e almeno $\Theta(m)$ (\leftarrow deve almeno fare una passata, per leggere e sapere cosa c'è in A)

Algoritmo di selezione (A, k) (con $|A| = m$)

Se $|A| < 50$

allora ordina A

ritorna $A[k]$

altrimenti

dividi A in $\lceil \frac{|A|}{5} \rceil$ sequenze di 5 elementi ciascuna.

ordina ogni sequenza di 5 elementi

sia M l'insieme delle mediane delle sequenze di 5 elementi

$m = \text{Selezione}(M, \lceil \frac{|M|}{2} \rceil)$

costruisci $A_1 = \{x \in A \mid x < m\}$

$A_2 = \{x \in A \mid x = m\}$

$A_3 = \{x \in A \mid x > m\}$

NOTA

SONO ARRAY

CON EVENTUALI RIPETIZIONI

se $|A_1| \geq k$

allora

$x = \text{Selezione}(A_1, k)$

altrimenti se $|A_1| + |A_2| \geq k$

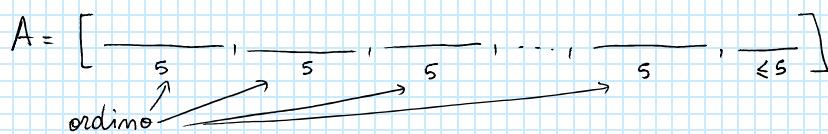
return m

altrimenti

$x = \text{Selezione}(A_3, k - |A_1| - |A_2|)$

return x

traccia di esecuzione:



$$[\underbrace{\quad}_{5}, \underbrace{\quad}_{5}, \dots, \underbrace{\quad}_{5}]$$

$m_1 \bullet, m_2 \bullet, \dots, m_k \bullet$

complessità:

- $T(m) \leq c$ se $m < 50$
- Se $m \geq 50$:
 - costruire cinquime $\Theta(m)$
 - ordinare cinquime $\Theta(m)$
 - costruire mediane M $\Theta(m)$
 - selezionare le mediane di M $T\left(\frac{m}{5}\right)$
 - costruire $A_1 A_2 A_3$ $\Theta(m)$
 - chiomate su A_1 o su A_3 $\leq T\left(\frac{3}{4}m\right)$ \leftarrow CASO PESSIMO

NOTA

CASO OTTIMO $|A_1| + |A_2| \geq k$
 CASO PESSIMO (ultimo altrimenti)

$$T(m) \leq \begin{cases} c & m < 50 \\ T\left(\frac{m}{5}\right) + T\left(\frac{3}{4}m\right) + dm & m \geq 50 \end{cases}$$

DISEQUAZIONE ALLE RICORRENZE

Tesi:

$$\forall m \in \mathbb{N}, T(m) \leq 20d \cdot m = \Theta(m)$$

Dim per induzione

- se $m < 50$ OK

(suppongo che) valga la tesi $\forall i < m$, faccio vedere che vale anche per m

$$\begin{aligned} T(m) &\leq T\left(\frac{m}{5}\right) + T\left(\frac{3}{4}m\right) + dm \leq 20d \frac{m}{5} + 20d \frac{3}{4}m + dm = \\ &= (4+15+1)dm = 20dm \quad \square \end{aligned}$$

QUINDI

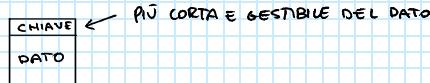
$$T(m) = \Theta(m)$$

2)

STRUTTURE DATI

Tabelle di Hash

- dizionario di chiavi $\emptyset \subseteq U$ universo
- idea: memorizzare le informazioni in modo che siano facilmente reperibili



Funzione di hash

$$f: U \rightarrow [0, 1, \dots, m-1] \text{ con } U \gg m \quad (\text{lo spazio disponibile è comunque } \gg \text{ dei dati memorizzati})$$

Tabella di hash

è un vettore di dimensione m che contiene le chiavi del dizionario

poiché $|U| \gg m$ esistono $x, y \in D$, $x \neq y$ tali che $f(x) = f(y)$

Quindi per operare con le tabelle di hash

1. costruire tabelle di hash h
2. risolvere le collisioni

↑
COLLISIONE

1)

Proprietà di h

- $h : D \rightarrow [0, m-1]$
- facile da calcolare (e preferibilmente a costo costante o \log)
- probabilità → $P(h(K_1) = h(K_2)) \leq \frac{1}{m}$ con K_1, K_2 chiavi casuali in U (distr. uniforme)

Esempi:

A) $h(x) = k \bmod m$

$x \in U = \mathbb{Z}$

B) $h(x) = \text{prime } k \text{ cifre di } x \bmod m$

C) cambiamento di base

Se $x = \sum_{i=0}^m a_i b^i$

allora $h(x) = \sum_{i=0}^n a_i b^i \bmod m$

D) folding

divido x in k blocchi di ugual lunghezza
li sommo e calcolo mod m

2) RISOLUZIONE COLLISIONI

Esistono 2 schemi

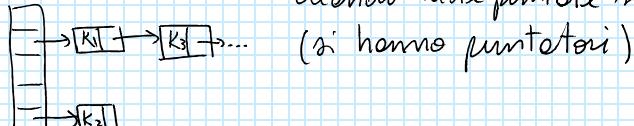
- indirizzamento aperto

in caso di collisione con un dato precedentemente inserito,

calcolo un nuovo indice (Rehash) $|K_3| |K_1| |K_2|$ (si hanno dati)

- indirizzamento chiuso

concentrazione: si uniscono i dati in collisione,
creando liste puntate sempre più lunghe



INSEGNAMENTO DI UN VALORE (x)

- posto da $h(x)$ in A

si occupa seguendo un percorso finché trova posizione libera
(si può avere clustering: affollamento di dati ai lati dell'array)

$$m = |A|$$

m numero di chiavi memorizzate

$$\alpha = \frac{m}{m}$$

Esempio

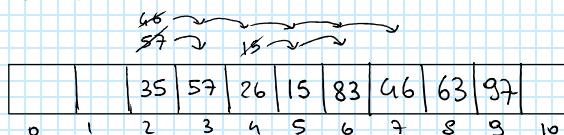
$$m = 11 \quad \text{chiavi} \quad S = \{35, 83, 57, 26, 15, 63, 97, 46\}$$

$$h(x) = x \bmod 11$$

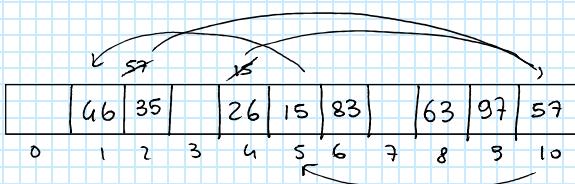
creare tab. hash A usando:

- scorrimento lineare
- hashing doppio $h_1(x) = 1 + [x]_{10}$
- hashing quadratico

ESECUZIONE CON SCANSIONE LINEARE



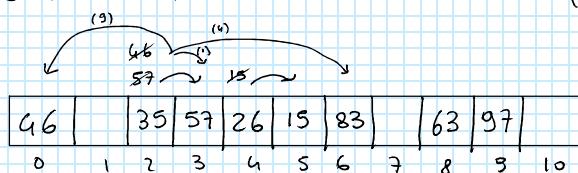
ESECUZIONE CON HASHING DOPPIO



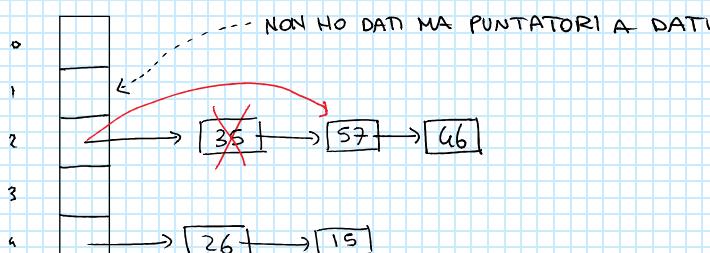
x	$h(x)$	$h_1(x)$
35	2	6
83	6	4
57	2	8
26	4	7
15	4	6
63	8	4
97	9	8
46	2	7

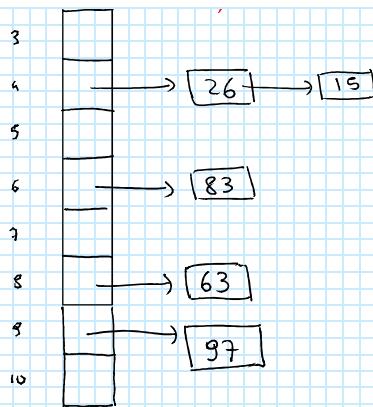
ESECUZIONE HASHING QUADRATICO

(se occupato me salto 1 poi 9, 36, 81, 25 (mod m))



INDIRIZZAMENTO CHIUSO





RICERCA ed ELIMINAZIONE CHIAVE

eliminare 35 (*in zona sopra*)

(per caso di scansione lineare: non posso limitarmi a cancellare il valore, devo risistemare tutti i valori a dx di esso e rimettere via via in ordine i in cui li trovo a partire dalle celle del vecchio valore. ovviamente nella cella 2 (caso del 25) devo mettere il primo che trovo con una compatibile)

3) ALBERI BINARI DI RICERCA (BST) del tipo



HEAP (NUCCIO)

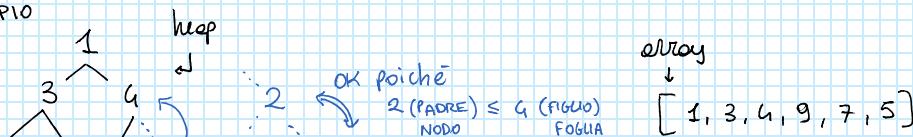
alberi bilanciati utili a gestire code con priorità

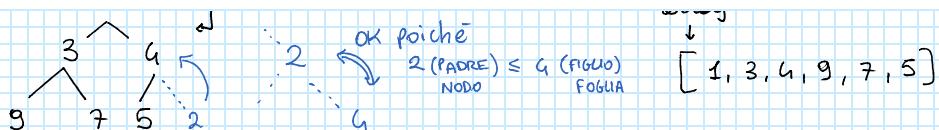
con struttura



con chiave del padre \leq a quelle dei figli
oss: la radice ha le chiavi più piccole

ESEMPIO





Simulare una heap con un array (utile nella programmazione)

$[1, \dots, m]$

radice \rightarrow pos 1

se nodo in pos. i

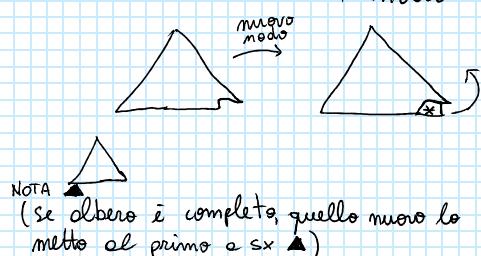
il figlio sx è in posizione $2i$

" " dx " " $2i+1$

QUINDI AVRÒ

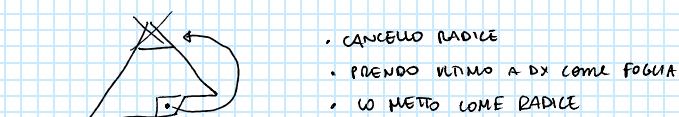
modo	posizione
radice	1
modo p	i
padre di p	$\lfloor i/2 \rfloor$
figlio sx di p	$2i$
figlio dx di p	$2i+1$

Inserimento di un nodo

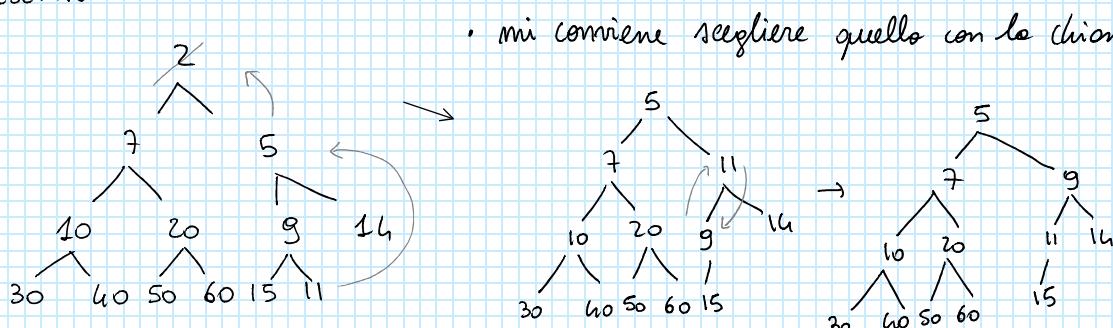


risale la heap finché il suo padre ha una chiave \leq a quella del figlio (esempio in blu sopra)

Eliminazione della radice



ESEMPIO



HEAP

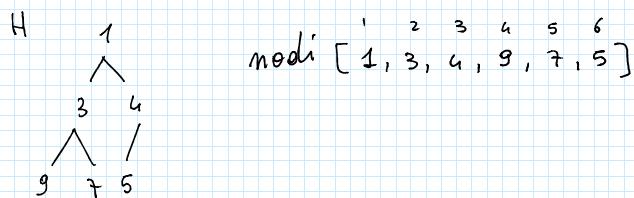
COSTRUZIONE di una min-heap

oss. se ho un insieme di chiavi con relazione d'ordine
posso costruire più heap tutte corrette

$$\text{es } S = \{7, 8, 9\} \rightarrow \begin{array}{c} 7 \\ / \quad \backslash \\ 8 \quad 9 \end{array} \quad \begin{array}{c} 7 \\ / \quad \backslash \\ 9 \quad 8 \end{array}$$

non c'è unicita'.

Teorema

Sia H una heap con m nodi memorizzata in un vettore $\text{modi}[m]$ 

Allora per ogni $1 \leq i \leq m$ i modi corrispondenti alle ultime i posizioni di nodi formano un insieme di alberi (foreste) ognuno dei quali è una heap.

(se parto dal fondo e aggiungo un nodo alla volta, avrò delle sotto-heap)

S' insieme delle chiavi, $|S| = m$

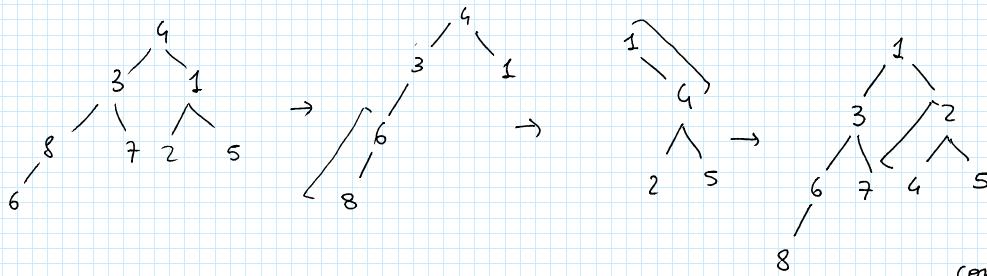
- ① Inserisco le chiavi di S a caso in un vettore $\text{modi}[m]$
- ② Per i da 1 a m permuto gli ultimi i elementi di nodi in modo che valga il teorema

NOTA: complessità

- almeno lineare
- al più $m \log m$

ESEMPIO

$$S = \{6, 3, 1, 8, 7, 2, 5, 4\}$$



HEAP VALIDA
con complessità $\Theta(m)$

Sia v nodo, chiamò $l(v)$ la massima distanza di v ad una foglia che sia suo discendente

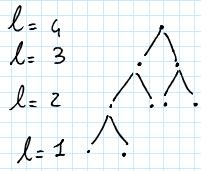
Sia m_l il numero di nodi v tale che

$$l(v) = l$$

$$m_l = |\{v \in H \mid l(v) = l\}|$$

$$\text{Allora } m_l \leq 2^{h-l} = \frac{2^h}{2^l} \leq \frac{m}{2^l}$$

con h altezza di H ($h \approx \log_2 |H|$)



$$\begin{aligned} h &= 4 \\ m_4 &\leq 2^{4-4} = 1 \\ m_3 &\leq 2^{4-3} = 2 \\ m_2 &\leq 2^{4-2} = 4 \\ m_1 &\leq 2^{4-1} = 8 \end{aligned}$$

Sia v nodo in posizione ierarca ($\text{modi}[i] = v$) tale che $l(v) = l$

Allora un eventuale spostamento di v che richiede un numero di mosse proporzionale a l .

$$\begin{aligned} T(m) &= \sum_{i=1}^h i \cdot m_i \leq \sum_{i=1}^h i \cdot \frac{m}{2^i} = m \cdot \sum_{i=1}^h \frac{i}{2^i} \leq K \cdot m \approx 2m \\ &\text{E poiché} \\ &\sum_{i=1}^{\log m} \frac{i}{2^i} = \frac{-\log m + 2^{\log m - 1} - 2}{2^{\log m}} = \frac{2 \cdot m - \log m - 2}{m} < 2 \end{aligned}$$

per il criterio
del rapporto
le serie converge

ESERCIZIO

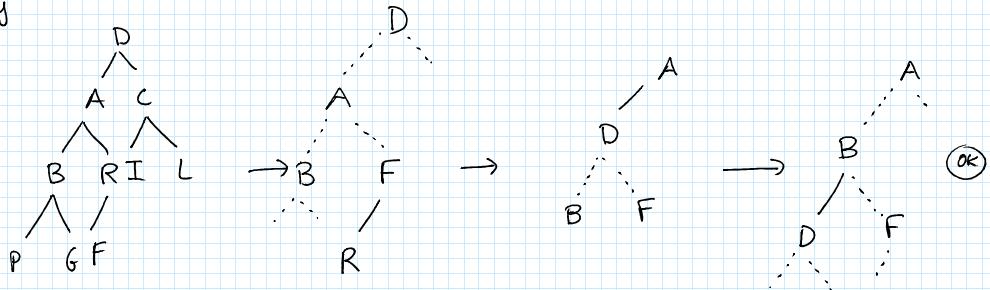
Ordinare $[D, A, C, B, R, I, L, P, G, F]$

rispettando ordinamento alfabetico.

1) creo la heap $\rightarrow \Theta(n)$

2) Per ogni i da $1 \dots n$, elimino la radice $\rightarrow \Theta(n \log n)$

creo heap da array



adesso via via si elimina la radice, riportandola in ordine nel nuovo array

ALBERI BINARI

L DI RICERCA
L ROSSI NERI (RBT)

RBT

modo : 6 campi

{ parent, left, right, key, data, color }

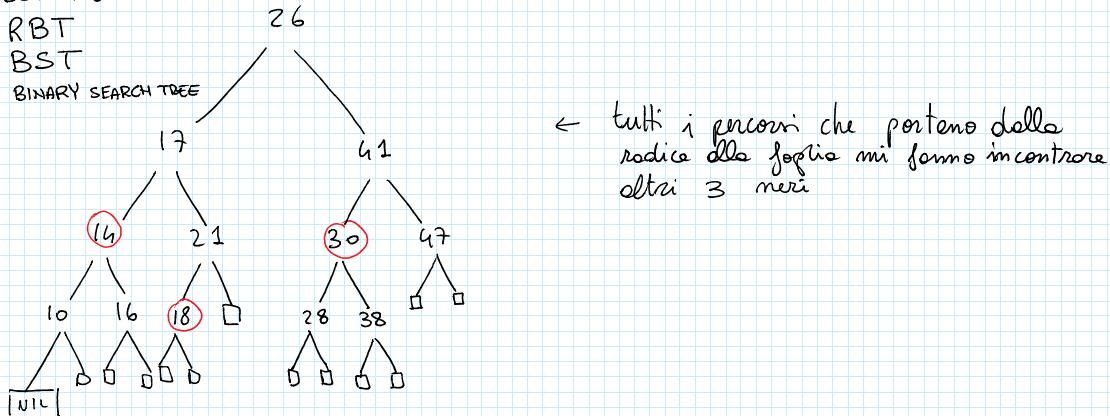
- ogni modo è rosso o nero
 - la radice è nera, ogni foglia è nera
 - se un modo è rosso, entrambi i figli sono neri
 - se in un modo, tutti i percorsi da n a un figlio suo discendente hanno lo stesso n° di modi neri

ESEMPIO

RBT

BST

BINARY SEARCH TREE



Bilanciamento

$$\beta(m) = | \text{altezza}(\text{left}(m)) - \text{altezza}(\text{right}(m)) |$$

T è bilanciato in altezza se per ogni nodo si ha $B(m) \leq 1$

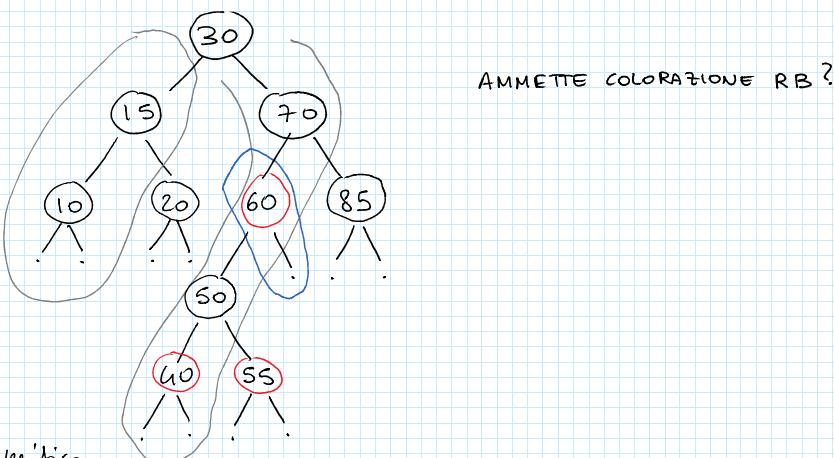
Gli alberi rossi meri non sono bilanciati.

Teoreme

L'altezza massima di un albero sono nero (RBT)

con n nodi interni è $\sim 2 \log(n+1)$

ESFRCA710



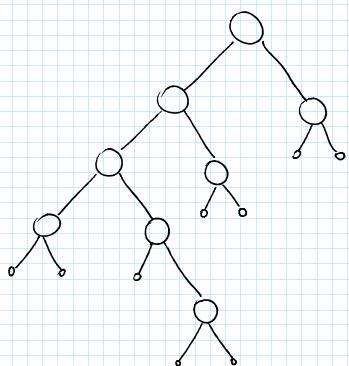
Verifico

- radice deve essere nera
 - profondità nera delle radici deve essere 2 o 3
 - la profondità nera delle radici, deve essere solo 3
 - 70 non può essere nero
 - devo mettere 2 rami tra 60 80 90

→ dal modo 60, si vede che la profondità è a sx 3 e a dx 1
 → quindi l'albero non è colorabile in modo da diventare uno nero
 perché non soddisfa tutte le condizioni di colorazione

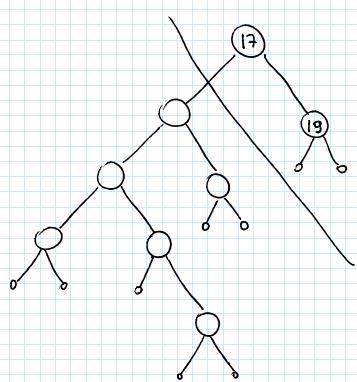
ESEMPIO

DATO

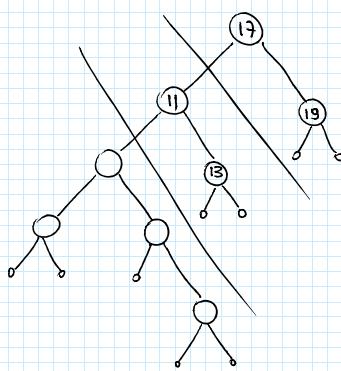


2) ASSEGNARE A O OGNI NODO
 LE CHIAVI S = {2, 3, 5, 7, 11, 13, 17, 19}
 IN MODO
 CHE RISULTI UN BST

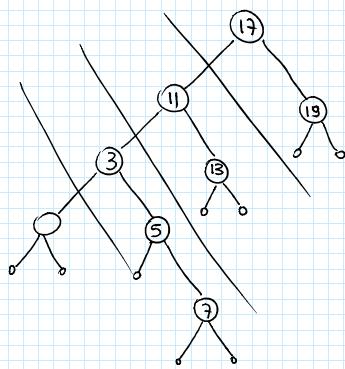
⇒



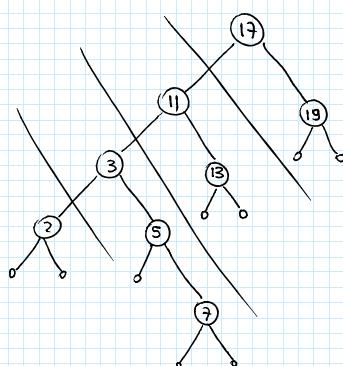
⇒



⇒



⇒



$$\begin{cases} h_{\min} = 2 \\ h_{\max} = 5 \end{cases}$$

2) T È COLORABILE RB?

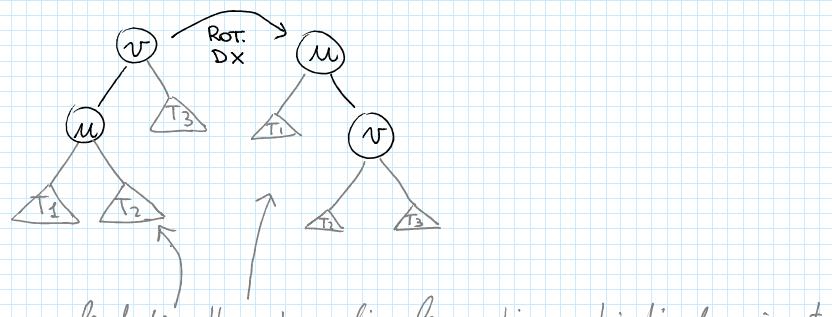
no troppo sbilanciato

($h_{\max} \leq 2h_{\min}$ per essere colorabile)

3) ESISTE UNA ROTAZIONE SU T CHE LO RENDA COLORABILE?

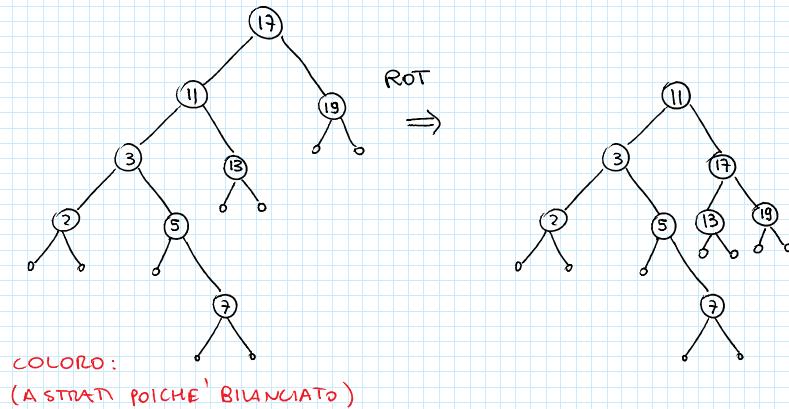
Sì, esiste sempre

NOTA: ROTAZIONI

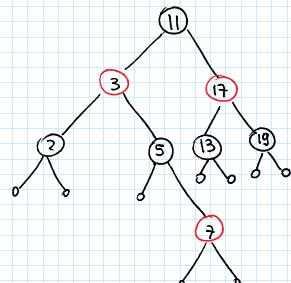


le foglie attecate agli elementi scomposti, le riporto dopo la rotazione nell'ordine in cui le ho trovate

allo stesso modo, se posto a sinistra avrà una rot. sx.



(ASTRAI POICHÉ' BILANCIATO)

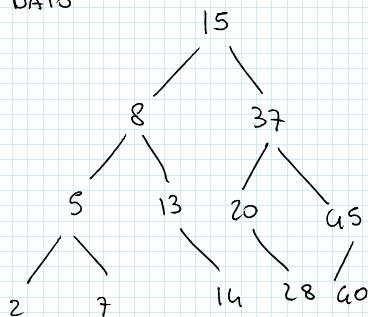


INSERIMENTO e RIMOZIONE

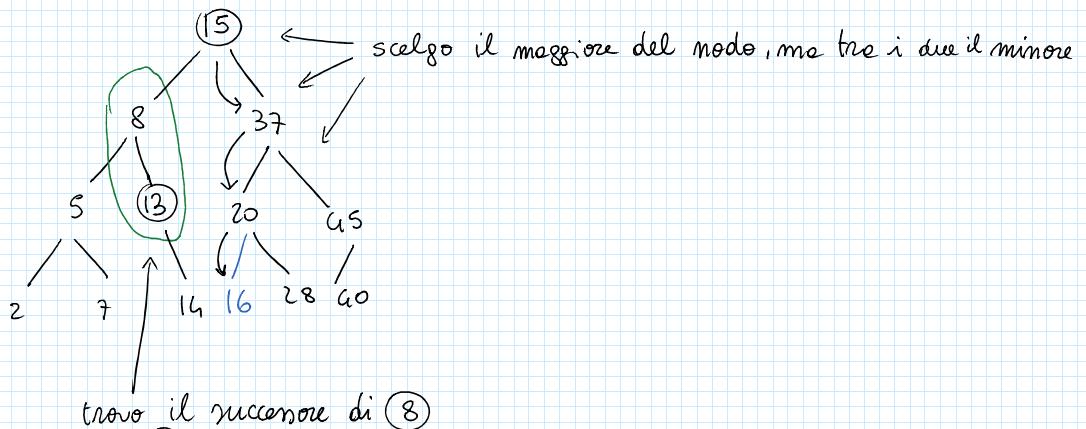
IN BST

1)

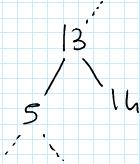
DATO



- inserire la chiusura 16 ed eliminare 8.

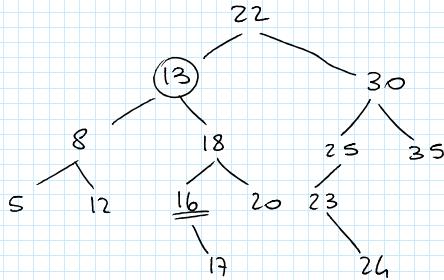


$\overset{\text{è}}{(3)}$
poiché successore è ottenuto a quello che devo rimuovere, basta che faccio scolare:



2)

DATO



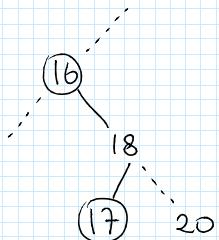
• Rimuovere 13

il successore di 13 è 16

1. devo eliminare 13

2. devo far scolare 16 al posto di 13

3. devo riottoccare i nodi



UNIONE e INTERSEZIONE

Siano $A, B \subseteq \mathbb{N}$ finiti

A, B rappresentati come BST bilanciati

Calcolare la complessità delle operazioni

$A \cup B \quad A \cap B \quad A \setminus B$

$$T_A, T_B \rightsquigarrow \begin{cases} |A| = m_A \\ |B| = m_B \\ h(T_A) \approx \log(m_A) \\ h(T_B) \approx \log(m_B) \end{cases}$$

UNIONE

COPIO T_A IN $T \rightarrow \Theta(m_A)$

VISITO $T_B \longrightarrow \Theta(m_B)$

CERCO se una chiave di T_B sta anche in $T_A \longrightarrow \Theta(\log m_A)$

se non c'è inserisco la chiave in $T \longrightarrow \Theta(\log(m_A + m_B))$

OSS. CASO PESSIMO SE $A \cap B = \emptyset$

Costo:

$$\Theta(m_A) + \Theta(m_B(\log m_A + \log(m_A + m_B)))$$

$$= \Theta(m_A + m_B \cdot \log(m_A + m_B))$$

quindi complexità è lineare rispetto a m_A
 $m \log m$ rispetto a m_B

INTERSEZIONE

vedere esercitazione successiva.

ESERCITAZIONE

martedì 27 gennaio 2015 — 08:43

1)

$A, B \subseteq N$ finiti

T_A, T_B BST bilanciati

$A \setminus B, A \setminus B, A \cap B$

a) $A \setminus B = C$

- copio T_A in T_C $\rightarrow \Theta(m_A)$
- visito T_B , elimino da T_C

le chiavi che comparevano anche in T_A \rightarrow

inizio $\Theta(m_B)$
ricerca $\Theta(\log m_A)$
cancello $\Theta(\log m_A)$

$\left. \right\} \Theta(m_A + m_B \log m_A)$

b) $A \cap B = C$

- inizializzo $T_C = \emptyset$ $\rightarrow \Theta(m)$

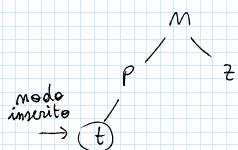
- visito T_B , se la chiave è in T_A
la inserisco in T_C

inizio $\Theta(m_B)$
ricerca $\Theta(\log m_A)$
inserimento $\Theta(\log m_A)$

$\left. \right\} \Theta(m_B \cdot \log m_A)$

2) INSERIMENTO RIMOZIONE IN RBT

[INSERIMENTO]

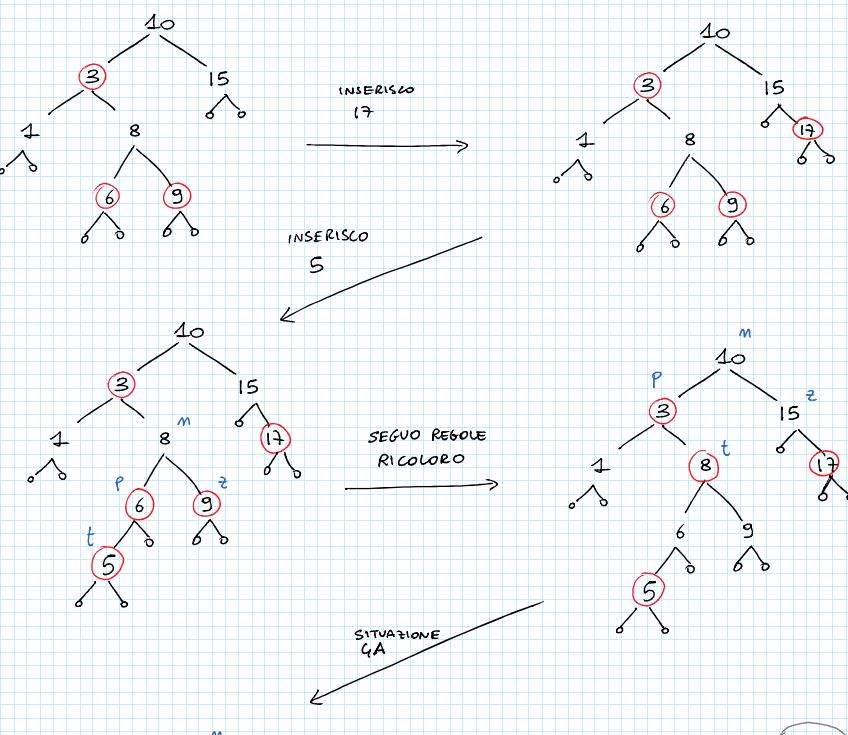


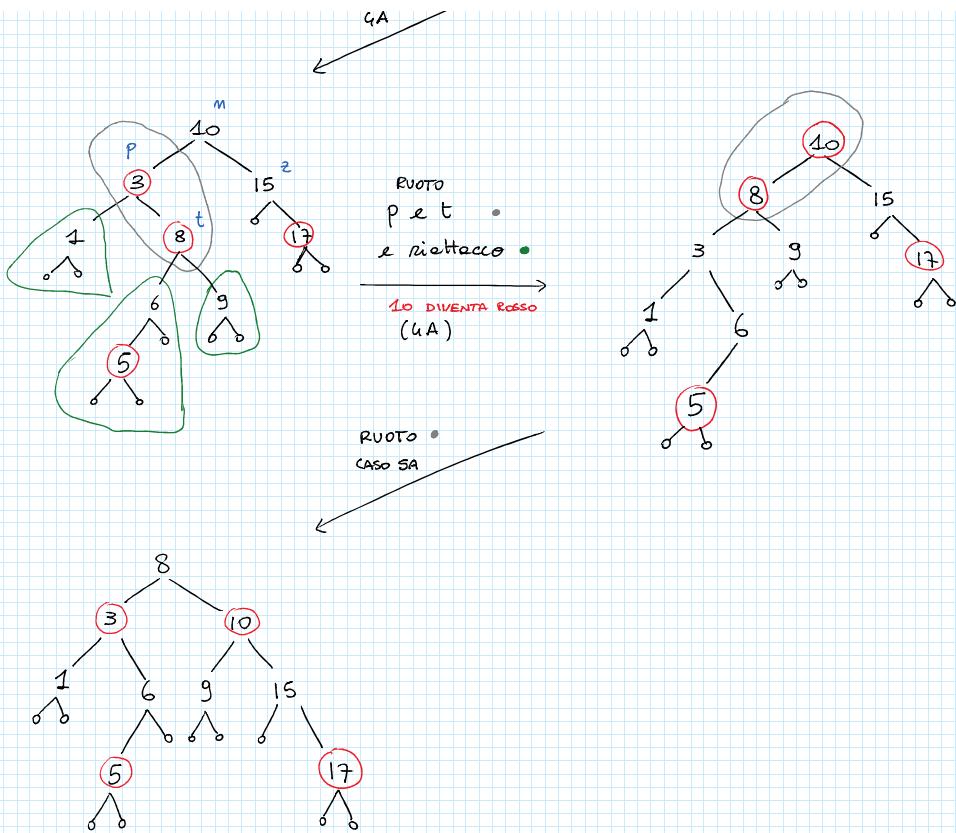
NOTA: complessità Θ proporzionale alle profondità dell'albero.
Se bilanciato è proporzionale al $\log(m)$

regole:

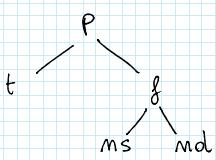
- il modo di inserire viene sempre colorato di rosso
- se t non ha un padre, t \rightarrow nero
- se il padre di t è nero \rightarrow ok
- se il padre è rosso, z è rosso \rightarrow p \rightarrow N, z \rightarrow N, m \rightarrow R
GA. se p \rightarrow R, m \rightarrow N, z \rightarrow N, p è figlio sx, t è figlio dx
 \rightarrow rotazione dx su p e p \rightarrow N, m \rightarrow R
GB. se p \rightarrow R, m \rightarrow N, z \rightarrow N, p è figlio dx, t è figlio sx
 \rightarrow rotazione sx su p e p \rightarrow N, m \rightarrow R
GA. se p \rightarrow R, m \rightarrow N, z \rightarrow N, p figlio sx, t figlio sx
 \rightarrow rotazione dx su m e p \rightarrow N, m \rightarrow R
GB. se p \rightarrow R, m \rightarrow N, z \rightarrow N, p figlio dx, t figlio dx
 \rightarrow rotazione sx su m e p \rightarrow N, m \rightarrow R

ESEMPIO





[RIMOZIONE]

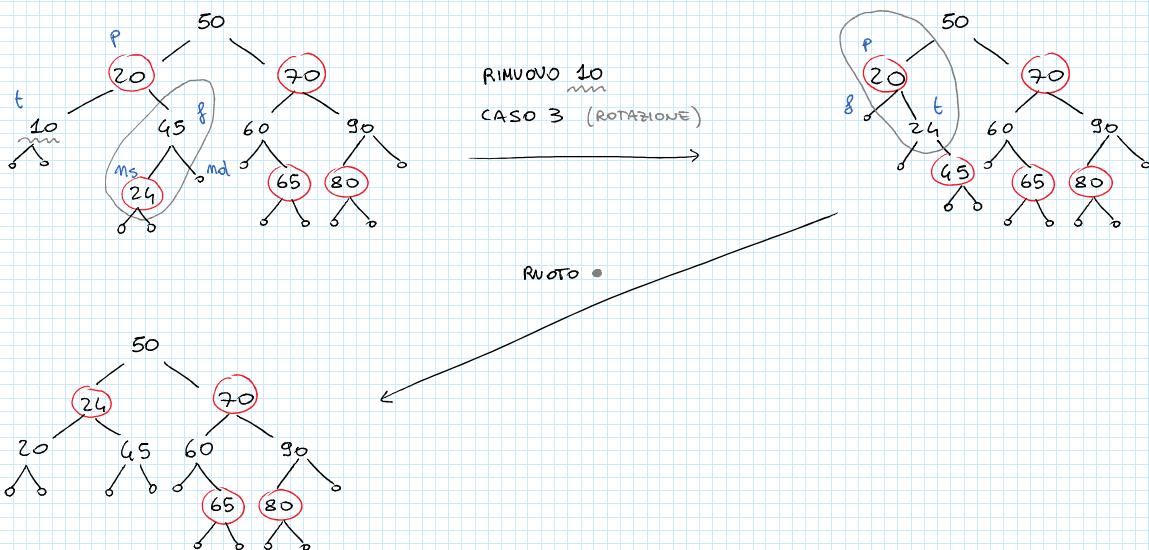


NOTA: complesità
→ vedi inserimento

regole:

1. se $p \rightarrow N$, $f \rightarrow R$, $ms, md \rightarrow N$
→ rotazione sx su p e $f \rightarrow N$, $p \rightarrow R$
2. se $f \rightarrow N$, $ms, md \rightarrow N$
→ allora $f \rightarrow R$ e se $p \rightarrow R$ allora $p \rightarrow N$
→ altrimenti se $p \rightarrow N$, $t = p$ e ricomincio
3. se $f \rightarrow N$, $ms \rightarrow R$, $md \rightarrow N$
→ rot. dx su f, $f \rightarrow R$, $ms \rightarrow N$
4. se $f \rightarrow N$, $md \rightarrow R$
→ rot. sx su p, $md \rightarrow N$, $p \rightarrow N$, $f \rightarrow$ vecchio colore di p

ESEMPIO



3) GRAFI

grafo : (V, E)

 | \
 vertici ordini (edge)

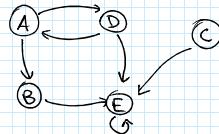
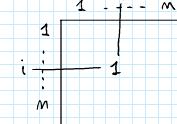
$$E \subseteq V \times V \quad 0 \leq |E| \leq V^2$$

grafi non orientati $\rightarrow E$ è simmetrica

RAPPRESENTAZIONE

- LISTE DI ADIACENZA

- MATRICE

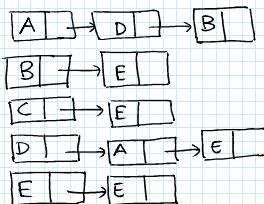


COMPLESSITÀ
 $\Theta(|V| + |E|)$

NOTA:

se ho pochi ordini \rightarrow liste

se ho molti ordini \rightarrow matrice

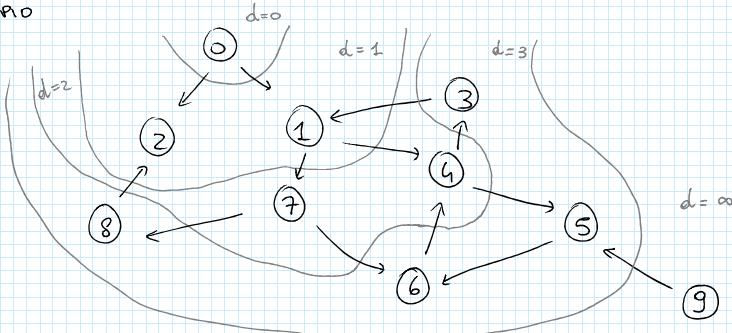


VISITA IN AMPIEZZA

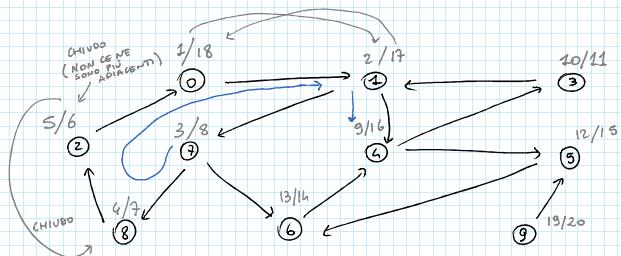
Visita in ampiezza, parto dalla sorgente, prima visita tutti i nodi a distanza 1, poi 2,

modo $\begin{cases} \text{bianco} & \rightarrow \text{da visitare} \\ \text{grigio} & \rightarrow \text{visitato, ma ha nodi adiacenti da visitare} \\ \text{nero} & \rightarrow \text{altrimenti} \end{cases}$

ESEMPIO



VISITA IN PROFONDITÀ

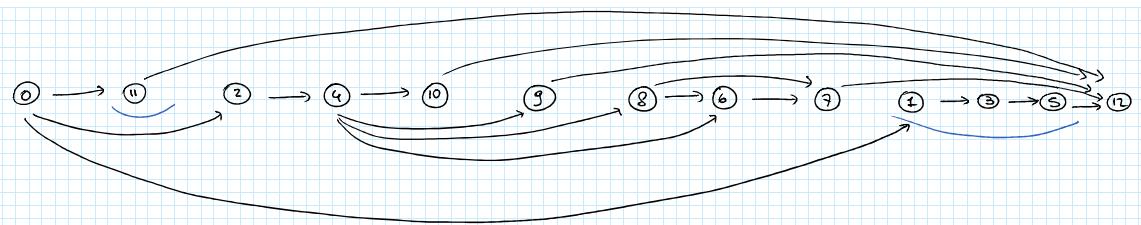


- Segnato in blu il procedimento della visita in profondità (cerco di tornare \uparrow e poi visito in b) (esempio mette 9 non in 6, ma risalgo fino a 1, e scalo a 4 non a 6 per il verso del suo arco)

- Segnato in grigio il procedimento delle visite iniziale e successiva chiusura.

ESEMPIO

NOTA



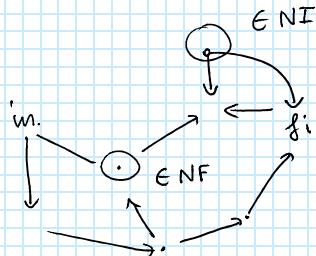
NOTA: gli ordini non si intersecano mai. utile nel caso di processi paralleli. (solo in caso acciuffo)

NOTA: i 2 — sono scomponibili

Tutorato

1) $G = (V, E)$ grafo orientato

inizio, fine $\in V$



- NI non raggiungibile da inizio
- NF " " " fine
- IF raggiungibile da inizio e da cui è raggiungibile fine

① Visita in profondità, dato $s \in V$ restituisce i vertici raggiungibili da s

Array indirizzato dai vertici di G t.c. $\forall u \in V \quad R[u] = \text{TRUE}$ se e solo se u è raggiung. da s

Raggiungibile (G, s)

INITIALIZ.	per ogni $u \in V(G)$ $R[u] = \text{FALSE}$ $\text{color}[u] = \text{white}$
------------	--

Visita ($G, s, R[s], \text{color}[s]$)

return r;

Visita (G, u, R, color)

$\text{color}[u] = \text{grey}$

$R[u] = \text{TRUE}$

per ogni $v \in \text{Adj}[u]$

if $\text{color}[v] = \text{white}$

then visita ($G, v, R[v], \text{color}[v]$)

$\text{color}[u] = \text{black}$.

$I = \text{raggiungibile}(G, \text{inizio}) \rightarrow \Theta(|V| + |E|)$

$$\begin{aligned}
 I &= \text{raggiungibile}(G, \text{inizio}) \rightarrow \Theta(|V| + |E|) \\
 NI &= V(G) \setminus I \rightarrow \Theta(n) \\
 G^T &= G^T \rightarrow \Theta(|V| + |E|) \\
 F &= \text{raggi} (G^T, \text{fine}) \rightarrow \Theta(|V| + |E|) \\
 NF &= V(G) \setminus F \rightarrow \Theta(|V|) \\
 IF &= I \cap F \rightarrow \Theta(|V|)
 \end{aligned}
 \quad \left. \right\} \Theta(|V| + |E|)$$

2)

$$T(m) = aT\left(\frac{m}{b}\right) + f(m) \quad \text{con } a, b > 1$$

MASTER THEOREM

$$\textcircled{1} \quad \text{se } f(m) = O\left(m^{\log_b a - \varepsilon}\right) \quad \varepsilon > 0 \\ \text{allora } T(m) = \Theta\left(m^{\log_b a}\right)$$

$$\textcircled{2} \quad \text{se } f(m) = \Theta\left(m^{\log_b a} (\log m)^k\right) \quad k \geq 0 \\ \text{allora } T(m) = \Theta\left(m^{\log_b a} (\log m)^{k+1}\right)$$

$$\textcircled{3} \quad \text{se } f(m) = \Omega\left(m^{\log_b a + \varepsilon}\right) \quad \varepsilon > 0 \\ \text{e } af\left(\frac{m}{b}\right) \leq c f(m) \quad \text{per } c < 1 \\ \text{allora } T(m) = \Theta(f(m))$$

ES.

$$T(m) = 2T\left(\frac{m}{2}\right) + m^3 \quad \text{es. } \varepsilon = \frac{1}{2} \\ \log_2 2 = 1 \quad f(m) = m^3 = \Omega\left(m^{3+\frac{1}{2}}\right) \quad (\text{TERZO caso})$$

$$2\left(\frac{m}{2}\right)^3 \leq c m^3 ?$$

$$\frac{2}{8} m^3 \leq \frac{1}{4} m^3$$

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

$$T(m) = \Theta(m^3)$$

ES.

$$T(m) = T\left(\frac{9}{10}m\right) + m^2$$

$$T(m) = T\left(\frac{9}{10}m\right) + m^2$$

$$\log_{\frac{9}{10}} 1 = 0 \quad f(m) = m^2 = m^{0+2} = \underline{\mathcal{R}}(m^{\log_{\frac{9}{10}} 1 + \epsilon})$$

$$\epsilon = \frac{5}{2}$$

$$\left(\frac{9m}{10}\right)^2 = \frac{81}{100}m^2 \quad c = \frac{81}{100} < 1$$

$$T(m) = \Theta(m^2)$$

ES.

$$T(m) = 16T\left(\frac{m}{4}\right) + m^2$$

$$\log_4 16 = 2$$

$$m^2 = \Theta\left(m^{\log_4 16} (\log m)^0\right) \quad (\text{caso 2})$$

$$T(m) = \Theta(m^2 \log m)$$

FS.

$$T(m) = 2T\left(\frac{m}{2}\right) + \sqrt{m}$$

$$\log_2 2 = \frac{1}{2}$$

$$f(m) = \sqrt{m} = \Theta\left(m^{\log_2 2} (\log m)^0\right) \quad (\text{caso 2})$$

$$T(m) = \Theta(\sqrt{m} \log m)$$

ES.

$$T(m) = mT(m-1)$$

noto che $mT(m-1) = m!$

$$T(m) = T(\sqrt{m}) + 1$$

$$x = \log_2 m \quad 2^x = m$$

$$T(2^x) = T(2^{\frac{x}{2}}) + 1$$

$$S(x) = T(2^x)$$

$$S(x) = S\left(\frac{x}{2}\right) + 1 =$$

$$\begin{aligned}
 &= \left(S\left(\frac{x}{2}\right) + 1 \right) + 1 = \\
 &= \left(\left(S\left(\frac{x}{2}\right) + 1 \right) + 1 \right) + 1 = \dots \\
 &= S\left(\frac{x}{2^i} + i\right) = S(0) + \log_2 x \\
 i &= \log_2 x \quad S(x) = \Theta(\log x)
 \end{aligned}$$

$$T(m) = S(x) = \Theta(\log x) = \Theta(\log \log m)$$

3) void useless (int m)

for ($i=1$; $i \leq m$; $i++$) | ripetuto m volte
aux(i);

void aux(int m)

if ($m == 1$)
return;
else if (odd(m))
aux($m-1$);
else
aux($\frac{m}{2}$)

$$m = 2^k \cdot h$$

$$\underbrace{m; \frac{m}{2}; \frac{m}{4}; \dots; \frac{m}{2^k}}_{\leftarrow \text{chiamate}} = h; h-1; \dots$$

$$2^{k+1}$$

- Se m potente di 2, aux di m costa $\log_2 m$
- caso peggiore $\rightarrow 2 \log_2(m)$ ricorsioni $\rightarrow \Theta(\log_2 m)$

4) $T(m) = 2T(m-4) + O(\log m)$

$$\log m = O(m^\varepsilon)$$

$$\begin{aligned}
 T(m) &= 2T(m-4) + \log m \\
 &= 2T(m-8) + \log(m-4) + \log m
 \end{aligned}$$

$$\begin{aligned}
&= 2T(m-16) + \log(m-8) + \log(m-4) + \log m \\
&= 2T(m-16) + \sum_{j=0}^{i-1} \log(m-16^j) \\
&= 2^{\frac{m}{16}} T(0) + \sum_{j=0}^{\frac{m}{16}-1} 2^j \log(m-16^j) \leq \log m \sum_{j=0}^{\frac{m}{16}-1} 2^j = \\
&= \log \frac{1-2^{\frac{m}{16}-1}}{1-2} = \log \left(2^{\frac{m}{16}-1} - 1 \right) \\
T(m) &= O \left(2^{\frac{m}{16}} \log m \right)
\end{aligned}$$

