22 SETTEMBRE

$$M = \langle Q, q_I, q_f, \Sigma, \sqcup, \delta \rangle$$

$$\delta : (Q \setminus q_f) \times \Sigma \to Q \times \Sigma \times \{-1, 0, +1\}$$

A q_f NON CI INTERESSA SAPERE LA MACCHINA COSA FA

PROGRAMMA CHE FA REVERSE DI UNA STRINGA

⊔	⊔	⊔	T	U	R	I	N	G	⊔	⊔	⊔

q_i

⊔	⊔	⊔	G	N	I	R	U	T	⊔	⊔	⊔

q_f

⊔	⊔	⊔	T	U	R	I	N	G	⊔	⊔	⊔

$q_i \Delta$

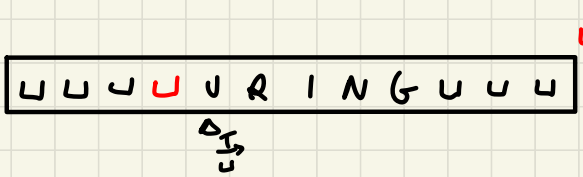
QUESTI SONO SIMBOLI A CASO

$$\delta = \left\{ \frac{\alpha}{\beta}, \frac{\alpha}{\beta} \mid \alpha, \beta \in \Sigma \right\} \cup \left\{ \Delta, \Delta \mid \alpha \in \Sigma \setminus \sqcup \right\}$$

STATI

$$q_I = \frac{\sqcup}{\sqcup} \qquad q_f = \frac{\sqcup}{\sqcup}$$

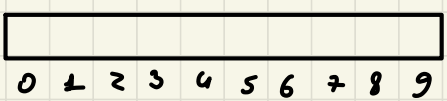
$$\delta\left(\frac{\sqcup}{\sqcup}, \alpha\right) = \left(\frac{\alpha}{\sqcup}, \sqcup, +1\right)$$



$$\delta\left(\underset{\sqcup}{\overset{q}{\rightarrow}}, \beta\right) = \left(\underset{\sqcup}{\overset{q}{\rightarrow}}, \beta, +1\right)$$

$Q \setminus \Sigma$	Σ		Q	Σ	$\{-1, 0, +1\}$
$\underset{\sqcup}{\overset{\sqcup}{\rightarrow}}$	A		$\underset{\sqcup}{\overset{A}{\rightarrow}}$	␣	+1
$\underset{\sqcup}{\overset{\sqcup}{\rightarrow}}$	B		$\underset{\sqcup}{\overset{B}{\rightarrow}}$	␣	+1
$\underset{\sqcup}{\overset{A}{\rightarrow}}$	A		$\underset{\sqcup}{\overset{A}{\rightarrow}}$	A	+1
$\underset{\sqcup}{\overset{B}{\rightarrow}}$	B		$\underset{\sqcup}{\overset{B}{\rightarrow}}$	B	+1

RAM (RANDOM ACCESS MACHINE)



OGNI CELLA HA UN INDICE

1) ASSIGNEMENT/ READING — A[:]
2) SEQUENZA — prg 1, prg 2
3) SELECTION — if cond. prog 1 else prog 2
4) ITERATION — while cond do prg

TEMPI

1) $T(\text{prog}) = O(1)$

2) $T(\text{prg}_1, \text{prg}_2) = T(\text{prg}_1) + T(\text{prg}_2)$

3) $T(\text{if} \dots) = T(\text{COND.}) + \text{MAX}\{T(\text{prg}_1), T(\text{prg}_2)\}$

4) WHILE COND. DO PROG.

ESEMPI:

PROBLEMA 1. CONTEGGIO DELLE COPPIE $\{(i,j) \in \mathbb{N} \times \mathbb{N} \mid 0 \leq i \leq j \leq m\}$

INPUT = $m \in \mathbb{N}$

OUTPUT = $\{(i,j) \in \mathbb{N} \times \mathbb{N} \mid 0 \leq i \leq j \leq m\}$

- ALGORITMO 1.

```
1) | COUNTER ← 0
2) | for i = 0 to m do
3) |     for j = 0 to m do
4) |         if i ≤ j then
5) |             COUNTER + 1
6) | RETURN COUNTER
```

23 SETTEMBRE

i t sono i costi TEMPO

- **ALGORITMO 1.**

```
1) | COUNTER ← 0
2) | for i := 0 to n do
3) |   | for j := 0 to n do
4) |   |   | if i ≤ j then
5) |   |   |   | COUNTER + 1
6) | RETURN COUNTER
```

t_1

$t_2 \rightarrow (n+1)\left[t_2 + (n+1)(t_3+t_4+t_5)\right]$

t_3

t_4 (t_3) $\}(t_4+t_5)$ $\}(n+1)(t_3+t_4+t_5)$

t_6

SE NON ESEGUE L'IF, FA SOLO t_4

DA t_1 A t_6 $\rightarrow t_1 + (n+1)t_2 + (n+1)^2(t_3+t_4+t_5) + t_6 =$

$= (t_3+t_4+t_5)n^2 + (2(t_3+t_4+t_5)+t_2)n + (t_1+t_2+t_3+t_4+t_5+t_6) =$

$= c_2 n^2 + c_1 n + c_0$

Algoritmo 2

1) COUNTER ← 0

2) for i = 0 to m do

3) | for j = i to m $\begin{pmatrix} t_3 \\ t_5 \end{pmatrix}$ $(m-i+1)[t_3+t_5]$ $\sum_{i=0}^{m} (t_2+(m-i+1)[t_3+t_5])$

5) | COUNTER ++

6) RETURN COUNTER

$$= t_1 + \sum_{i=0}^{m} \left(t_2 + (m-i+1)\left[t_3+t_5\right]\right) + t_6 =$$

$$= t_1 + t_6 + \sum_{i=0}^{m} \left(t_2 + (m+1)\left[t_3+t_5\right] - i\left[t_3+t_5\right]\right)$$

$$= t_1 + t_6 + \sum_{i=0}^{m} \left[t_2 + (m+1)(t_3+t_5)\right] - \sum_{i=0}^{m} i\left[t_3+t_5\right] =$$

$$= t_1 + t_6 + (m+1)\left[t_2 + (m+1)(t_3+t_5)\right] - (t_3+t_5)\sum_{i=0}^{m} i =$$

$$\sum_{i=0}^{m} i = \frac{m(m+1)}{2}$$

$$= t_1 + t_6 + (m+1)(m+1)(t_3+t_5)] - \frac{m(m+1)}{2}(t_3+t_5)$$

$$= (m+1)^2(t_3+t_5) - \frac{m(m+1)}{2}(t_3+t_5) + (m+1)t_2 + t_1 + t_6 =$$

$$= (t_3+t_5)(m+1)\left[m+1-\frac{m}{2}\right] + mt_2 + (t_2+t_1+t_6) =$$

$$= \left[\frac{m^2}{2} + m + \frac{m}{2} + 1\right](t_3+t_5) + mt_2 + (t_2+t_1+t_6) =$$

$$= \frac{(t_3+t_5)}{2} n^2 + \left[\frac{3}{2}(t_3+t_5)+t_2\right] n + (t_2+t_1+t_6+t_3+t_5)$$

Algoritmo 3

1) COUNTER ← 0 — t_1
2) for $i = 0$ to n do — t_2 ⎫ $(n+1)(t_7+t_2)$
7) ⎪ COUNTER ← COUNTER + $(n - i + i)$ — t_7 ⎭
6) return COUNTER — t_6

$$(t_2+t_7)n + (t_1+t_6+t_7+t_2)$$

SOLUZIONE EASY

$$\sum_{i=0}^{n} (n-i+1) = (n+1)\sum_{i=0}^{n} 1 - \sum_{i=0}^{n} i$$

COUNTER ← $\frac{n(n+1)}{2}$

27 SETTEMBRE

$$F(m) = \begin{cases} 1, & \text{se } m \in \{0,1\} \\ F(m-1) + F(m-2), & \text{ALTRIMENTI} \end{cases}$$

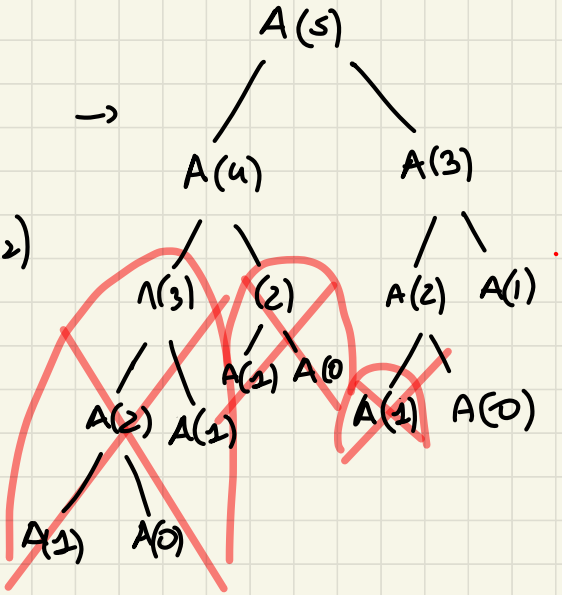
↓

FIBONACCI

ALGORITMO RICORSIVO A(m)

1) if $m \in \{0,1\}$
2) | RETURN 1
ELSE
3) | RETURN $A(m-1) + A(m-2)$

→



$$F(m) = \frac{\varphi^m + (\varphi - 1)^m}{\sqrt{5}}$$

$$\varphi = \frac{1+\sqrt{5}}{2} = 1,6.. \quad \rightarrow \text{SEZIONE AUREA}$$

ALGORITMO A2(n)

```
1) a, b ← 1
2) WHILE n ≥ 2 do
3)     a ← a+b
4)     b ← a-b
5)     n ← n-1
6) RETURN a
```

NOTAZIONE ASINTOTICA

$f(n) = o(g(n))$ → "o" PICCOLO

$f(n) = \omega(g(n))$ → OMEGA PICCOLO

SONO TUTTE FUNZIONI $f: N \to N$ → TEMPO IMPIEGATO

- $\exists n_0 \in N : \forall n \geq n_0 \quad f(n) \geq 0$

FUNZIONI NON DECRESCENTI

- $\exists n_0 \in N : \forall n \geq n_0 \quad f(n) \leq f(n+1)$

  K $n_0 \leq n_1 \leq n_2 \quad f(n_1) \leq f(n_2)$

$$f(n) \notin o(f(n))$$

DEFINIZIONE DI "o" PICCOLO E ω OMEGA PICCOLO

- $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \rightarrow$ o PICCOLO

- $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \rightarrow \omega$ PICCOLO

ALTERNATIVA:

- $\forall c \in R^+ : \exists n_0 \in N \quad \forall n \geq n_0 \quad f(n) \leq c\, g(n) \rightarrow$ "o"

  ↓ VALORE DI RISCALAMENTO

Definizione di Limite $\rightarrow \lim_{n \to \infty} h(n) = L \quad \forall \varepsilon > 0 \ \exists n_0 \in N \ \forall n \geq n_0$ ($\varepsilon \in R^+$)

$$|h(n) - L| < \varepsilon$$

$$\left| \frac{f(n)}{g(n)} - 0 \right| < \varepsilon \rightarrow \frac{f_n}{g_n} < \varepsilon$$

↓ IL RAPPORTO È PERFORZA POSITIVO

$$f(n) < \varepsilon \cdot g(n)$$

- $\forall M \in \mathbb{R}^+ \ \exists m_0 \in \mathbb{N} : \forall n \geq m_0 \ h(n) \geq M \ \rightarrow f(n) \geq c \cdot g(n)$

$\lim_{n \to \infty} h(n) = +\infty$

$\frac{f(n)}{g(n)} \geq M \rightarrow f(n) \geq M\, g(n)$

INDEBOLIMENTO DI QUESTE NOZIONI:

1) $f(n) = O(g(n))$ → O GRANDE (PIÙ DEBOLE DI "O")

2) $f(n) = \Omega(g(n))$ → OMEGA GRANDE

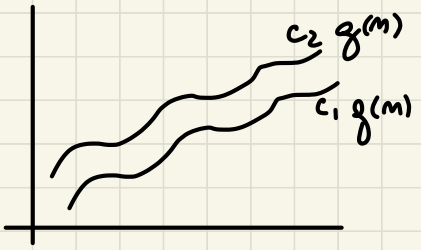
3) $f(n) = \Theta(g(n))$ → TETA GRANDE

1) $f(n) = O(g(n)) \rightarrow \exists c \in \mathbb{R}^+ \ \exists m_0 \in \mathbb{N} \ \forall n \geq m_0 \ f(n) \leq c \cdot g(n)$

2) $f(n) = \Omega(g(n)) \rightarrow \exists c \in \mathbb{R}^+ \ \exists m_0 \in \mathbb{N} \ \forall n \geq m_0 \ f(n) \geq c \cdot g(n)$

3) $f(n) = \Theta(g(n)) \rightarrow \Theta(g(n)) \triangleq O(g(n)) \cap \Omega(g(n)) : \exists c_1, c_2 \in \mathbb{R}^+$

$\exists m_0 \in \mathbb{N} \ \forall n \geq m_0 \ c_1 g(n) \leq f(n) \leq c_2 g(n)$



$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$

$f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$

$f(n) = \Theta(g(n))$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = L \in \mathbb{R}^+ \Rightarrow f(n) = \Theta(g(n))$$
$$> 0$$

Dim.

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = L \in \mathbb{R}^+ \quad \forall \varepsilon \in \mathbb{R}^+ \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0$$

$\forall n \geq n_0 \ g(n) > 0$

$$\left| \frac{f(n)}{g(n)} - L \right| \leq \varepsilon$$

$$\Updownarrow$$

$$-\varepsilon \leq \frac{f(n)}{g(n)} - L \leq \varepsilon$$

/

$$L - \varepsilon \leq \frac{f(n)}{g(n)} \leq L + \varepsilon$$

$$(L - \varepsilon)\, g(n) \leq f(n) \leq (L + \varepsilon)\, g(n)$$

$L - \varepsilon > 0$ $\quad$ $L + \varepsilon > 0$

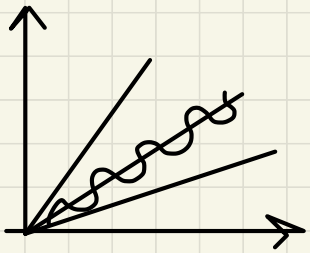
$\varepsilon < L$

$$c_1\, g(n) \leq f(n) \leq c_2\, g(n)$$

$\exists c_1, c_2 \in \mathbb{R}^+$ $\quad$ $\forall n \geq n_0$

$\exists n_0 \in \mathbb{N}$

29 SETTEMBRE



$x + \cos = O(x)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = L \in \mathbb{R}^+ \quad \begin{matrix} < \infty \\ > 0 \end{matrix} \Rightarrow f(n) = \Theta(g(n))$$

$\not\Leftarrow$

$$f(n) = 2n^2 + 3n \log n + 5\frac{n^2}{\log n} + 3$$

$g(n) = n^2$

= ASINTOTICAMENTE n^2

$$1 \overset{\text{piccolo}}{\ll\ll} \log(\log n) \ll\ll \log n \ll n \log n \ll n^2 \ldots$$

↳ o piccolo di n^2

$$\frac{f(n)}{g(n)}$$

$\ln(e^2 \cdot n) = \ln(e^2) + \ln n$

$2 \ln e + \ln n$

$2 + \ln n$

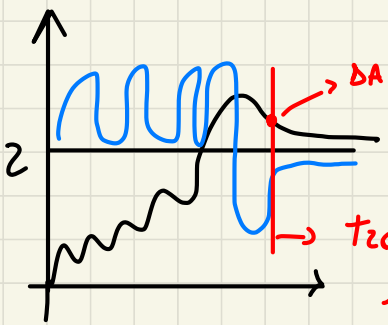
$$\frac{f(n)}{g(n)} = \frac{2n^2 + 3n\log n + 5\frac{n^2}{\log n + 3}}{n^2} =$$

$$= 2 + \underbrace{3\frac{\log n}{n}}_{\to 0} + \underbrace{5\frac{1}{\log n}}_{\to 0} + \underbrace{\frac{3}{n^2}}_{\to 0} =$$

$$\lim_{n\to\infty} h(n) = 2$$

↳ questa è una costante, ma non sappiamo se è c_1 o c_2

↳ crescente o decrescente



1) STUDIO DELLA MONOTONIA di $h(n)$

COSE DA NON FARE

NON USARE LA REGOLETTA → $\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$

$f(x)[g(x)]^{-1}$ ← MA USA

PROPRIETÀ:

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

2) $f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n))$

$g(n)$ O GRANDE O, $g(n) \leq \frac{1}{c_1} f(n)$

$\exists c_1, c_2 \in \mathbb{R}^+$

$\exists n_0 \in \mathbb{N}$

$\forall n \geq n_0 \quad c_2 g(n) \leq f(n) \leq c_2 g(n)$

$\frac{1}{c_2} f(n) \leq g(n)$

$\exists c_1', c_2' \in \mathbb{R}^+$

$\exists n_0 \in \mathbb{N}$

$c_1' f(n) \leq g(n) \leq c_2' f(n)$

$\frac{1}{c_2} \qquad \frac{1}{c_1}$

3) $f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \overset{?}{\Longrightarrow} f(n) = \Theta(h(n))$

$\exists c_1, c_2 \in \mathbb{R}^+$ $\quad c_1 g(n) \leq f(n) \leq c_2 g(n) \wedge$ $\quad \exists c_1', c_2' \in \mathbb{R}^+$ $\quad c_1' h(n) \leq f(n) \leq c_2' h(n)$

$\exists n_0 \in \mathbb{N}$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\exists n_0' \in \mathbb{N}$

$\exists c_1'', c_2'' \in \mathbb{R}^+$

$\exists \bar{n}_0 \in \mathbb{N}$

$c_1'' h(n) \leq f(n) \leq c_2'' h(n)$

$c_2 g(n) \leq c_2 c' h(n)$

HO MOLTIPLICATO ENTRAMBI PER c_2

$\uparrow$

$\max \{n_0, \bar{n}_0\}$

quindi $f(n) \leq c_2 g(n) \leq c_2 c' h(n) = c''$

TRANSITIVITÀ

NON TUTTE LE EQUIVALENZE SONO CONGRUENZE

SOLO RELAZIONI D'ORDINE E NON DI EQUIVALENZA

1) $f(n) = O/\Omega(f(n))$

2) $f(n) = O/\Omega(g(n)) \wedge g(n) = O/\Omega(h(n))$

---

1) $O/\Omega/\Theta(k \cdot f(n)) = O/\Omega/\Theta(f(n))$

$c_1 k f(n) \leq f(n) \leq c_2'' k f(n)$ ($\frac{1}{k}$, $1 \to k > 1$)

$c_1 = c_2 = \frac{1}{k}$

2) $O(f(n) + g(n)) = O(\max\{f(n), g(n)\})$ $\to k = 2$

$\Omega(f(n) + g(n)) = \Omega(\min\{f(n), g(n)\})$

3) $f(n) = O(h(n))$
$g(n) = O(h(n))$

$f(n) + g(n) = O(h(n))$

$f_1(n) = O(g_1(n))$
$f_2(n) = O(g_2(n))$

$\Rightarrow f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

$p(n) \rightarrow$ polinomio di grado d

$p(n) = \Theta(n^d)$

$f(n) = \Theta(n^k) \quad g(n) = \Theta(h(n))$

$f(g(n)) = \Theta((h(n))^k)$

↖ CONGRUENZA UGUAGLIANZA PERCHÉ n^k È MONOTONA CRESCENTE ALTRIMENTI NON FUNZIONEREBBE

30 SETEMBRE

1) Ip: $f(n) = \Theta(g(n))$

2) $\overset{?}{\Rightarrow} (f(n))^k = \Theta((g(n))^k), \forall k \in \mathbb{N}$

3) $\overset{?}{\Rightarrow} \log(f(n)) = \Theta(\log(g(n)))$

4) $\overset{?}{\Rightarrow} 2^{f(n)} = \Theta(2^{g(n)})$

BASE DI VERIFICA PER TUTTE

Dim.

1) ✓ $\exists c_1, c_2 \in \mathbb{R}^+ \; \exists n_0 \in \mathbb{N} \; \forall n \geq n_0 \quad c_1 g(n) \leq f(n) \leq c_2 g(n)$

2) ✓ $\exists c_1', c_2' \; \exists n_0 \in \mathbb{N}$ (= c_1^k, c_2^k ; = $n_0 \in \mathbb{N}$)

$(c_1 g(n))^k \leq (f(n))^k \leq (c_2 g(n))^k$

$c_1^k \cdot (g(n))^k \leq (f(n))^k \leq c_2^k \cdot (g(n))^k$

3) ✓ $\log(c_1 g(n)) \leq \log(f(n)) \leq \log(c_2 g(n))$

$\log(c_1) + \log(g(n)) \leq \log(f(n)) \leq \log(c_2) + \log(g(n))$

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

$\log(g(n)) = \Theta(\log(g(n)))$ ← CHE CI BASTA

$\log(c_2) = O(\log(g(n)))$ → FUNZIONA PER LA "O"

MA LA SOLUZIONE MIGLIORE È FARE IL LIMITE:

$$\lim_{n\to\infty} \frac{\log(c_2) + \log(g(n))}{\log(g(n))} = 1$$

4)

$$2^{c_1 \cdot g(n)} = 2^{f(n)} \leq 2^{c_2 g(n)}$$

$$(2^{c_1})^{g(n)} \leq 2^{f(n)} \leq (2^{c_2})^{g(n)}$$

CI VUOLE UN CONTRO-ESEMPIO

$f(x)$ è O PICCOLO $\to o(x)$

$$x = \text{H}(x + f(x))$$

$$c_1 x \leq x + f(x) \leq c_2 x$$

$a^x \ll b^x \to$ se $a < b$ e $1 \leq a < b$

ESERCIZI:

1) MAX SUB SEQUENCE SUM

INPUT: $n \in \mathbb{N}$, $\vec{a} \in \mathbb{Z}^n$ $\quad \vec{a} = (a_0, a_1 \dots, a_{n-1})$

OUTPUT: SOTTOSEQ. LA CUI SOMMA È MASSIMA

$$0 \le i \le j < n : S_{i,j} = \max \left\{ S_{i,j} \mid 0 \le i' \le j' < n \right\}$$

$$\sum_{k=i}^{j} \vec{a}_k$$

Es. $\vec{a} = (-2, 1, -3, 4, -1, 2, 1, -5, 4)$ (posizioni 1 2 3 4 5 6 7 8 9)

$S_{i,j} = (3,6) = 6$

ALGORITMO: $A_1(\vec{a}, n)$

ADESSO VEDIAMO IL COSTO:

```
1) MAX-SUM ← 0                                              H1(1)
2) for i:=0 to n-1 do
3)     for j:=i to n-1 do                          1 ] H1
4)         SUM ← 0                                 ] H1
5)         for k=i to j do                         1 ] H1   (1 esecuzione)
6)             SUM = SUM + a_k                     1 ] H1   ] H(j-i)
7)         if MAX-SUM < SUM THEN                   1 ] H1   ] Σ_{k=i}^{j} H1 = (j-i+1)H1 : H(j-i)
8)             MAX-SUM ← SUM                                ] H(j-1)
9)     RETURN MAX SUM                                       H1(9)
```

4 OTTOBRE

CI SONO ESERCIZI PER CASA

ALGORITMO $A_1(\vec{a}, m)$

```
1) maxSUM ← 0
2) for i = 0 to m-1 do
3)     for j = i to m-1 do
4)         sum ← 0                                   Θ1
5)         for k = i to j do                         Θ(1)  (j - i + 1) Θ2 = Θ(j-i)
6)             sum ← sum + a_k
7)         maxsum ← max {maxsum, sum}                Θ1
8) return maxsum
```

$\sum_{j=i}^{m-1} \Theta(j-i)$ $\sum_{i=0}^{m-1}\sum_{j=i}^{m-1} \Theta(j-i)$

$\sum_{i=0}^{m-1}\sum_{j=i}^{m-1} \Theta(j-i) \leq \Theta\left(\sum_{i=0}^{m-1}\sum_{j=i}^{m-1} j-i\right)$

FARE QUESTO CALCOLO CI PERMETTE DI CAPIRE L'ANDAMENTO ASINTOTICO

QUESTO PUÒ PASSARE DAVANTI, QUINDI SONO UGUALI

SVOLGIMENTO:

$\sum_{j=0}^{m-1} j - \sum_{j=0}^{i-1} j$ $\quad i\cdot(m-1-i+1)$

$$\sum_{i=0}^{m-1}\sum_{j=i}^{m-1}(j-i) = \sum_{i=0}^{m-1}\left[\sum_{j=i}^{m-1} j - i\sum_{j=i}^{m-1} 1\right] = \sum_{i=0}^{m-1}\left[\frac{m(m-1)}{2} - \frac{i(i-1)}{2} - i(m-i)\right] =$$

$$= \sum_{i=0}^{m-1} \left[\frac{m(m-1)}{2}\right] = \sum_{i=0}^{m-1} \frac{i(i-1+2m-2i)}{2} = \left[\frac{m(m-1)}{2}\right] \sum_{i=0}^{m-1} 1 = \frac{m^2(m-1)}{2} = \Theta(m^3)$$

$\sum_{i=0}^{m-1} 1 = m$

DOPO PASSO SI ABBASSA AD m^2

$$= \frac{m^2(m-1)}{2} - \sum_{i=0}^{m-1} \frac{2mi - i(i-1)}{2}$$

$\frac{m^2(m-1)}{2}$

$$\sum_{i=0}^{m-1} m \cdot i - \frac{1}{2} \sum_{i=0}^{m-1} i(i+1) = m \sum_{i=0}^{m-1} i - \frac{1}{2} \sum_{i=0}^{m-1} i(i+1)$$

$$\frac{1}{2} \sum_{i=0}^{m-1} i(i+1) = \frac{1}{2}\left[\sum_{i=0}^{m-1} i^2 + \sum_{i=0}^{m-1} i\right] =$$

$$= \frac{1}{2}\left[\sum_{i=0}^{m-1} i^2 + \frac{m(m-1)}{2}\right] =$$

$$\sum_{i=0}^{m-1} i^2 = \sum_{i=0}^{m-1} i \cdot i =$$

$$= \sum_{i=0}^{m-1} i \cdot \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{m-1} \sum_{j=1}^{i-1} i =$$

QUESTO DIVENTA 1

UNICA SOMMATORIA CON 2 INDICI

$$\sum_{i=1}^{m-1} \sum_{j=1}^{i-1} i \qquad \sum_{1 \le j \le i \le m-1} i = \sum_{j=1}^{m-1} \sum_{i=j}^{m-1} i =$$

$z = m-1$

$$= \sum_{j=1}^{m-1} \left[\sum_{i=0}^{m-1} i - \sum_{i=0}^{j-1} i\right] =$$

$$= \sum_{j=1}^{z} \left[\frac{z(z+1)}{2} - \frac{j(j-1)}{2}\right] = \frac{z(z+1)(z-1)}{2} - \sum_{j=1}^{z} \frac{j(j-1)}{2} =$$

$z^2 - 1$

$\sum_{i=0}^{z} i^2$ ↑ SOMMA

$$= T(z) - \frac{1}{2} \sum_{j=1}^{z} j^2 + \frac{1}{2} \sum_{j=1}^{z} j$$

$T(z)$ → H_z

$\sum_{j=1}^{z} j$

$$S_z = H(z) - \frac{1}{2} S_z \Rightarrow \frac{3}{2} S_z = H(z) \Rightarrow S_z = \frac{2}{3} H(z) \Rightarrow S_z = \frac{1}{6} m(m+1)(2m+1)$$

$$\sum_{k=i}^{j+1} a_k = \sum_{k=i}^{j} a_k + a_{j+1}$$

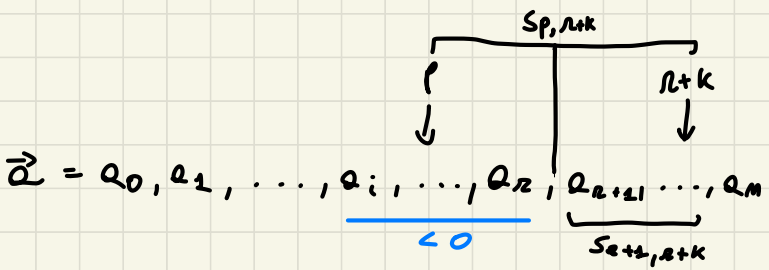
Algoritmo $A_2(\vec{a}, n)$

1) max sum $\leftarrow 0$
2) for $i := 0$ to $n-1$ do
2.5) sum $\leftarrow 0$
3) for $j = i$ to $n-1$ do
4) sum $\leftarrow$ sum $+ a_j$
5) max sum $\leftarrow \max\{$max sum, sum$\}$
6) return max sum

$H(1)$ $H(2)$ $H(n-1-j+1)$ $\sum_{i=0}^{n-1} H(n-i)$

$H\left(\sum_{i=0}^{n-1} n-i\right)$

$\sum_{i=0}^{n-1} n-i = \sum_{j=1}^{n} j$



SEMPLICE PROPOSIZIONE

$\exists p$ (a) $\forall i \le j < n$, $s_{ij} \ge 0$

(b) $s_{in} < 0$

$s_{ij} = \sum_{k=i}^{j} \vec{a}_k$

$\text{Thi}(a) \Rightarrow \forall i \le p \le j < n$, $s_{pj} < s_{ij}$

$(a) + (b) \Rightarrow \forall i \le p < n$, $\forall k \in \mathbb{N}$ $\quad s_{p,n+k} < s_{n+1,n+k}$

$$s = \underbrace{s_{i,p-1}}_{\ge 0} + s_{pj}$$

$$s_{ij} - s_{i,p-1} = s_{pj}$$

$$\Downarrow$$

$$s_{ij} \ge s_{pj}$$

$$s_{p,n} + s_{n+1,n+k}$$

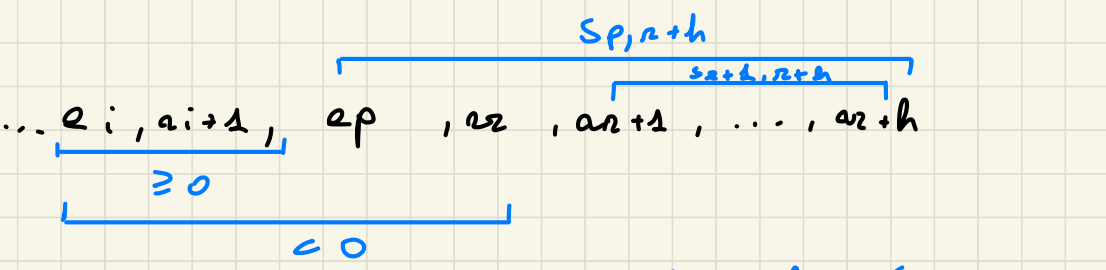
$$\le$$

$$s_{in} < 0$$

6 OTTOBRE

Algoritmo $A_3(\vec{a}, n)$

```
1) | max sum , sum ← 0                                   H1
2) | for J = 0 to n-1 do                                 H(n)
3) |   sum ← sum + a_J                                   H1
4) |   if sum < 0 then sum ← 0                           H1
5) |   max sum ← max { max sum , sum }                   H1
6) | RETURN max sum                                      H1
```



$s_{p,r+h} < s_{r+1,r+h}$

2° PARTE DEL CORSO / ALGORITMI DI ORDINAMENTO

PROBLEMA (ORDINAMENTO SEQUENZE (SORTING))

$(D, \leq)$

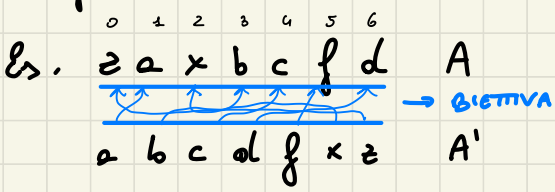
D: INSIEME DI DATI — $\leq$: RELAZIONE D'ORDINE TOTALE

INPUT: ARRAY/SEQUENZA $A \in D^*$ (SEQ. ARBITRARIA DI DATI), $n = |A| \in \mathbb{N}$ (A NON HA DUPLICATI)

$\forall i, j \in [0, n), i \neq j \Rightarrow A[i] \neq A[j]$

OUTPUT: a) $A' \in D^*$, $|A'| = n$

b) $\exists f : [0, n) \to [0, n)$ BIETTIVA $\forall i \in [0, n): A'[i] = A[f(i)]$



PERMUTAZIONE

↓

A' PERMUTAZIONE DI A

↓

STESSI ELEMENTI RIARRANGIATI

c) $\forall i \in [0, n-2] \Rightarrow A'[i] \leq A'[i+1]$

c') $\forall i, j \in [0, n), i < j \Rightarrow A'[i] \leq A'[j]$

INSERTION SORT

Algoritmo (A, m)

RICORSIVO

```
1) if m > 1 then
2)     INSERTION SORT (A, m-1)
3)     X ← A [m-1]        (X: pivot)
4)     J ← m-2
5)     WHILE ( J ≥ 0 ∧ A[J] > X )
6)         A[J+1] ← A[J]
7)         J - -
8)     A[J+1] ← X
```

Algoritmo 2 (A, m)


```
1) for i = 1 to m-1 do
2)     X ← A [m-1]        (X: pivot)
3)     J ← i - 1
4)     WHILE ( J ≥ 0 ∧ A[J] > X )
5)         A[J+1] ← A[J]
6)         J - -
7)     A[J+1] ← X
```

7 OTTOBRE

STIMA ASINTOTICA PER:

CASO PEGGIORE (WORST CASE) $\Theta(n^2)$

CASO MIGLIORE (BEST CASE) $\Theta(n)$

CASO MEDIO (AVERAGE CASE) $\Theta(n^2)$

Algoritmo InsertionSort (A, n)

1) for $i = 1$ to $n-1$ do
2) INSERT (A, i) $\quad T_{INS.}(A, i)$

$$\left.\sum_{i=1}^{n}\right] T_{INS.}(A, i) \leq n \cdot \Theta(n) = \Theta(n^2)$$

$\Theta(3 + 2 \cdot k_i^A) = \Theta(k_i^A)$ → STIMA ASINTOTICA SENZA IL WHILE

$$\sum_{i=1}^{n} T_{INS}(A, i) = \sum_{i=1}^{n} \Theta(k_i^A) = \Theta\left(\sum_{i=1}^{n} k_i^A\right)$$

$k_i^A \leq i$

CASO PEGGIORE

0	1	2	3	4	5	6	7	8	9	10
10	9	8	7	6	5	4	3	2	1	0
10	10									

→ 1 WHILE

9	10	8	7	6	5	4	3	2	1	0
9	10	10								
9	9	10								
8	9	10								

→ 3 WHILE

$K_i^A = \sum_{i=1}^{m} i \quad \frac{m(m+1)}{2} = \Theta(m^2)$

$K_i^A \; 1 = \Theta m \rightarrow$ <u>CASO MIGLIORE</u>

Es.

	0	1	2	3	4	5
A =	8	2	4	9	3	6

$$\left| \{(i,j) \in [0,m) \mid i < j \wedge A[i] > A[j]\} \right|$$

1	2	8				
	2	8	4			
1	2	4	8			
0	2	4	8	9		
	2	4	8	9	3	
3	2	3	4	8	9	
	2	3	4	8	9	6
2	2	3	4	6	8	9

↓

N. SCAMBI = 7 = WHILE ESEGUITO 7 VOLTE

$$T_{INS}(A,m) = \Theta(1) + \begin{cases} 0 & , m \leq 1 \\ T_{INS}(A, m-1) + \Theta(K^A_{m-1}) \end{cases} =$$

$$= \begin{cases} \Theta(1) & , m \leq 1 \\ T_{INS}(A, m-1) + \Theta(K^A_{m-1}), & \text{ALTRIMENTI} \end{cases}$$

$$\leq \cdot \leq$$

$$\begin{cases} a \\ T_{INS}(A, m-1) + c_1 K^A_{m-1} \end{cases} \qquad \begin{cases} a \\ T_{INS}(A, m-1) + c_2 K^A_{m-1} \end{cases}$$

$$T^A(n) = \begin{cases} e & , \text{ if } n \leq 1 \\ T^A(n-2) + ck^A_{n-2} & , \text{ altfel} \end{cases}$$

11 OTTOBRE

$$T^A(m) = \begin{cases} a & \text{, if } m = 1 \\ T^A(m-1) + cK^A_{m-1} & \text{, altrimenti} \end{cases}$$

METODI PER RISOLVERLO

1) $T^A(m) = T^A(m-1) + c\,K^A_{m-1}$ per $m > 1$

$T^A_{(m-1)} = T^A(m-2) + cK^A_{m-2}$

m $T^A(m-2) = T^A(m-3) + c\,K^A_{m-3}$

$\vdots$

$T^A(2) = T^A(1) + cK^A_1$ → caso 2

$T^A(1) = a$ → caso base

$$\sum_{i=0}^{m-2} T^A(m-1) = \sum_{i=1}^{m-1} T^A(m-1) + cK^A_{m-1}$$

$$\sum_{i=0}^{m} \overset{\shortparallel}{T^A(1)} \qquad \sum_{i=1}^{m-1} \left[\overset{\shortparallel}{T^A(i)} + cK^A_i\right] + a =$$

$$= \sum_{i=1}^{m-1} T^A(i) + c\sum_{i=1}^{m-1} K^A_i + a$$

$$\sum_{i=1}^{m} T^A(1) - \sum_{i=1}^{m-1} T^A(i) = c \cdot \sum_{i=1}^{m-1} K^A_i +$$

$$c(m-1) + a = cm + (a-c) = \Theta(m)$$ CASO MIGLIORE

$$c\,\frac{m(m-1)}{2} + a = \Theta(m^2)$$ CASO PEGGIORE

MERGE SORT

Algoritmo Merge sort (A, m)

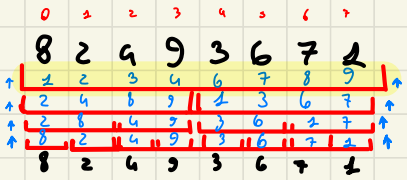
```
MergSort (A, 0, m-1)
```

Algoritmo Merge sort (A, p, r)

```
1) if (p < r) then
2)     q ← ⌈(p+r)/2⌉   (MEDIANO)
3)     MergSort (A, p, q)
4)     MergSort (A, q+1, r)
5)     Merge (A, p, q, r)
```

$n > 1 \rightarrow$ G

Es.



Algoritmo Merge (A, p, q, r)

1) $L \leftarrow Copy(A, p, q)$
2) $R \leftarrow Copy(A, q+1, r)$
3) $i, j \leftarrow 0$
4) for $k = p$ to r do
5) if $(i \leq q-r) \wedge ((j \geq r-q) \vee (L[i] \leq R[j]))$ then
6) $A[k] \leftarrow L[i]$
7) $i \leftarrow i+1$
   else
8) $A[k] \leftarrow R[j]$
9) $j \leftarrow j+1$

$$T^A(p,r) = \begin{cases} a & \text{, if } p \geq r \\ T^A(p, \underbrace{\left[\frac{p+r}{2}\right]}_{q}) + T^A(\underbrace{\left[\frac{p+r}{2}\right]+1}_{q+1}, r) + \underbrace{T^A_{merge}(p,r)}_{\Theta(r-p)} \end{cases}$$

$m = r-p+1$

$$T^A(m) = \begin{cases} a & \text{, se } m \leq 1 \\ T^A(\lfloor \frac{n}{2} \rfloor) + T^A(\lceil \frac{m}{2} \rceil) + cm \end{cases}$$

13 OTTOBRE

$$T(m) = \begin{cases} a \\ T(\lceil \frac{m}{2} \rceil) + T(\lfloor \frac{m}{2} \rfloor) + cm \end{cases}$$

$$T(m) = \begin{cases} a & , m \leq 1 \\ 2T\left(\frac{m}{2}\right) + cm, & \text{altrimenti} \end{cases}$$

$$2^0 T(m) = 2T\left(\frac{m}{2}\right) + cm$$

$$2^1 T\left(\frac{m}{2}\right) = 2^2 T\frac{m}{4} + c\frac{m}{2}$$

$$2^2 T\left(\frac{m}{4}\right) = 2^3 T\frac{m}{8} + c\frac{m}{4}$$

$$\vdots$$

$$2^k T(\leq 1) = 2^k a$$

$\} k$

$$\Rightarrow T(m) = kcm + 2^k a$$

$$\sum_{i=0}^{k} 2^i T\left(\frac{m}{2^i}\right) = \sum_{i=0}^{k-2} \left[2^{i+1} \cdot T\left(\frac{m}{2^{i+1}}\right) + c\right] + 2^k \cdot a$$

$$T(m) + \sum_{i=2}^{k} 2^i T\left(\frac{m}{2^i}\right) = \sum_{i=0}^{k-1} \left[2^{i+1} \cdot T\left(\frac{m}{2^{i+1}}\right)\right] + \sum_{i=0}^{k-1} [cm] + 2^k a$$

$$\sum_{i=0}^{k} 2^{i+1} \cdot T\left(\frac{m}{2}\right) \qquad k \cdot cm + 2^k a$$

APPLICO SOSTITUZIONE

$$\frac{m}{2^{k(m)}} \leq 1 \Rightarrow m \leq 2^{k(m)} \Rightarrow \log_2 m \leq \log_2(2^{k(m)}) = k(m)$$

$$k(m) = \lceil \log_2 m \rceil$$

$k_m = \log_2 m$

$$T(m) = cm\,k + 2^k a = cm \log_2 m + 2^{\log_2 m} a = cm \log_2 m + ma =$$

$$= \Theta(m \log_2 m) \rightarrow \text{ANDAMENTO DEL MERGESORT}$$

REGOLA RICORSIVA

$$T\left(\frac{n}{2}+1\right) = T\left(\left\lceil \frac{\frac{n}{2}+1}{2} \right\rceil\right) + T\left(\left\lfloor \frac{\frac{n}{2}+1}{2} \right\rfloor\right) + c\left(\frac{n}{2}+1\right)$$

$$T\left(\left\lceil \frac{n}{2} \right\rceil\right) = T\left(\left\lceil \frac{\frac{n}{2}}{2} \right\rceil\right) + T\left(\left\lfloor \frac{\frac{n}{2}}{2} \right\rfloor\right) + c\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

Esercizi

$$T(n) = \begin{cases} a & \text{, if } n \le 1 \\ 3T\left(\frac{n}{4}\right) + n & \text{, altrimenti} \end{cases}$$

$$T(n) = 3T\left(\frac{n}{4}\right) + n$$

$$3T\left(\frac{n}{4}\right) = 3^2 T\left(\frac{n}{16}\right) + 3\frac{n}{4}$$

$$3^2 T\left(\frac{n}{16}\right) = 3^3 T\left(\frac{n}{64}\right) + 3^2 \frac{n}{16}$$

$$\vdots$$

$$3^k T(1) = 3^k a$$

$$\Rightarrow T(n) = \sum_{i=0}^{k-1} \left(\frac{3}{4}\right)^i n + 3^k a =$$

$$= n\left[\sum_{i=0}^{k-1} \left(\frac{3}{4}\right)^i\right] + 3^k a$$

$$\sum_{i=0}^{n} r^i = \frac{1 - r^{n+1}}{1 - r} = \frac{r^{n+1} - 1}{r - 1}$$

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \qquad 0 < r < 1$$

$$n \frac{1 - \left(\frac{3}{4}\right)^k}{\frac{1}{4}} + 3^k a = 4n \left(1 - \left(\frac{3}{4}\right)^k\right) + 3^k a$$

$$\frac{n}{4^k} \leq 1$$

$$n \leq 4^k$$

$$\log_4 n \leq \log_4 4^k$$

$$\log_4 n \leq k$$

$$k = \log_4 n$$

$$T(n) = 4n \left(1 - \left(\frac{3}{4}\right)^k\right) + 3^k a = \underset{\uparrow}{\underset{k = \log_4 n}{}} 4n \left(1 - \left(\frac{3}{4}\right)^{\log_4 n}\right) + 3^{\log_4 n} a =$$

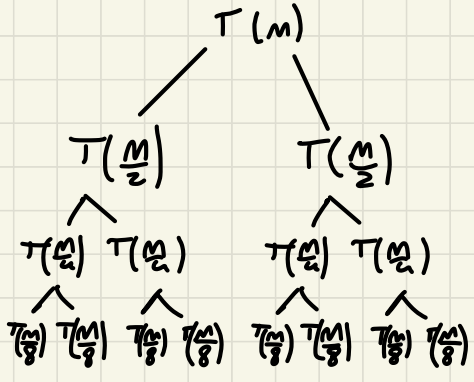
$$= 4n \left(1 - \frac{3^{\log_4 n}}{4^{\log_4 n}}\right) + 3^{\log_4 n} a$$

$$4^{\log_4 n} = n$$

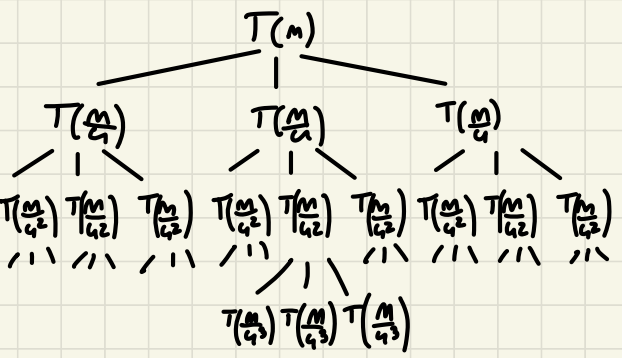
$$3 = 4^{\log_4 3} \qquad 3^{\log_4 n} = \left(4^{\log_4 3}\right)^{\log_4 n} = \left(4^{\log_4 n}\right)^{\log_4 3} = n^b \qquad b < 1$$

$$T(n) = 4n \left(1 - \frac{n^b}{n}\right) + n^b a = \Theta(n)$$

ALBERO DI RICORSIONE



$i=0$	$1=2^i$	$cn = c\frac{n}{2^0}$	$2^i \cdot c\frac{n}{2^i} = cn$
$i=1$	$2=2^i$	$c\frac{n}{2} = c\frac{n}{2^1}$	$2^i \cdot c\frac{n}{2^i} = cn$
$i=2$	$4=2^i$	$c\frac{n}{4} = c\frac{n}{2^2}$	$2^i \cdot c\frac{n}{2^i} = cn$
$i=3$	$8=2^i$	$c\frac{n}{8} = c\frac{n}{2^3}$	$2^i \cdot c\frac{n}{2^i} = cn$

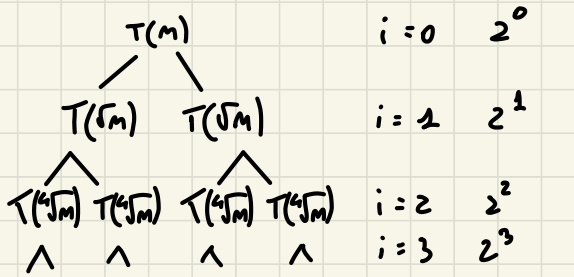


$i=0=3^0$	$\left(\frac{3}{4}\right)^0 n$
$i=1=3^1$	$\left(\frac{3}{4}\right)^1 n$
$i=2=3^2$	$\left(\frac{3}{4}\right)^2 n$
$i=3=3^3$	$\left(\frac{3}{4}\right)^3 n$

$$\sum_{i=0}^{\log_4 n} \left(\frac{3}{4}\right)^i n$$

14 OTTOBRE

$$T(m) = \begin{cases} 1 & \text{if } m \leq 2 \\ 2T(\sqrt{m}) + 1, & \text{altrimenti} \end{cases}$$



$i = 0$	2^0	1	$2^0 = 2^i$
$i = 1$	2^1	1	$2^1 = 2^i$
$i = 2$	2^2	1	$2^2 = 2^i$
$i = 3$	2^3	1	$2^3 = 2^i$

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

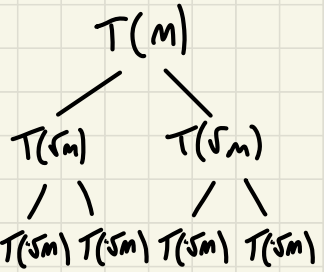
$T(3\sqrt{m}) + T(2\sqrt{m}) + 1$ N. NODI

$\sqrt[2^i]{m} = m^{\frac{1}{2^i}} \leq 2 \Rightarrow m \leq 2^{2^k} \Rightarrow \log_2 m \leq \log_2 2^{2^k} = 2^k \Rightarrow \log_2(\log_2(m)) \leq k$

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1 = 2^{\log_2(\log_2(m))+1} - 1 =$$

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

$$T(m) = \begin{cases} 1 \text{ o} & \text{if } m \le 2 \\ 2T(\sqrt{m}) + m, & \text{altrimenti} \end{cases}$$



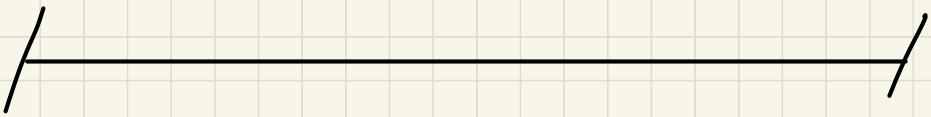
$2^0 m$

$2^1 \cdot \sqrt[2^1]{m}$

$2^2 \cdot \sqrt[2^2]{m}$

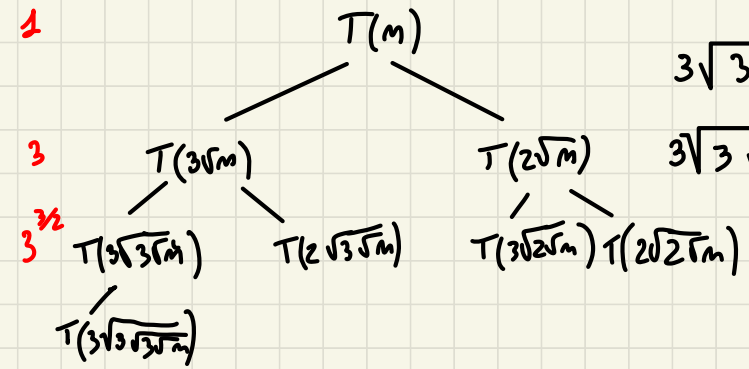
$2^i \cdot (m)^{\frac{1}{2^i}}$

$$\sum_{i=0}^{k-1} 2^i (m)^{\frac{1}{2^i}}$$



Es.

1



$$3\sqrt{3\sqrt{m}} = 3\sqrt{3} \cdot \sqrt[4]{m}$$

$$3\sqrt{3\sqrt{3\sqrt{m}}} = 3\sqrt{3\sqrt{3}} \cdot \sqrt[8]{m} =$$

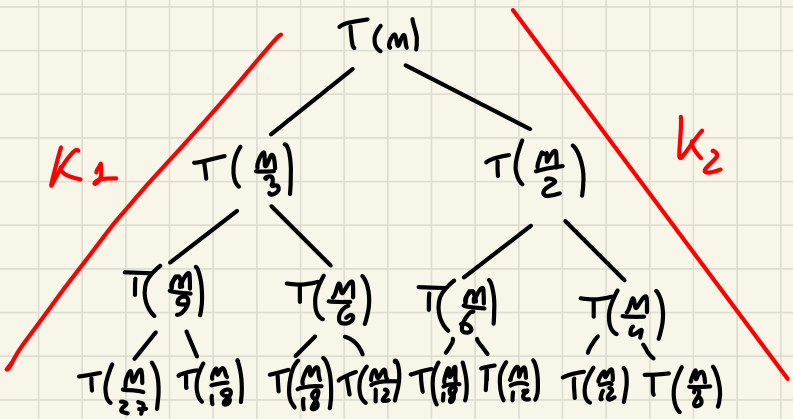
$$= 3 \cdot \left(3^{3/2}\right)^{\frac{1}{4}} =$$

$$= 3 \cdot 3^{3/4} = 3^{\frac{7}{4}}$$

$$2 \ge \frac{2^i - 1}{3^{2^i - 1}} \cdot \frac{1}{m^{2^i}}$$

ESERCIZIO FATTIBILE

$$T(m) = \begin{cases} 1 & , m \le 1 \\ T\left(\frac{m}{3}\right) + T\left(\frac{m}{2}\right) + m & , \text{altrimenti} \end{cases}$$



RAMO 1° $\left(\frac{1}{3} + \frac{1}{2}\right) m = \frac{5}{6} m$

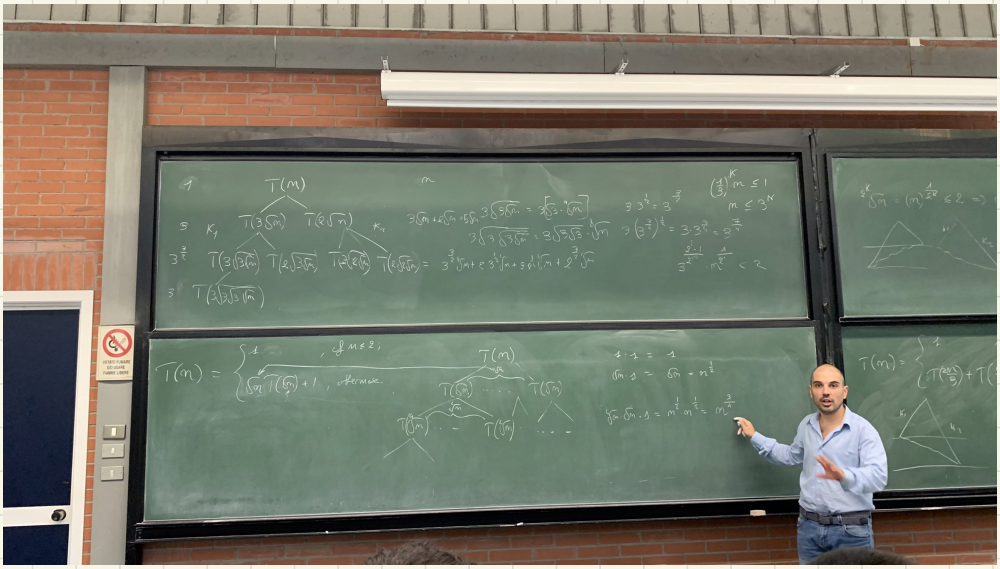
RAMO 2° $\left(\frac{1}{9} + \frac{1}{6} + \frac{1}{6} + \frac{1}{4}\right) m = \frac{4+6+6+9}{36} m = \frac{25}{36} m$

RAMO 3° $\frac{1}{27} + \frac{3}{18} + \frac{3}{12} + \frac{1}{8} = \left(\frac{5}{6}\right)^1 m$

$\left(\frac{5}{6}\right)^i m$

$$\sum_{i=0}^{k_1 - 1} \left(\frac{5}{6}\right)^i m = m \sum_{i=0}^{k-1} \left(\frac{5}{6}\right)^i < m \sum_{i=0}^{\infty} \left(\frac{5}{6}\right)^i = m \frac{1}{1 - \frac{5}{6}} = 6m \quad O(m)$$

$$\sum_{i=0}^{k_2 - 1} \left(\frac{5}{6}\right)^i m = m \frac{1 - \left(\frac{5}{6}\right)^{k_2}}{1 - 5/6} = 6m \left(1 - \left(\frac{5}{6}\right)^{k_2}\right)$$



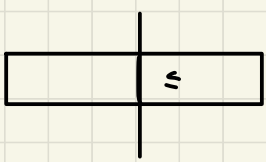
18 OTTOBRE

$$T(n) = \begin{cases} 1 & , \text{se } n \leq 2 \\ \sqrt{n}\, T(n) + 1 & , \text{altrimenti} \end{cases}$$

$$\sum_{i=0}^{k} n^{\frac{2^i - 1}{2^i}} = \sum_{i=0}^{k} n^{1 - \frac{1}{2^i}} = \sum_{i=0}^{k} \frac{n}{n^{\frac{1}{2^i}}} = n \underbrace{\sum_{i=0}^{k} \frac{1}{n^{\frac{1}{2^i}}}}_{\leq k+1}$$

ALGORITMO SELECTION SORT (A, n)

1) for i = n - 1 downto 1 do (VAI A RITROSO) — $\sum_{i=1}^{n-1} i$
2) $\quad$ m ← MAX (A, i) — Θ(1)
3) $\quad$ SWAP (A, i, m)

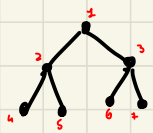


ALGORITMO MAX (A, i)

1) m ← 0
2) for j = 1 to i do — Θ(i)
3) $\quad$ if A[m] < A[j]
4) $\quad\quad$ m ← j
5) RETURN m

1) $T = \emptyset$ ALBERO BINARIO AB

2) $\exists\, r \in T,\ \exists\, L, R \subseteq T \setminus r$
$L \cap R = \emptyset$
$L, R \in AB$



ALBERO BINARIO

STESSI N. NODI

L'ALTEZZA È LOGARITMICA

h	T	n
-1	$\emptyset$	0
0	•	1
1		3
2		7

$$n(h) = 2^{h+1} - 1$$

$$n(h+1) = 2n(h) + 1 = 2\left(2^{h+1} - 1\right) + 1 = 2^{h+2} - 2 + 1 = n(h+1)$$

$$2^{h+1} - 1 \geq n$$

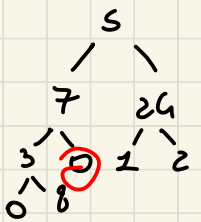
$$h \geq \left\lceil \log_2(n+1) - 1 \right\rceil$$

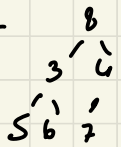
DEFINIZIONE DI ALBERO COMPLETO BINARIO

UN ALBERO COMPLETO BINARIO HA TUTTI I NODI INTERI CON 2 FIGLI TRANNE 1 E LE FOGLIE SONO AD ALTEZZA h O $h-1$

COME RAPP. ARRAY CON UN ALBERO

0	1	2	3	4	5	6	7	8
5	7	24	3	0	1	2	0	8





8	3	4	5	6	7

È ISOMORFO

$L(i) = 2i + 1$

$R(i) = 2i + 2$

MAX HEAP (SI DEVE FARE IN ORDINE DECRESC.)

Heapify (quasi HEAP AD HEAP)

Build Heap

Heap Sort

20 OTTOBRE

Heap Sort (A, n)

```
1) BuildHeap (A, n)
2) for i = n-1 to 1 do
3)     swap (A, 0, i)
4)     Heapify (A, i, 0)
```

$$\log_2\left(\prod_{i=1}^{n-1} i\right) = \log_2((n-1)!)$$

$$\uparrow$$

$$O\left(\sum_{i=2}^{n-1} \log_2 i\right)$$

BUILD Heap

```
1) for i = ⌊n/2⌋ - 1 down to 0 do
2)     Heapify (A, n, i)
```

$\longrightarrow O(\log_2 i)$

Heapify (A, n, i)

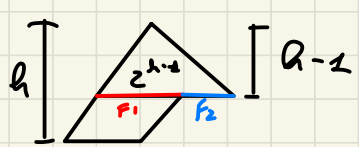
```
1) (max, l, r) ← (i, L(i), R(i))
2) if (l < n) ∧ (A[max] < A[l]) then
3)     max ← l
4) if (r < n) ∧ (A[max] < A[r]) then
5)     max ← r
6) if (max ≠ i) then
7)     swap (A, max, i)
8)     Heapify (A, n, max)
```

$max \leftarrow \underset{i,l,r}{\arg\max} \{A[i], A[l], A[r]\}$

$n(h) = 2^{h+1} - 1$

$n(h-1) = 2^h - 1$

$f(h) = 2^h \quad f(h-1) = 2^{h-1}$



$f_1 + f_2 = f_1 + 2(2^{h-1} - f_1) = f_1 + 2^h - 2f_1 = 2^h - f_1 \rightarrow f_2 \leq 2^{h-1}$

21 OTTOBRE

$$\sum_{i=1}^{k} i \frac{n}{2^{i+2}} = n \sum_{i=2}^{k} i \left(\frac{1}{2}\right)^{i+1} = \frac{n}{4} \sum_{i=2}^{k} i \left(\frac{1}{2}\right)^{i-1} = \frac{n}{4} \sum_{i=1}^{k} \left[i \cdot x^{i-1} \right]_{x=\frac{1}{2}} =$$

$$\left[\frac{n}{4} \sum_{i=1}^{k} i \cdot x^{i-1} \right]_{x=\frac{1}{2}} = \left[\frac{n}{4} \sum_{i=1}^{k} \frac{d}{dx} x^i \right]_{x=\frac{1}{2}} =$$

$$= \left[\frac{n}{4} \frac{d}{dx} \sum_{i=1}^{k} x^i \right]_{x=\frac{1}{2}} \leq \left[\frac{n}{4} \frac{d}{dx} \sum_{i=1}^{\infty} x^i \right] = \left[\frac{n}{4} \frac{d}{dx} \frac{1}{1-x} \right]_{x=\frac{1}{2}} =$$

$$= \left[\frac{n}{4} \frac{d}{dx} (1-x)^{-1} \right]_{x=\frac{1}{2}} = \left[\frac{n}{4} \frac{(-1)\cdot(-1)}{(1-x)^2} \right]_{x=\frac{1}{2}} = \frac{n}{4} \cdot 4 = n$$

Algoritmo Quick sort (A, n)

```
Quick sort(A, 0, n-1)
```

Algoritmo Quick sort (A, p, r)

```
1) if p < r then
2)     q ← partizione (A, p, r)
3)     Quick sort (A, p, q)
4)     Quick sort (A, q+1, r)
```

PARTIZIONE PROPRIETÀ (A, p, r)

1) $p \leq q < r$

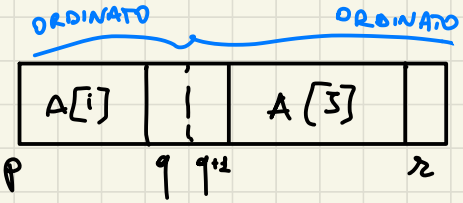
2) $\forall p \leq i \leq q, \forall q+1 \leq j \leq r, \; A[i] \leq A[j]$

3) A dopo partizione sia una permutazione di A prima di partizione

$r - p + 1 = n$

$q - p + 1 < r - p + 1 = n$

$r - (q+1) + 1 = r - q < r - q + 1 \leq r - p + 1 = n$



Algoritmo (A, p, r)

1) $x \leftarrow A[p]$

2) $i \leftarrow p - 1$

3) $j \leftarrow r + 1$

repeat

  repeat

4) j --

5) until $(A[j] \leq x)$

```
6)  | repeat
    |   i ++
7)  | until (A[i] ≥ x)
8)  | if i < j then
9)  |   swap (A, i, j)
10) until (j ≤ i)
11) return j
```

$d_0 = j_0 - i_0 = (n+1) - (p-1) = n - p + 2$ → PRIMA DEL REPEAT

$d_1 = j_1 - i_1 \leq d_0 - 2$ → PRIMO PASSAGGIO
j --, i ++

$j_1 < j_0$
$i_1 > i_0$

↖ le celle vengono toccate o da i o da j al più 1 sola volta

COSTO →
- i repeat → $\Theta(n)$
- + swap → $\Theta(\frac{n}{2})$

↳ tutto → $\Theta(n)$

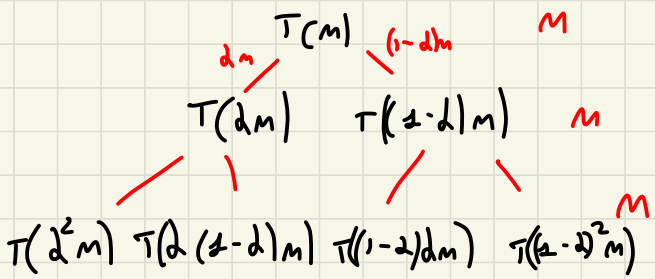
$$T(n) = \begin{cases} 1 & , \text{se } n \leq 1 \\ T(1) + T(n-1) + \Theta(n) & , \text{altrimenti} \end{cases}$$

→ CASO PEGGIORE

CASO MIGLIORE → TUTTI ELEMENTI UGUALI

CASO PARTICOLARE

$$T(n) = \begin{cases} 1 & , \text{se } n \leq 1 \\ T(\alpha n) + T((1-\alpha)n) + \Theta n & , \text{altrimenti:} \\ & \alpha \in (0,1) \end{cases}$$



25 OTTOBRE

i_0 j_0

$\forall p \le \hat{\imath} < i_0 \quad \forall j_0 < \hat{\jmath} \le r \quad A[\hat{\imath}] \le A[\hat{\jmath}]$

$\forall p \le \hat{\imath} < i_K \quad \forall j_K < \hat{\jmath} \le r \quad A[\hat{\imath}] \le A[\hat{\jmath}]$

VALE PER $K+1$?

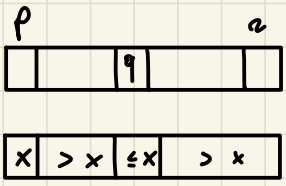
$A[i] \le x \le A[j]$

$\forall p \le \hat{\imath} < i_{K+1} \quad \forall j_{K+1} < \hat{\jmath} \le r$

$T_M(m)$

RANGO DEL PIVOT q :

$$1 \le \left| \{ i \in [p, r] \mid A[i] \le A[q] \} \right| \le m \triangleq r - p + 1$$



RANGO	q	$[p, q]$
1	p	1
2	p	1
3	p+1	2
m	p+m-2	m-1

$$T_M(m) = \begin{cases} k_1 & \text{, se } m \le 1 \\ \frac{1}{m}\sum_{q=1}^{m} T_M(q_2) + T_M(m-q_2) + k_2 m & \text{, se } m > 1 \end{cases}$$

Θ_m

$T_M(m)$ ↓ Tempo Medio

$T_M(q_2)$ ↓ cella a sx

$T_M(m-q_2)$ ↓ cella a dx

$$T_M(m) = \frac{1}{m}\sum_{q=1}^{m}\left[T_M(q_2) + T_M(m-q_2) + k_2 m\right] =$$

(CASO: 1) $$= \frac{1}{m}\left[T_M(q_1) + T_M(m-q_1) + k_2 m + \sum_{q=2}^{m}\left[T_M(q_2) + T_M(m-q_2) + k_2 m\right]\right] =$$

$T_M(q_1) = T_M(1) = k_1$

$T_M(m-1) = k_3 m^2$

$$= \left[\frac{k_1}{m} + k_3 m + k_2\right] + \frac{1}{m}\sum_{q=2}^{m}\left[T_M(q_2) + T_M(m-q_2) + k_2 m\right]$$

$\Theta_m = k_4 m$

Θ_m Θ_1

$\frac{k_1}{m} \to 0$ 1

↓ piccolo di 2

$$\le k_4 m + \frac{1}{m}\sum_{q=2}^{m}\left[T_M(q-1) + T_M(m-(q-1)) + k_2 m\right] =$$

$$= k_4 m + \frac{1}{m}\sum_{i=1}^{m-1}\left[T_M(i) + T_M(m-i) + k_2 m\right] =$$

$$= k_4 m + \frac{k_2 m(m-1)}{m} + \frac{1}{m}\sum_{i=1}^{m-1}\left[T_M(i) + T_M(m-i)\right] =$$

$$= (k_4 + k_2)m - k_2 + \frac{1}{m}\sum_{i=1}^{m-1}\left[T_M(i) + T_M(m-i)\right] =$$

$$= (k_4 + k_2)m - k_2 + \frac{2}{m}\sum_{i=1}^{m-1} T_M(i)$$

$$T_M(n) = \begin{cases} k_1 & , \text{se } n \leq 1 \\ (k_a + k_2)\, n - k_2 + \frac{2}{n} \sum_{i=1}^{n-1} T_M(i) & , \text{altrimenti} \end{cases}$$

DOBBIAMO DIMOSTRARE PER INDUZIONE

$$T_M(n) \leq c\, n \log n$$

$$T_M(2) = (k_a + k_2)\, 2 - k_2 + \frac{2}{2} \cdot k_1 \leq c\, 2 \cdot \log 2$$

$$\downarrow$$

$$c \geq (k_a + k_2) - \frac{k_2}{2} + \frac{k_1}{2}$$

$$\forall n \leq k \quad T_M(n) \leq c\, n \log n \Rightarrow T_M(k+1) \leq c\, k \log k$$

27 OTTOBRE

$$T_M(m) = \begin{cases} K_1 & , m \le 1 \\ K_3 m - K_2 + \frac{2}{m} \sum_{p=1}^{m-1} T_M(p) & , \text{altrimenti} \end{cases}$$

THEORAM, $T_M(m) \le c\, m \log m$, per qualunque $c \in \mathbb{R}^+$

IPOT. IND. $T_M(p) \le c\, p \log p$, $\forall p < m$

$$T_M(m) = K_3 m - K_2 + \frac{2}{m} \sum_{p=1}^{m-1} T_M(p) \le K_3 m - K_2 + \frac{2c}{m} \sum_{p=1}^{m-1} p \log p$$

($p \log p \to p \log m$)

$$\sum_{p=1}^{m-1} p \log p = \sum_{p=1}^{\lfloor \frac{m-1}{2} \rfloor - 1} p \log p + \sum_{p=\lfloor \frac{m-1}{2} \rfloor}^{m-1} p \log p = \log \frac{m}{2} \sum_{p=1}^{\lfloor \frac{m-1}{2} \rfloor - 1} p + \log m \sum_{p=\frac{m-1}{2}}^{m-1} p =$$

($\log p \to \log m$)

$$= (\log m - \log 2) \sum_{p=1}^{\lfloor \frac{m-1}{2} \rfloor - 1} p + \log m \sum_{p=\frac{m-1}{2}}^{\lfloor \frac{m-1}{2} \rfloor - 1} p = \log m \sum_{p=1}^{m-1} p - \sum_{p=1}^{\lfloor \frac{m-1}{2} \rfloor - 1} p =$$

$$= \log m \, \frac{m(m-1)}{2} - \frac{\lfloor \frac{m-1}{2} \rfloor \left(\lfloor \frac{m-1}{2} \rfloor - 1 \right)}{2} \le \frac{1}{2} \log m \, m^2 - \frac{m^2}{8}$$

$$\le K_3 m - K_2 + \frac{2c}{m} \left[\frac{m^2}{2} \log m - \frac{m^2}{8} \right] =$$

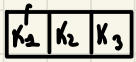
$$= c\, m \log m + K_3 m - K_2 - c \frac{m}{4} \le cm - \log m$$

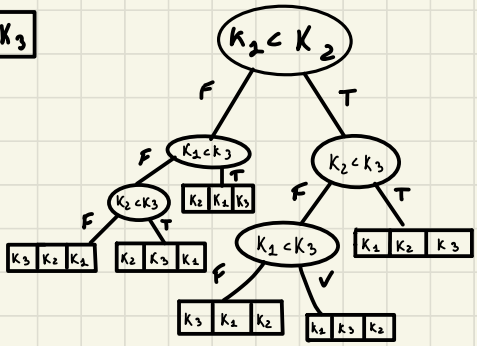
VOGLIAMO CHE $K_3 m - K_2 - \frac{cm}{4} \le 0$

$$4 \left(\frac{K_3 m - K_2}{m} \right) \le c \quad \to \quad 4K_3 - \frac{4K_2}{m} \le c$$

VISTO CHE LA C È ABBASTANZA GRANDE L'ANDAMENTO È DIMOSTRATO

$T_M(n) \leq c\, n \log n$

$A =$ 



$$n! = \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n e^{\alpha(n)}$$

IL NUMERO DI POSSIBILI INPUT È $n!$

$$\frac{1}{12n+1} \leq \alpha(n) \leq \frac{1}{12n}$$

$$2^{h-1} \leq n! \leq 2^h$$

$$\log(2^{h-1}) \leq \log n! \leq \log 2^h$$

$$h-1 \leq \log\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha(n)}\right) \leq h$$

$$h-1 \leq \log\left(\sqrt{2\pi n}\right) + \log\left(\left(\frac{n}{e}\right)^n\right) + \log e^{\alpha(n)} \leq h$$

$$\log(\sqrt{2\pi}) + \frac{1}{2}\log n + n\log n - n\log e + \alpha(n)\log e \leq h$$

$$h-1 \leq n\log n - n\log e + \frac{1}{2}\log n + \alpha(n)\log e + \log(\sqrt{2\pi}) \leq h$$

$n \log n$

26 OTTOBRE

$$T_M(n) = \frac{\sum_{i \in Path(D)} |i|}{n!}$$

i → NOME DEL PERCORSO

$Path(D)$ → N. DI PERCORSI

$\sum_{i \in Path(D)} |i|$ → PERCORSO ESTERNO DELL'ALBERO

$|i|$ → LUNGHEZZA DEL PERCORSO

NOI VOGLIAMO IL $T_{M_{min}}$

$$T_{M_{min}}(n) = \min_{D} T_M^D(n)$$

ALTERNATIVA A STIRLING (PARENTESI ALLA LEZIONE DEL 27/10)

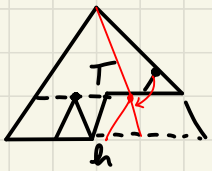
Si: $\log n! = \Theta(n \log n)$

$$\log\left(\prod_{i=1}^{n} i\right) = \sum_{i=1}^{n} \log(i)$$

$$\sum_{i=\frac{n}{2}}^{n} \log\left(\frac{n}{2}\right) \leq \sum_{i=\frac{n}{2}}^{n} \log(i) \leq \sum_{i=1}^{n} \log(i) \leq \sum_{i=1}^{n} \log n = n \log n$$

$$\log \frac{n}{2} = \frac{n}{2} \log n - \frac{n}{2} = \Theta(n \log n)$$

UN ALBERO COMPLETO SE LO TRASFORMO IN NON COMPLETO DOBBIAMO CAPIRE IL PERCORSO CHE FA



$f(h) \leq g(h)$

$P_E\ T$ = PERCORSO ESTERNO

$$P_E(T) - 2(h-1) + h - 2 - (h-1) + 2h$$

$$P_E(T) - \cancel{2h} + \cancel{2} + \cancel{h} - \cancel{2} - \cancel{h} + 1 + \cancel{2h}$$

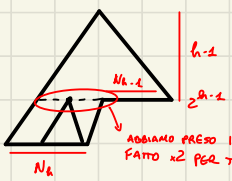
$$P_E(T) + 1 \longrightarrow \text{PERCORSO AUMENTATO DI 1}$$

VEDIAMO QUANTE FOGLIE SONO

$$N_h + N_{h-1} = n!$$

$$2\left(2^{h-1} - N_{h-1}\right) = N_h$$

2^{h-1} ↓ n. fogl albero pieno



$$2^h - N_h = 2N_{h-1}$$

$$2^h - 2N_{h-1} = N_h$$

$$\begin{cases} N_h + N_{h-1} = n! \\ N_h + 2N_{h-1} = 2^h \end{cases}$$

$$\begin{cases} N_{h-1} = 2^h - n! \\ N_h = 2n! - 2^h \end{cases} \longrightarrow \text{LA SOLUZIONE È LA RISOLUZIONE}$$

CALCOLARE IL PERCORSO ESTERNO È BANALE

$$P_e(n) = h \cdot N_h + (h-1) N_{h-1} = h \cdot (2n! - 2^h) + (h-1)(2^h - n!) =$$

$= \cancel{2}\log m! - \cancel{\log 2^h} + \cancel{\log 2^h} - \cancel{\log m!} - 2^h + m! =$

$= \log m! + m! - 2^h = (h+1)\, m! - 2^h$

$$T_{M_{min}}(n) = \frac{P_\varepsilon(n)}{n!} = \frac{(h+1)\, n! - 2^h}{n!} = (h+1) - \frac{2^h}{n!}$$

h SI CALCOLA $= \lfloor \log n \rfloor$

↓

altezza ALBERO completo

CALCOLARE NUMERO NODI INTERNI SAPENDO N. FOGLIE

FOGL.	NODI INT.	
1	0	
2	1	
6	5	
n	$n-1$	→ STIMA ASINTOTICA

SE LE FOGLI SONO $n!$ → N. NODI $= 2n! - 1$

$h = \log(2n! - 1) \leq \log 2 + \log n!$

APPROSSIMANDO IL -1 L'ABBIAMO TOLTO

$$= (\log n! + \log 2 + 1) - \frac{2^{\log n! + \log 2}}{n!} = \log n! + 2 - \frac{2n!}{n!} = \log n!$$

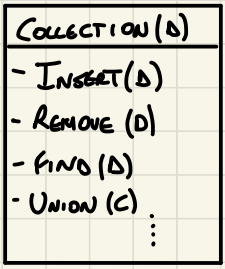
$\log n! \in \Theta(n \log n)$

QUALUNQUE ALGORITMO NON PUÒ ESSERE $< \Theta\, n \log n$

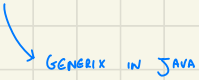
HO FATTO REGISTRAZIONE RIASSUNTO

STRUTTURE DATI E COLLEZIONI DI DATI

FUNZIONI DA USARE SULLE COLLEZIONI DI DATI



VEDREMO I METODI



3 NOVEMBRE

TOTACV => ARG(D) ORDERED(D)
• Soet

STACK
PUSH(D)
TOP(D):D
POP
TOP & POP():D
IS EMPTY():B

CODA

S.PUSH(D) = S ← PUSH(S, D)

↳ SI IMPLEMENTA CON UN ARRAY

QUEUE(D)
ENQUEUE(D)
HEAD():D
DEQUEUE()
HEAD e DEQUEUE()
IS EMPTY():B

PILA

B. L'ARRAY PUÒ ESSERE VISTO COME COLLEZIONE

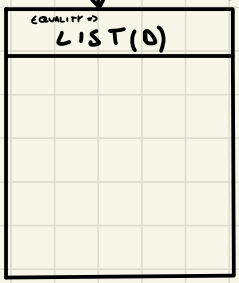
EQUALITY(D) => COLLECTION (D)
• INSERT (D)
• REMOVE (D)
• SEARCH/FIND (D) :B

COLLECTION

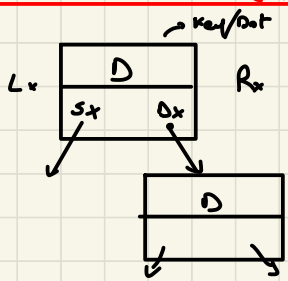
EQUALITY(D) => ARRAY(D)	
• INSERT (D)	O(1) → CASO MEDIO (AMMORTIZZATO)
• REMOVE (D)	O(n)
• SEARCH/FIND (D)	O(n)

TOTALLY ORDERED => ARRAY
• INSERT
• REMOVE
• FIND O(log n)

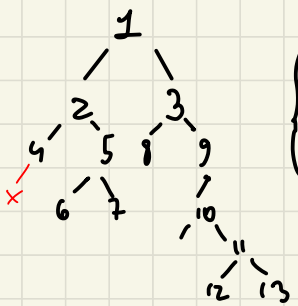
MEGLIO USARE LA LISTA
COSÌ AGGIUNGI SOLAMENTE
L'ARGOMENTO NUOVO ANZICCHÉ COPIARE L'ARRAY, AGGIUNGI E RICOPIARE



DEFINIAMO CONCETTO DI ALBERO



es.



A : 1 2 3 4 5 8 9 6 7 10 11 12 13

- PRE ORD: 1 2 4 5 6 7 3 8 9 10 11 12 13
- IN ORD: 4 2 6 5 7 1 8 3 10 12 11 13 9
- POST ORD: 4 6 7 5 2 8 12 13 11 10 9 3 1

DEPTH FIRST VISIT (DFV)
(BFS)

Algorithm DFV Generic. (x, F, a) $\quad a \in A$

BT Ref (Δ)

$f : \Delta \times A \to A$

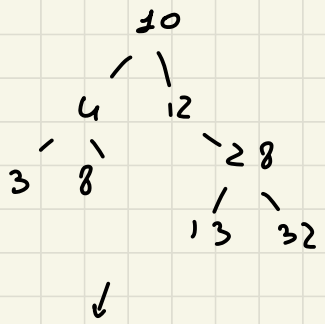
1) if $x \neq \perp$ then
2) $\quad a \leftarrow F(x.\text{Key}/x.\text{cat}, a)$
3) $\quad a \leftarrow \text{DFV Pre}(x.S_x, F, a)$
4) $\quad a \leftarrow \text{DFV Pre}(x.D_x, F, a)$
5) RETURN a

IN

POST

10 NOVEMBRE

```
Algorithm search ( x, d)
1) if x = ⊥ then
2)    return ⊥
      else
3)    if x.dat < d then
4)       return search (x.dx, d)
5)    else if x.dat > d then
6)       return search (x.sx, d)
      else
7)       return T
```



3

13

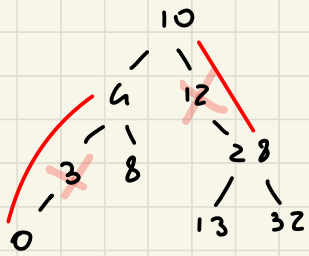
32

8

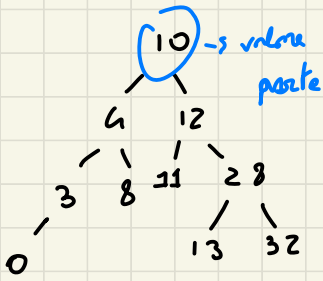
ogni nodo ha un contatore

```
Algoritmo Insert (x, d)
1) if x = ⊥ then
2)     return BuildNode (d)
   else
3)     if x.dat < d then
4)         x.dx ← Insert (x.dx, d)
5)     else x.dat > d then
6)         x.sx ← INSERT (x.sx, d)
       else
7)         return x
```

PROVIAMO A CANCELLARE 12, 3



ADESSO CANCELLIAMO IL 10



→ valore maggiore della parte sx

↓

quindi prendiamo il minimo dell'albero a dx

Algorithm Min(x)

```
1) if x.sx = ⊥ then
2)     return x
   else
3)     return Min(x.sx)
```

UGUALE PER IL MAX

Algorithm Min(x) -> VERSIONE ITERATIVA

```
1) WHILE x.sx ≠ ⊥ do
2)     x ← x.sx
3) return x
```

Algorithm Get & Delete Min(x) (RICORSIVA)

```
1) if x.sx = ⊥ then
2)     d ← x.dat
3)     r ← x.rx
4)     Deallocate(x)
5)     return (d, r)
   else
6)     (d, r) ← Get & Delete Min(x.sx)
```

```
7)  x.sx ← 2
8)  return (d, x)
```

Algorithm Get & Delete Min (x, p)

```
1)  if x.sx = ⊥ then
2)  |  d ← x.dat
3)  |  swap child (p, x, x.dlx)
4)  |  return d
    else
5)  |  return Get & Delete (x.sx, x)
```

```
if p.sx = x then
|  p.sx ← y
else
|  p.dlx ← y
Delete code (x)
```

Algorithm Delete (x, d)

```
1)  if x = ⊥ then
2)  |  return ⊥
    else
3)  |  if x.dat < d then
4)  |  |  x.dlx ← delete (x.dat, d)
5)  |  else if x.dat > d then
6)  |  |  x.sx ← delete (x.sx, d)
    |  else
```

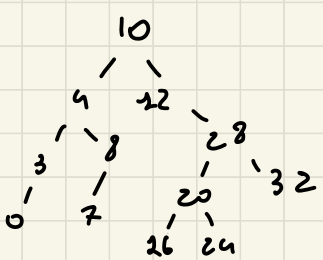
```
3)  return DeleteNode(x)        ( cancella il dato del nodo)
```

Algorithm DeleteNode (x)

```
if x.sx = ⊥ then
    y ∈ x.dx
    Deallocate(x)
    return y                    } SkipRight(x)
else if x.dx = ⊥ then
    SkipLeft(x)                 =
else
    x.dat ← GetAndDeleteMin(x.dx, x)
return x
```

Ricerca del Successore in un insieme

d	Successor(d)
1	3
5	7
13	16
8	10
12	16
32	1
-1	0
24	28



8 NOVEMBRE

BFV

Algoritmo (x, F, a) Q = coda

1) if $x \neq \perp$ then (VEDERE SE L'ALBERO NON È VUOTO)

2) Q ← ENQUEUE(EMPTY QUEUE, x) Q ← SINGELTONQUEUE (x)

REPEAT

3) x ← Head & Dequeue (Q)

4) a ← F (x.dat, a)

5) if xSx ≠ ⊥ then Q ← Enqueue (Q, xSx)

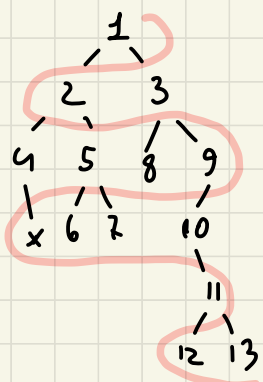
6) if xDx ≠ ⊥ then Q ← Enqueue (Q, xDx)

7) Until (isEmpty (Q))

8) Return a

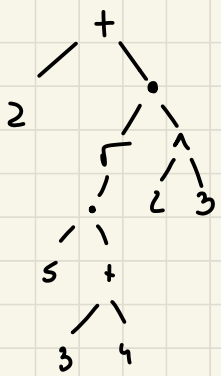
1	" "
2 3	" 1 "
3 4 5	" 1 2 3 "
4 5 8 9	⋮
5 8 9 x	
8 9 x 6 7	
9 x 6 7	
x 6 7 10	
6 7 10	
7 10	
10	
11	
12 13	
13	
∅	

QUESTA È LA CODA



$$f_d(d', a) = \begin{cases} T, & \text{se } d' = d \\ a, & \text{altrimenti} \end{cases}$$

$2 + \sqrt{5(3+4)} \cdot 2^3$ →



VISITA POST ORDER

$2\ 5\ 3\ 4\ +\ \cdot\ \sqrt{\ }\ 2\ 3\ \wedge\ \cdot\ +$

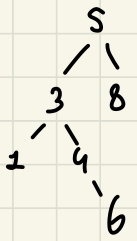
$2\ 5\ 7\ \cdot\ \sqrt{\ }\ 2\ 3\ \wedge\ \cdot\ +$

$2\ 35\ \sqrt{\ }\ 2\ 3\ \wedge\ \cdot\ +$

Alberi binari ricerca

T è un ABR/BST se

1) T è binario
2) $\forall x \in T, \forall y \in T_{x.sx} \forall z \in T_{x.dx}$ $y.dat \leq x.dat \leq z.dat$



1	3	4	5	6	8

Algoritmo Search (x, d)

```
1) if x = ⊥ then
2)     return 1
   else
3)     if x.dat = d then
4)         return T
5)     else if x.dat < d
6)         return Search(x.dx, d)
       else
7)         return Search(x.sx, d)
```

11 NOVEMBRE

```
Algorithm succession (x, d)
1) if x = ⊥ then
2)     return ⊥
   else
3)     if x.dat ≤ d then
4)         return succession (x.dx, d)
       else
5)         s ← succession (x.sx, d)
6)         if s = ⊥ then
7)             return x
           else
8)             return s
```

QUESTO NON È ANCORA RICORSIVO

NON RICORSIVI IN CODA HANNO BISOGNO DELLO STACK

```
Algorithm Successor (x, d, s)
1) if x = ⊥ then
2)     return s
   else
3)     if x.dat ≤ d then
4)         return successor (x.dx, d, s)
       else
5)         return successor (x.sx, d, x)
```

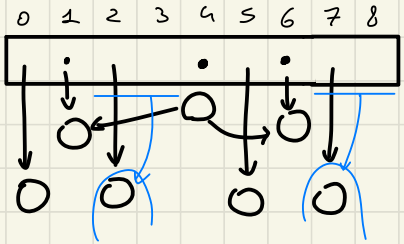
ALBERI BILANCIATI

ALBERO PERFETTAMENTE BILANCIATO

- SIMULA LA RICERCA BINARIA IN UN ARRAY

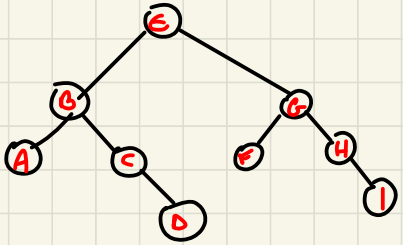
T è APB se $\forall x \in T \; \left| |T_{x.sx}| - |T_{x.dx}| \right| \leq 1$

↳ SONO GLI ALBERI PIÙ BASSI POSSIBILI → $\log n - 1$



Es.

A	B	C	D	E	F	G	H	I



ALBERI BILANCIATI

È BILANCIATO QUANDO L'ALTEZZA È $h = \Theta(\log n)$

ALBERI AVL

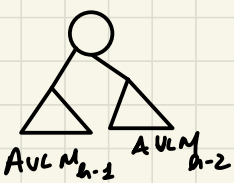
1) ABR (ALBERI BINARI DI RICERCA)

2) $\forall x \in T \quad \left| H(T_{sx}) - H(T_{x.dx}) \right| \leq 1$

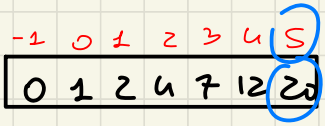
UN AVL minimo con ALTEZZA h È L'AVL DI ALTEZZA h CON IL MINIMO N. NODI

SE PRENDO:

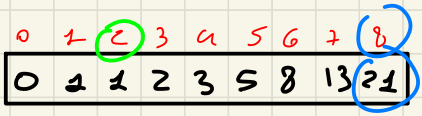
AVL Minimo h $\quad h \geq 1$



$$\begin{cases} N_h = 1 + N_{h-1} + N_{h-2} \\ N_{-1} = 0 \\ N_0 = 1 \end{cases}$$



↳ ANDAMENTO SIMILE A FIBONACCI



DISCREPANZA DI 3 E

DIFFERENZA DI 1

$N_h = Fib(h+3) - 1$

dim per induzione

Caso base: $(h = -1)$ $\quad 0 = N_{-1} = Fib(-1+3) - 1 = Fib(2) - 1 = 1 - 1 = 0$

Caso base: $(h = 0)$ $\quad 1 = N_0 = Fib(0+3) - 1 = Fib(3) - 1 = 2 - 1 = 1$

Caso ind: $(h \geq 1)$ $\quad N_h = 1 + N_{h-1} + N_{h-2}$

Per ipotesi ind.
$$\begin{cases} N_{h-1} = Fib((h-1)+3) - 1 = Fib(h+2) - 1 \\ N_{h-2} = Fib((h-2)+3) - 1 = Fib(h+1) - 1 \end{cases}$$

$$N_h = 1 + [Fib(h+2) - 1] + [Fib(h+1) - 1] =$$

$$= Fib(h+3) - 1$$

sez. aurea $\quad \varphi = \frac{1+\sqrt{5}}{2}$ $\qquad \frac{\varphi^n + (1-\varphi)^n}{\sqrt{5}}$

$$Fib(n) \geq \frac{1}{\sqrt{5}} \varphi^n$$

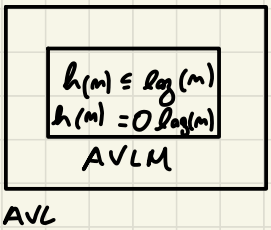
$$N_h = Fib(h+3) - 1 \geq \frac{1}{\sqrt{5}} \varphi^{h+3} - 1 \qquad \sqrt{5}(N_h + 1) \geq \varphi^{h+3}$$

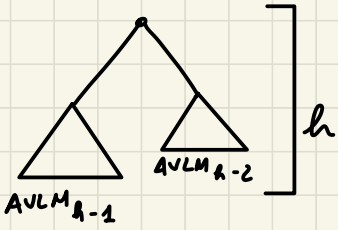
$$\log_\varphi(\sqrt{5}(n+1)) \geq h+3 \Rightarrow h \leq \log_\varphi(\sqrt{5}(n+1)) - 3$$

$\in \log n$

15 NOVEMBRE

90 e 25

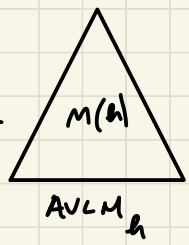


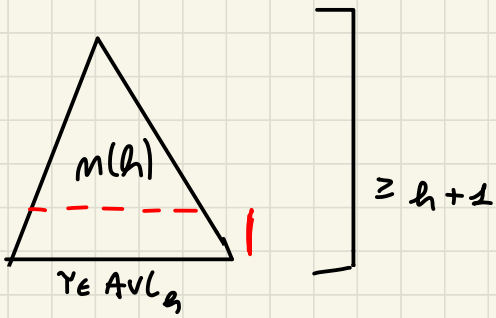


dim. x ASSURDO

DICIAMO CHE ESISTA

ALBERO MINIMO TRA GLI AVL





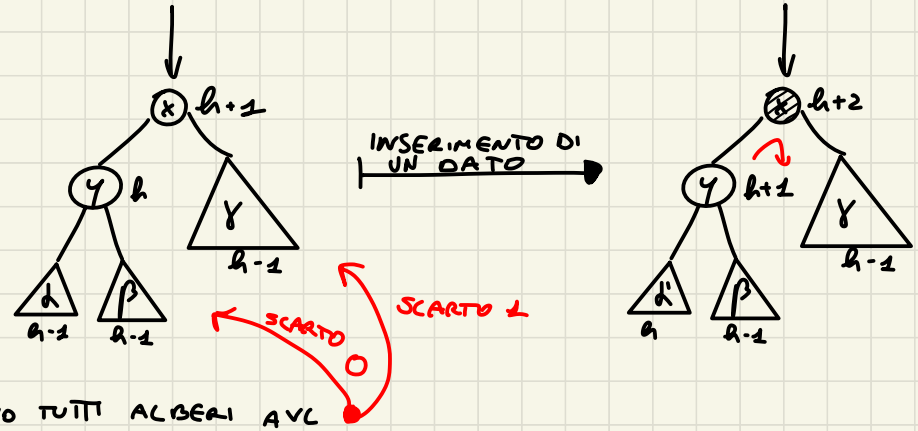
A PARITÀ DI NODI C'È UN ALBERO PIÙ PROFONDO

IL TAGLIO ROSSO RIDUCE IL NUMERO DI NODI E L'ALTEZZA QUINDI AVENDO UN NODO IN MENO DI QUELLO A SX CHE È IL MINIMO DIVENTEREBBE QUELLO A DX IL MINIMO

↳ CHE NON È POSSIBILE

HA CREATO UNA VIOLAZIONE

SEMPRE SE QUELLO A DESTRA È UN AVL, ALTRIMENTI NON C'È VIOLAZIONE

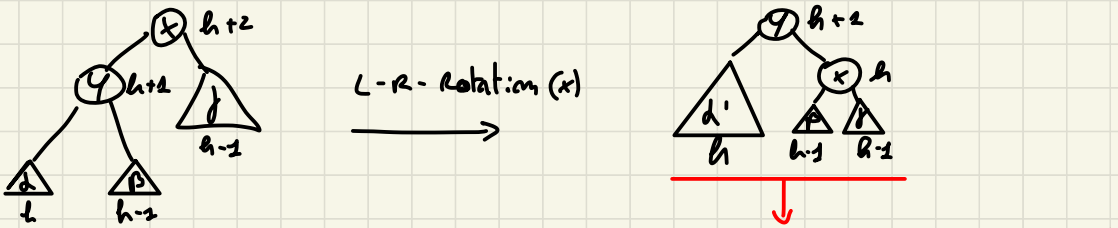


SONO TUTTI ALBERI AVL

PONIAMO (x) COME PROBLEMA (VIOLAZIONE) NELL'INSERIMENTO DI UN NODO

↳ PRENDIAMO QUESTO PERCHÉ È IL PIÙ BASSO

ROTAZIONE DA Sx e Dx INCENTRATA SUL NODO (x)

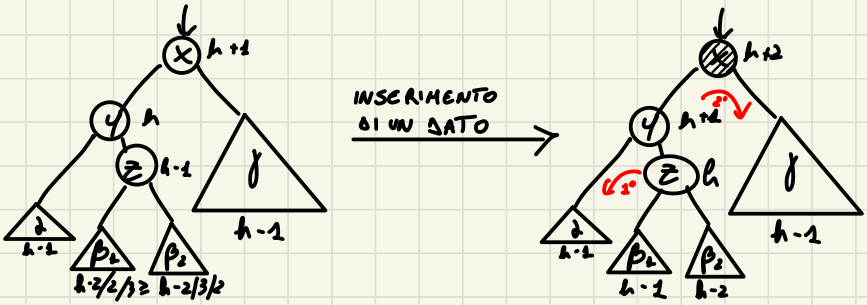


AVL (GLI SCARTI SONO GIUSTI)

quindi nella rotazione y diventa radice e x figlio

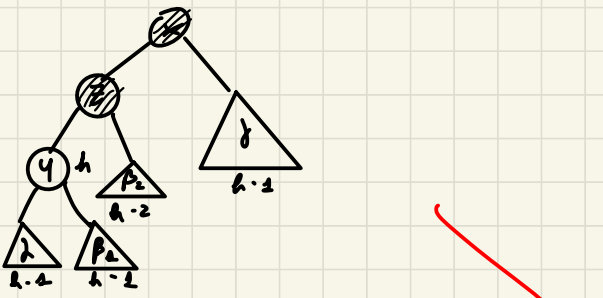
SONO SCOMPARSI I PROBLEMI

UN NODO È MAGGIORE DI TUTTO QUELLO CHE HA A Sx E MINORE DI TUTTO QUELLO CHE HA A Dx

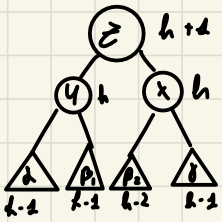


1° ROTAZIONE (R-L-Rotation(y))

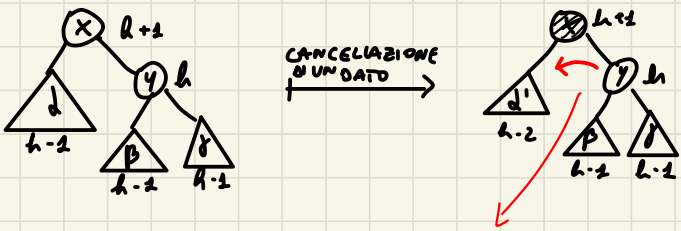
SIGNIFICA DANNEGGIATO



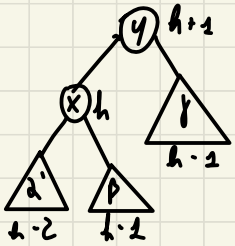
2° ROTAZIONE (L-R-Rot(x))



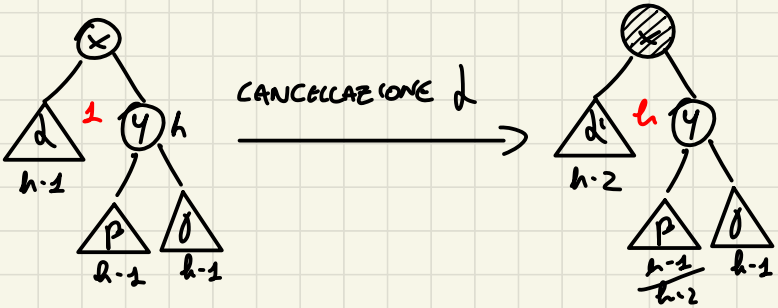
CANCELLAZIONE

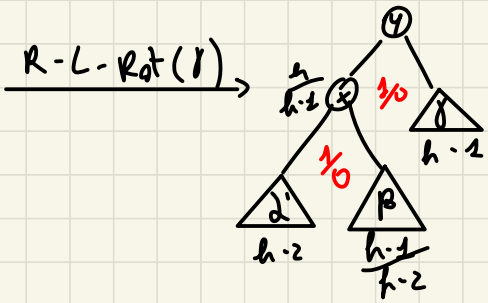


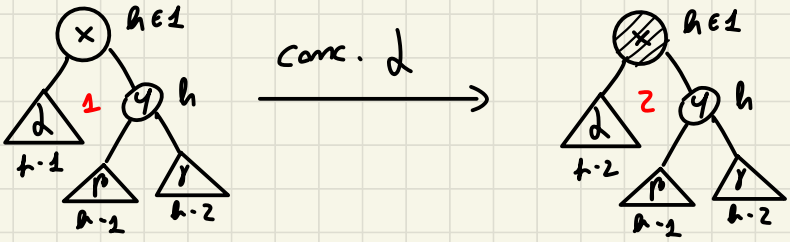
ROTAZIONE R - L - Rotar (x)

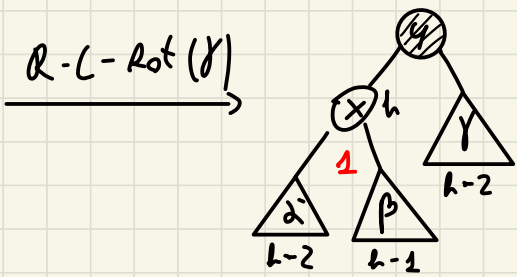


7 NOVEMBRE

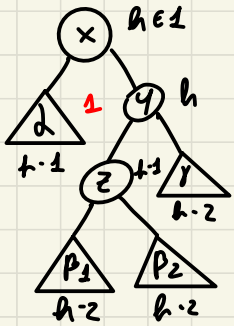

x
y
1
h
α
β
γ
h-1
h-1
h-1
CANCELLAZIONE α
h
h-2
h-2
h-1
h-2


R-L-Rot(γ)
y
x
1
h-1
1/0
1/0
α'
β
γ
h-2
h-1
h-2
h-1

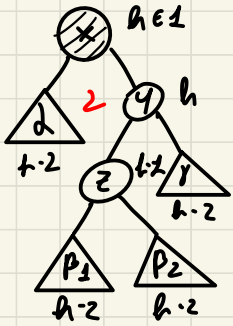

x
h∈1
y
1
h
α
β
γ
h-1
h-1
h-2
Conc. α
h∈1
2
h-2


R-L-Rot(γ)
x
h
1
α'
β
γ
h-2
h-1
h-2

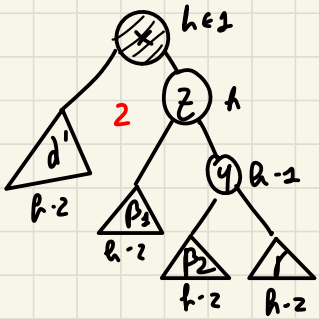
ANDIAMO A SPEZZARE L'ALBERO β



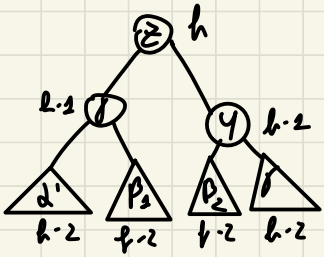
Conc. α



L - R - Rot(y)



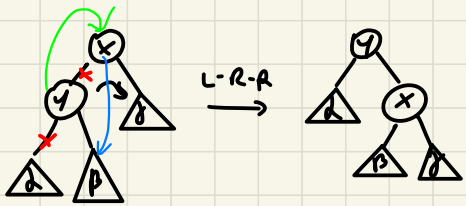
R - L - Rot(β)



ABBASSAMENTO DELL'ALBERO CHE PORTA AD UNA PROPAGAZIONE

ADESSO SCRIVIAMO TUTTI GLI ALGORITMI

ALGORITMO L-R. Rotation(x)



1) $y \leftarrow x.sx$ qui: abbiamo accesso a y
2) $x.sx \leftarrow y.dx$ •
3) $y.dx \leftarrow x$ •
4) return y

↳ complessità costante (Θ_1)

CONSIDERIAMO ADESSO DEI NODI SPECIALI

ABL / BST Node

D	
Sx	Dx

SEMPLICE

AVL Node (si aggiunge l'altezza)

D	h
Sx	Dx

ht → (tiene traccia dell'altezza)

SPECIALE

ALGORITMO Hgt AVL (x)

1) RETURN (if x = 1 then -1 else x.ht) → Θ_1

ALGORITMO Update Hgt AVL (x)

1) $H_L \leftarrow$ Hgt AVL (x.sx)
2) $H_R \leftarrow$ Hgt AVL (x.dx)
3) $x.ht \leftarrow 1 + \max\{h_L, h_R\}$

→ Θ_1

ALGORITMI DI ROTAZIONE

ALGORITMO L-R-Rotation(x)

1) y ← L-R-Rotation(x)
2) Update Hgt AVL(x)
3) Update Hgt AVL(y)
4) return(y)

Θ_1

ALGORITMO L-D-AVL Rotation(x) (Doppia Rotazione)

1) x.sx ← R-L-Rotation(x.sx)
2) return L-R-AVL Rotation(x)

Θ_1

↳ ANCHE QUESTO ALGORITMO CORREGGERÀ LE ALTEZZE

ALGORITMO LEFT BALANCE AVL(x) (SCOPRE IL PROBLEMA E LO CORREGGE)

ANALIZZIAMO I PROBLEMI DAL BASSO VERSO L'ALTO

→ ALBERO A SX PIÙ PROFONDO

1) if Hgt AVL(x.sx) − Hgt AVL(x.dx) = 2 then → abbiamo sicuramente uno sbilanciamento
2) if Hgt AVL(x.sx.dx) ≤ Hgt AVL(x.sx.sx) then → ci troviamo nel 1° caso
3) x ← L-R-AVL Rotation(x)
4) else x ← L-D-AVL ROTATION(x)

else

Θ_1

```
5)  Update Hgt_AVL (x)        → faccio un aggiornamento
6)  return x
```

ALGORITMI DI INSERIMENTO E CANCELLAZIONE AVL

ALGORITMO AVL INSERT (x, d)

```
2)  if x = ⊥ then
2)      return BuildNode AVL (d)
    else
3)      if x.dat > d
4)          x.sx ← AVL Insert (x.sx, d)
5)          x ← Balance (x)
6)      else if x.dat < d then
7)          x.dx ← AVL Insert (x.dx, d)
9           x ← R. Balance (x)
9)  return x
```

$O(a^{x.hg})$

ALGORITMO AVL Delete (x, d)

```
1) if x ≠ ⊥ then
2)     if x.dat > d then
3)         x.sx ← AVL Delete (x.sx, d)
4)         x ← R. Balance (x)
5)     else if x.dat < d then
6)         x.dx ← AVL Delete (x.dx, d)
7)         x ← L. Balance (x)
       else
8)         x ← Delete Node AVL        → (cancellazione del dato presente nel nodo)
9) return (x)
```

Delete Node AVL (x)

```
1) if x.sx = ⊥ then
2)     x ← Skip Right (x)
3) else if x.dx = ⊥ then
4)     x ← Skip Left (x)
   else
5)     x.dat ← Get & Delete Min AVL (x.dx, x)
6)     x ← Balance (x)
7) return x
```

Get & Delete Min AVL (x, p) → parent

```
1) if x.sx = ⊥ then
2)     d ← x.dat
3)     y ← skip 1 : get (x)
   else
4)     d ← Get & Delete Min AVL (x.sx, x)
5)     y ← x.balance (x)
6) swap child (p, x, y)
7) return d
```

CLASSE RED BLACK

AVL $h \approx 1.44 \log(n+2) - 0.328$ RB $h = 2 \log(n)$

Definizione RB Tees

1) ABR / BST

2a) Dati solo su nodi interni

2b) Le foglie (che non possono contenere dati/2a) sono identiche e assimilabili ad un nodo reale null

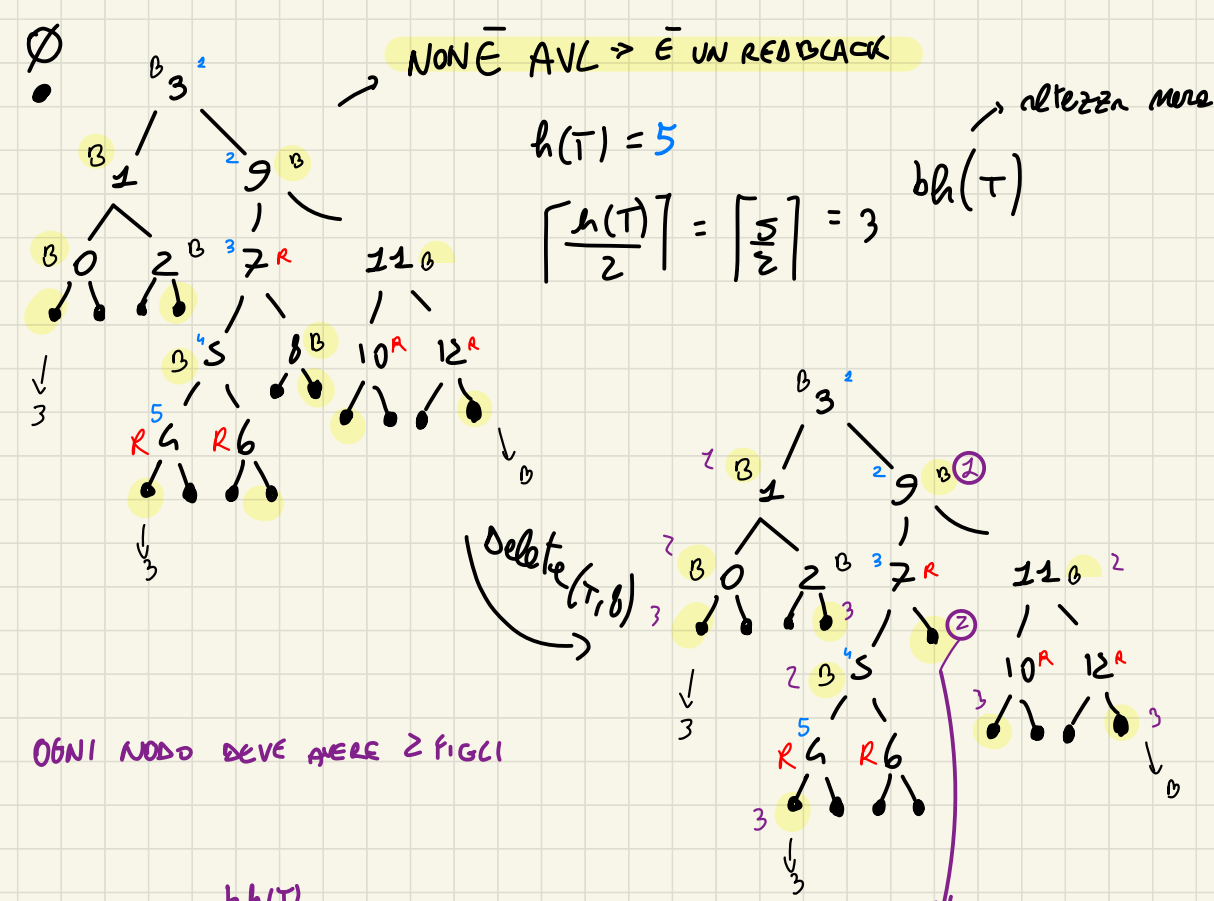
3a) Tutti i nodi hanno un colore rosso o nero

3b) Tutte le foglie sono nere

3c) Ogni nodo rosso non può avere figli rossi → tutti i figli sono neri

3d) $\forall x \in T . \forall \hat{\gamma} \in Path_T(x) \quad |\hat{\gamma}|_N = h$ → LUNGHEZZA NERA DEL PERCORSO $\hat{\gamma}$

↳ percorso in Tole x a una foglia dell'albero T_x



OGNI NODO DEVE AVERE 2 FIGLI

$IN(T) \geq 2^{bh(T)} - 1$

↓

INTERNAL NODE (NUMERO NODI INTERNI)

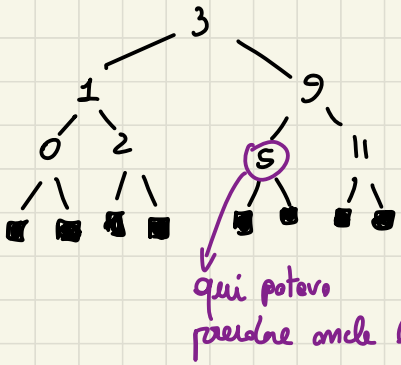
↓

SENZA CONSIDERARE LE FOGLIE

NON È PIÙ RED BLACK

↓

VIOLAZIONE DEL VINCOLO LEVANDO L'8 non ho più 3 NERI (Proprietà 3 d.)



→ ALBERO SOLO NERI (È UN MODO ASTRATTO PER CAPIRE CHE L'ALTEZZA DELL'ALBERO NERO CORRISPONDE AD UN ALBERO PIENO)

↓

E LA SUA ALTEZZA È $\leq \lceil \log(n+1) \rceil$

CI POSSONO ESSERE PIÙ ROSSI CHE NERI, MA NON ABBASTANZA DA CAMBIARE IL VALORE ASINTOTICO

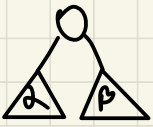
ADESSO DIMOSTRIAMO $IN(T) \geq 2^{bh(T)} - 1$ per induzione

Proof per Induzione sull'altezza dell'albero $h(T)$

Caso base $h(T) = 0$ $\{\blacksquare\}$ → $0 = \#IN(T) \geq 2^0 - 1 = 0$ ✓

$\blacksquare$ ↓ contiene solo 1 foglia e non altri

CASO INDUTTIVO $h(T) > 0$



$h(\alpha), h(\beta) \leq h(T) - 1 \Leftarrow h(T) = 1 + \max\{h(\alpha), h(\beta)\}$

PER IP. INDUTT. $\begin{cases} \#IN(\alpha) \geq 2^{bh(\alpha)} - 1 \\ \#IN(\beta) \geq 2^{bh(\beta)} - 1 \end{cases}$

$$\begin{cases} 1^\circ \text{ OSSERVAZIONE } \quad bh(\alpha) = bh(\beta) \quad \text{STESSO NUM. NODI} \\ 2^\circ \text{ oss.} \quad bh(T) \leq 1 + bh(\alpha) = 1 + bh(\beta) \end{cases}$$

$$\#IN(T) = 1 + \#IN(\alpha) + \#IN(\beta) \geq 1 + (2^{bh(\alpha)} - 1) + (2^{bh(\beta)} - 1) =$$

(1° oss.)

$$= 2 \cdot 2^{bh(\alpha)} - 1 = 2^{(bh(\alpha)+1)} - 1 \geq 2^{bh(T)} - 1$$

(1° oss.; 2^a monotona; 2° oss.)

$$\#IN(T) \geq 2^{bh(T)} - 1 \Rightarrow \#IN(T) + 1 \geq 2^{bh(T)} \Rightarrow bh(T) \leq \log(n+1)$$

(n)

$$h(T) \leq 2\, bh(T) \leq 2 \log(n+1) \Rightarrow h(T) = O(\log n)$$

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

$$h(T) = \Omega(\log n)$$

SE AGGIUNGO UN NODO ROSSO RISPETTO TUTTI I VINCOLI

↓

CON IL PADRE NERO

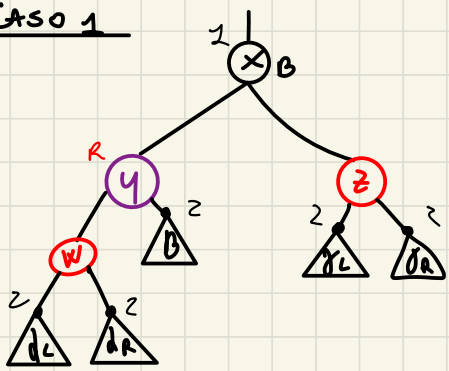
SE IL PADRE DEL NODO AGG. È ROSSO SI VIOLA IL 3.c)

↓

COME RISOLVERLO

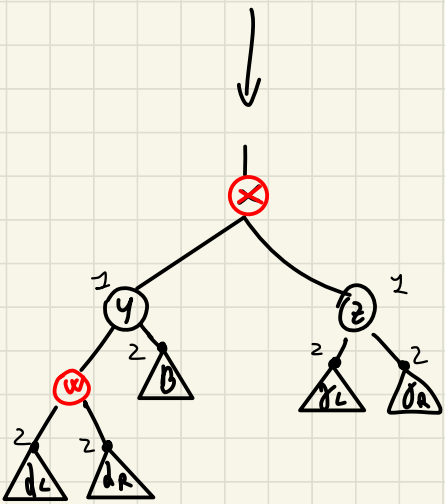
INSERIMENTO SI DIVIDE IN 3 CASI

CASO 1



○ → NODO CON PROBLEMI per 3c

→ SI CAMBIA COLORE



ADESSO IL PROBLEMA È RISOLTO

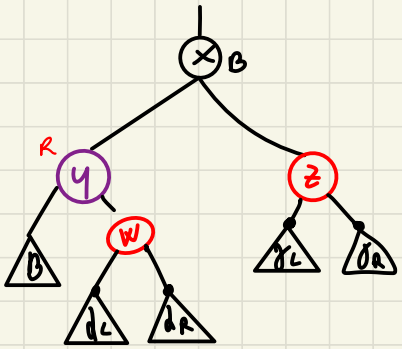
↓

VEDIAMO SE IC 3c È RISPETTATO

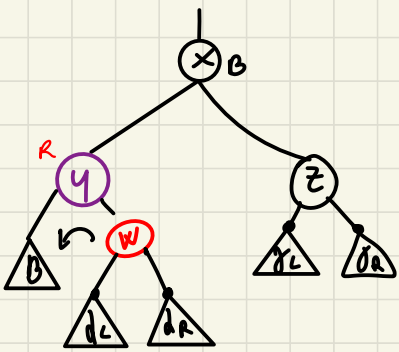
↓

IL NUMERO DEI NODI NERI RIMANE UGUALE (x NON È UNA RADICE, PERCIÒ LA PRENDIAMO)

CASO 1b



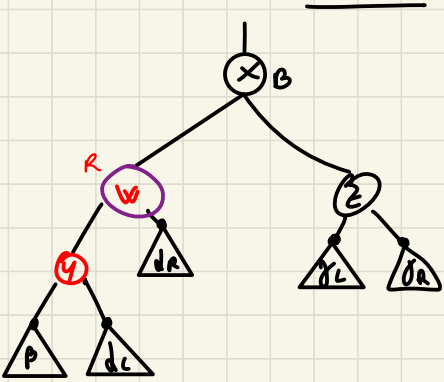
CASO 2



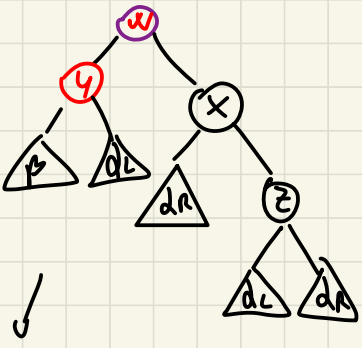
qui applichiamo le rotazioni:

RC - Rot(y)

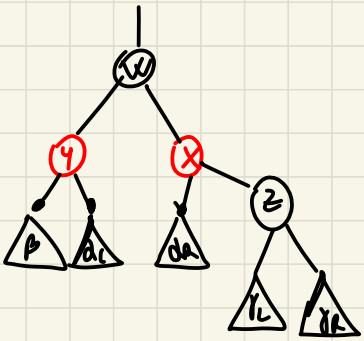
CASO 3



L - R. Rot(x)

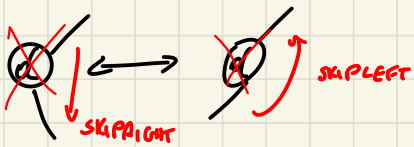


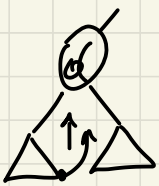
SI CAMBIANO I COLORI


w
y
x
z
β
αL
αR
γL
γR

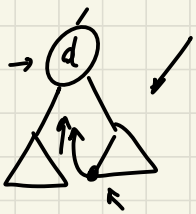
22 NOVEMBRE

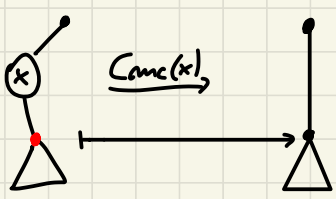
Delete (x, o(1)

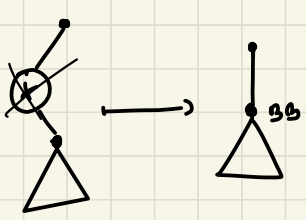




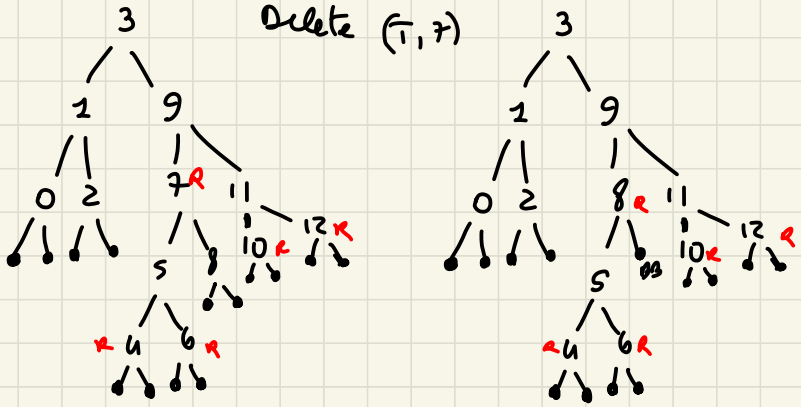
oppure



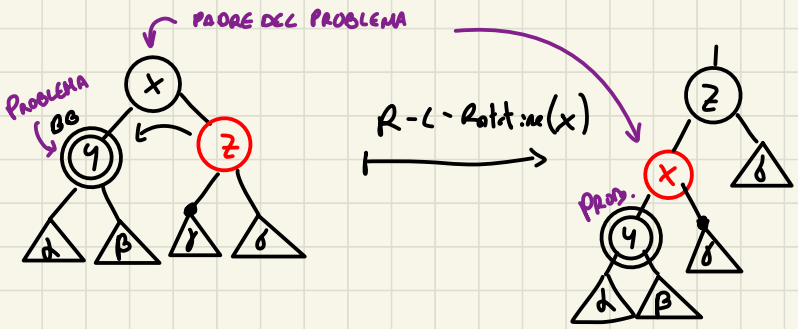




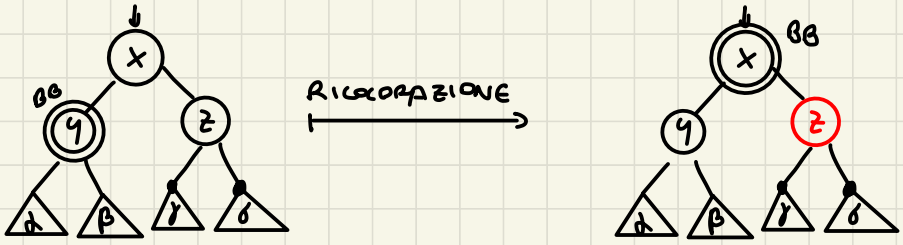
VIOLAZIONE, SI È PRESENTATO UN DOPPIO NERO


Delete (T, 7)

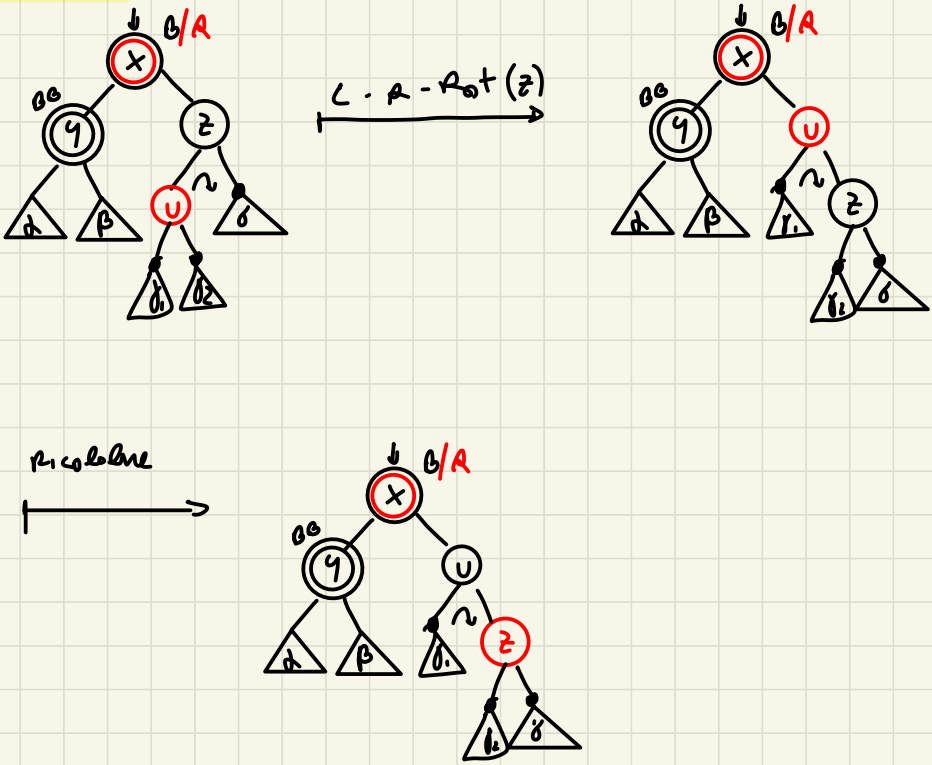
CASO 1


PADRE DEL PROBLEMA
PROBLEMA
BB
R-L-Rotation(x)
PROB.

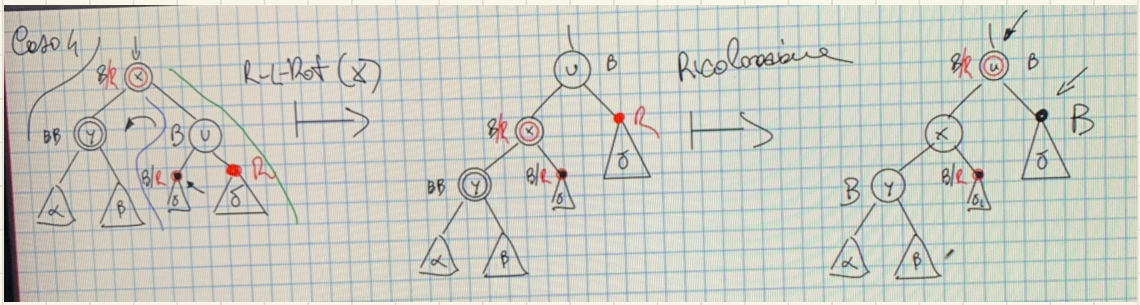
CASO 2

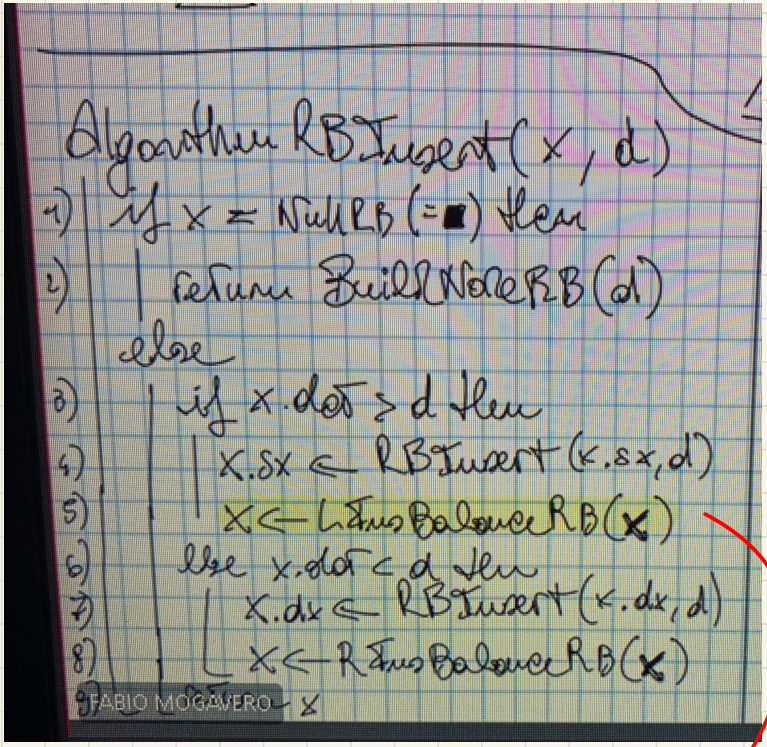

BB
RICOLORAZIONE
BB

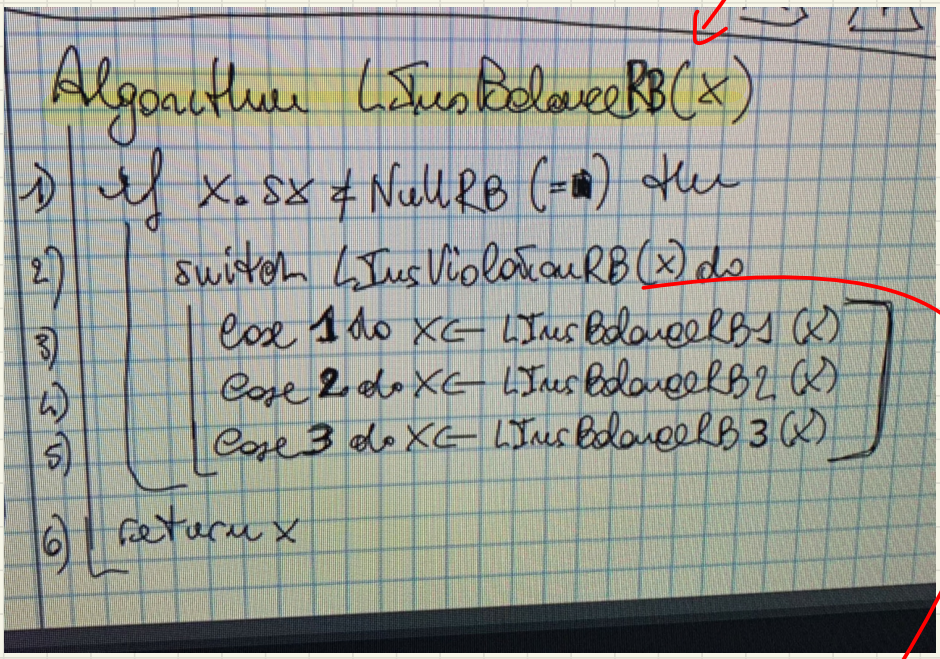
CASO 3

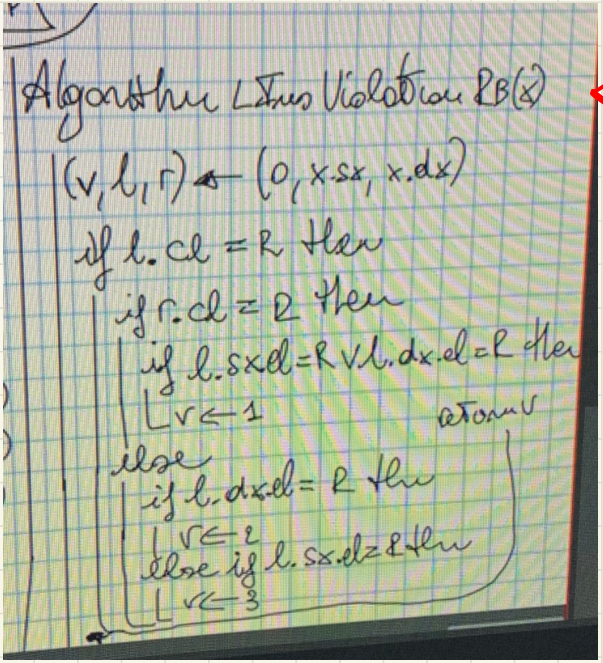

B/A
BB
L - R - Rot (z)
B/A
BB
Ricolorare
B/A
BB

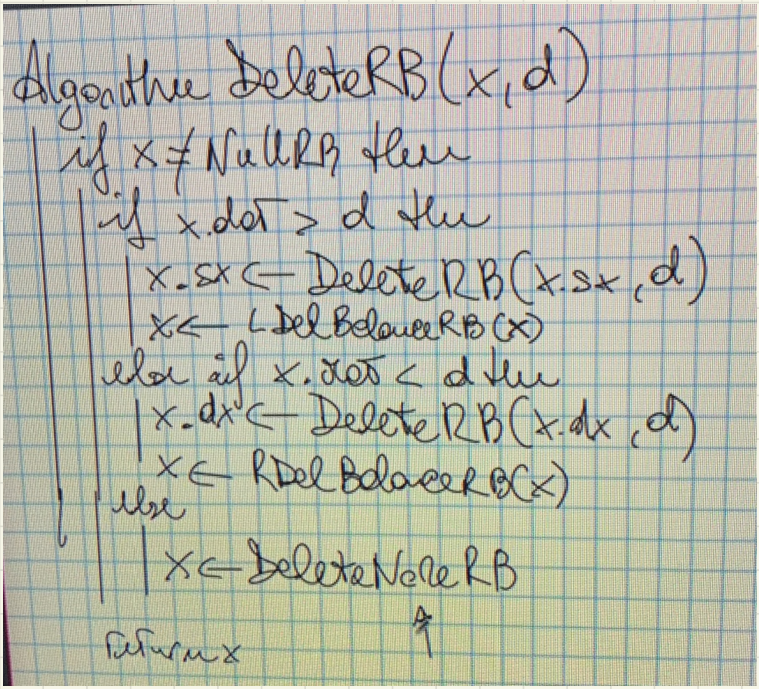
CASO 4


Caso 4
R-L-Rot (x)
Ricolorazione


Algorithm RBInsert(x, d)
1) if x = NullRB (=■) then
2) return BuildNodeRB(d)
else
3) if x.dat > d then
4) x.sx ← RBInsert(x.sx, d)
5) x ← LInsBalanceRB(x)
6) else x.dat < d then
7) x.dx ← RBInsert(x.dx, d)
8) x ← RInsBalanceRB(x)
FABIO MOGAVERO
x

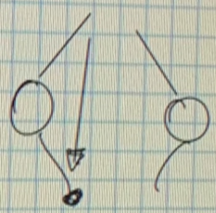

Algorithm LInsBalanceRB(x)
1) if x.sx ≠ NullRB (=■) then
2) switch LInsViolationRB(x) do
3) case 1 do x ← LInsBalanceRB1(x)
4) case 2 do x ← LInsBalanceRB2(x)
5) case 3 do x ← LInsBalanceRB3(x)
6) return x


Algorithm LRRB Violation RB(x)
(v, l, r) ← (0, x.sx, x.dx)
if l.cl = R then
if r.cl = R then
if l.sx.cl = R ∨ l.dx.cl = R then
v ← 1
else
if l.dx.cl = R then
v ← 2
else if l.sx.cl = R then
v ← 3
return v


Algorithm DeleteRB(x, d)
if x ≠ NullRB then
if x.dat > d then
x.sx ← DeleteRB(x.sx, d)
x ← LDelBalanceRB(x)
else if x.dat < d then
x.dx ← DeleteRB(x.dx, d)
x ← RDelBalanceRB(x)
else
x ← DeleteNodeRB
return x

Algorithm DeleteNodeRB(x)

```
if x.sx = NullRB then
    y ← SkipRightRB(x)
else x.dx = NullRB then
    y ← SkipLeftRB(x)
else
    x.dat ← GetAndDeleteMin(x.dx, x)
    x ← RDelBalanceRB(x)
return x
```



24 NOVEMBRE

Algoritm Get & Delete Min RB(x,p)

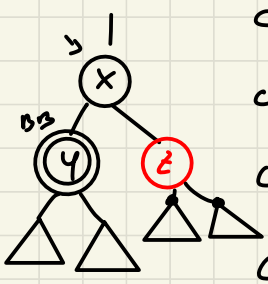
```
1) if x.sx = Null RB tree
2)     d ← x.dat
3)     y ← Skip Right RB(x) *
   else
4)     d ← Get DeleteMin RB(x.sx, x)
5)     y ← L. DelBalance RB(x) *
6) SWAP CHICA (p, x, y)
7) return d
```

Algoritm Skip Right RB (x)

```
1) if x.cl = B then Propagate Black RB(x.dx)
2) return Skip Right (x)
```

Algoritmo L_del BalanceRB(x)

```
1) if x.sx ≠ NullRB ∨ x.dx ≠ NullRB then
2)     switch L_del ViolationRB(x) do
3)         caso 1 :   x ← L_del BalanceRB1(x)
4)         caso 2 :   //
5)         caso 3 :   //
6)         caso 4 :   //
7) return x
```



CASO 1 $z \neq R$

CASO 2 $z = R \wedge z.sx = z.dx = B$

CASO 3 $z = B \wedge z.sx = R \wedge z.dx = B$

CASO 4 $z = B \wedge z.dx = R$

Algoritmo L_del ViolationRB(x)

```
1) (v, l, r) ← (0, x.sx, x.dx)
2) if l.col = BB then
3)     if r.col = R then
4)         v ← 1
       else
5)         if r.dx.col = R then
6)             v ← 4
```

```
7)      else if x.sx.cl = R then
8)          v ← 3
9)      else
            v ← 2
10)     return v
```

Algorithm RecMin (x)

```
1)  if x.sx = ⊥ then
2)      return x.dat
    else
3)      return RecMin(x.sx)
```

Algorithm Fold(L, F, a)

```
if L = ⊥ then
    return a
else
    a ← F(L.dat, a)
    return Fold(L.next, F, a)
```

```
WHILE L ≠ ⊥ do
    a ← F(L.dat, a)
    L ← L.next
return a
```

```
Algorithm PostFold (L, F, a)
    if L = ⊥ then
        return a
    else
        a ← postFold(L.next, F, a)
        return F(L.dat, a)
```

```
Algorithm PreDFV (x, F, a)
1)  if (x) = ⊥ then
2)      return a
    else
3)      a ← F.(x.dat, a)
4)      a ← PreDFV(x.sx, F, a)
5)      a ← PreDFV(x.dx, F, a)
6)      return a
```

Algorithm RecFun (x, i, j)

Stack $s_x, s_j, s_{a_L}, s_{z_L}, s_{h_R}$

1) $a \leftarrow Fini(i, j)$
2) if $x = \perp$ then
3) $\quad m \leftarrow F_\perp$ (
   else
4) $\quad (k_L, h_L) \leftarrow F_{pre}(x, a)$
5) $\quad z_L \leftarrow RecFun(x.sx, i, k_L)$
6) $\quad (k_R, h_R) \leftarrow F_{in}(x, a_L, z_L)$
7) $\quad z_R \leftarrow RecFun(x.sx, k_R, j)$
8) $\quad m \leftarrow F_{post}(x, h_L, z_L, h_R, z_R)$
9) $z \leftarrow F_{FIN}(m)$
10) return z

25 NOVEMBRE

quello di prima RICORSIVO IN ITERATIVO

Algoritmo ItToFun (x, i, J)

0) stack $S_x, S_J, S_{hL}, S_{ZL}, S_{hR}$ → BISOGNA SCRIVERLO
1) call ← T
2) While (call V ¬ IS EMPTY (S_x)) do
3) if call then
4) a ← F_{IN} (i, J)
5) if x = ⊥
6) m ← $F_{\perp}$ (a) } ret ← F_{FIN} ($F_{\perp}$ (a))
7) z ← F_{inn} (m)
8) ret ← z
9) (call, cont) ← (⊥, x)
else
10) (x_L, h_L) ← F_{pre} (x, a)
11) S_x ← Push (S_x, x)
12) S_J ← Push (S_J, J)
13) S_{hL} ← Push (S_{hL}, h_L)
14) (x, J) ← (x.S_x, x_L)

```
15)  else
         x ← Top(Sx)
16)      if Last = x.Sx then
17)          zL ← ret
18)          S ← Top(SS)
19)          hL ← Top(ShL)
20)          (kR, hR) ← FIN(x, hL, tL)
21)          if x.dx = ⊥ then
22)              zR ← FIN(FL(FIN(kR, S)))
23)              m ← FPost(x, hL, zR, hR, zR)
24)              ret ← FFIN(m)
25)              Last ← x
26)              (Sx, SS, ShL) ← (Pop(Sx), Pop(S), Pop(ShL))
27)          else
28)              SzL ← Push(SzL, zL)
29)              ShR ← Push(ShR, hR)
29)              (x, i) ← (x.dx, kR)
30)              call ← ⊤
31)      else
32)          zR ← ret
32)          hL ← Top & Pop(ShL)
```

33) $z_L \leftarrow$ top & pop (Sz_L)

34) $h_R \leftarrow$ top & pop (Sh_R)

35) $S_J \leftarrow$ top & pop (S_J)

36) $at \leftarrow F_{fim}(F_{part}(x, h_L, z_L, h_R, z_R)$

37) $(S_x, Root) \leftarrow (pop(S_x | x)$

38) Return at

29 NOVEMBRE

Esempio di ALGORITMI PreFold (x, F, a)

```
1) if x = ⊥ then
2)     return a
   else
3)     z ← F(x.dat, a)
4)     z ← PreFold(x.sx, F, z)
5)     z ← PreFold(x.dx, F, z)
6)     return z
```

Algorithm Its PreFold (x, F, a)

```
0) Stack Sx
1) call ← T
2) while (call ∨ ¬ IS EMPTY(Sx)) do
3)     if call then
4)         if x = ⊥ then
5)             ret ← a
6)             (call, last) ← (⊥, x)
           else
7)             z ← F(x.dat, a)
8)             Sx ← Push(Sx, x)
```

```
9)        (x, a) ← (x sx, z)
      else
10)     x ← Top(sx)
11)     if x.sx = Last Item
12)       z ← ret
13)       if x.olx ≠ Item
14)         (x, a) ← (x.olx, z)
15)         call ← T
          else
16)         ret ← z
17)         (sx, Last) ← Pop(sx), x)
        else
          z ← ret
          ret ← z
          (sx, Last) ← Pop(sx, x)
      Return ret
```

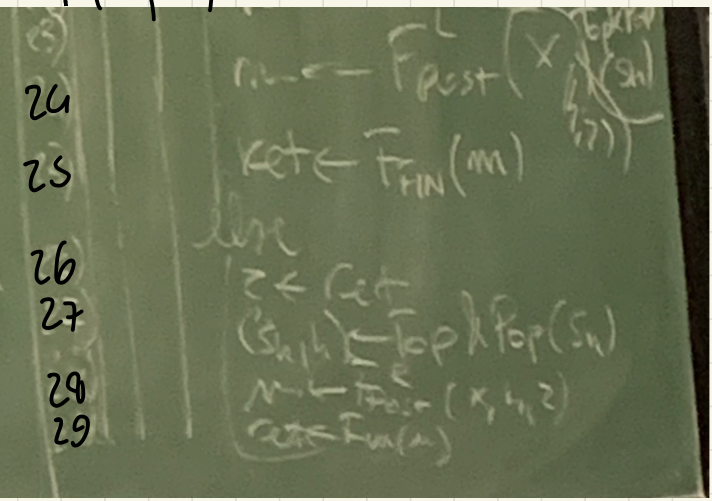
```
z ← PreFold(x.dx, F, 2)
        ↓
if x.olx = ⊥ then
    z ← a
else
    z ← PreFold(x.dx, F, z, t
    Return z
    if x.sx ≠ ⊥ then
        z ← PreFold(x.olx, F, z)
```

Algorithm RecFun (x, i, j)

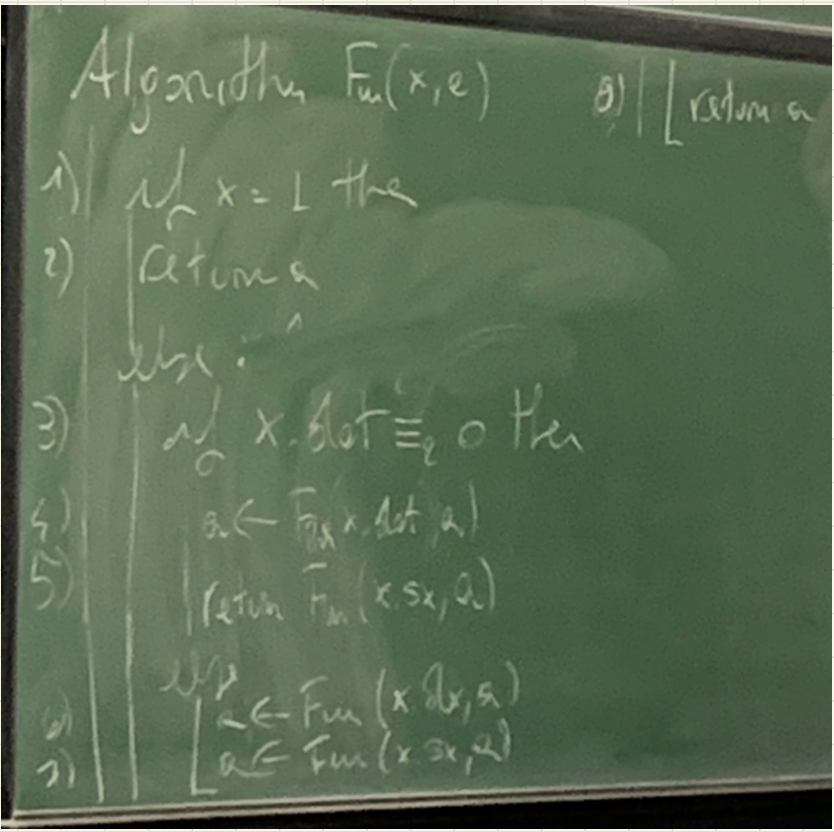
2) $a \leftarrow F_{imi}(i, j)$
2) if $x = \bot$ then
3) $m \leftarrow F_{\bot}(a)$
else
4) if $C_L(x)$ then
5) $(\hat{i}, \hat{j}, h) \leftarrow F^{L}_{pre}(x, a)$
6) $z \leftarrow RecFun(x.sx, \hat{i}, \hat{j})$
7) $m \leftarrow F_{post}(x, h, z)$
8) else if $C_R(x)$ then
9) $(\hat{i}, \hat{j}, h) \leftarrow F^{R}_{pre}(x, a)$
10) $z \leftarrow RecFun(x.dx, \hat{i}, \hat{j})$
11) $m \leftarrow F^{R}_{post}(x, h, z)$
else
12) $m \leftarrow F_{LOC}(x, a)$
13) return $F_{fin}(m)$

```
Algorithm IterFun(x, i, S)
0)  Stack Sx, Sh
1)  call ← T
2)  while (call V ¬ isEmpty(Sx)
3)    if call then
4)      a ← Fini(i, S)
5)      if x = ⊥ then
6)        m ← F⊥(a)
7)        res ← Fin(m)
8)        (call, ast) ← (⊥, x)
        else
9)        if Cc(x) then
10)         (î, ŝ, h) ← Fpre(x, a)
11)         (Sx, Sh) ← (Push(Sx, x), Push(Sh, h))
12)         (x, i, S) ← (x.cx, î, ŝ)
13)       else if CR(x) then
14)         (î, ŝ, h) ← F^R_pre(x, a)
15)         z ← IterFun(x.dx, î, ŝ)
16)         m ← F^R_post(x, h, z)
```

```
                                    else
17)                                     m ← F_Loc(x, e)
18)                                     ret ← F_FIN(m)
19)                                     (call, last) ← (⊥, x)
                                else
20)                                 (sk, x) ← Top & Pop(sk)
21                                  if CL(x) then
22                                      z ← ret
23                                      (sh, h) ← Top & Pop(sh)
24                                      m ← F^L_PART(x, h, z)
```




```
39) return ret
```


Algorithm Fun(x, e)
8) return a
1) if x = L then
2) return a
else:
3) if x.slot ≡ e 0 then
4) a ← Fun(x.slot, a)
5) return Fun(x.sx, a)
else
a) a ← Fun(x.dx, a)
7) a ← Fun(x.sx, a)

1 DICEMBRE

Algoritmo QuickSort (A, p, r)

```
1) if p < r then
2)     q ← Partition (A, p, r)
3)     QuickSort (A, p, q)
4)     QuickSort (A, q+1, r)
```

QUESTA È VERA ↑

2) $p \leq q < r$

VERSIONE ITERATIVA Algoritmo Itr QuickSort (A, p, r)

```
    stack Sr, Sq
1)  call ← T
2)  while (call V ¬ is empty (Sq)) do
3)      if call then
4)          if p < r then
5)              q ← Partition (A, p, r)
6)              (Sq, Sr) ← (Push (Sq, q), Push (Sr, r))
7)              r ← q
            else
8)              (call, ast) ← (⊥, r)
        else
9)          q ← Top (Sq)
```

copia codice

→ ripristino

NOTE:

LA RETURN, ANCHE SE NON C'È, BISOGNA SIMULARLA LO STESSO

il Qsort SI USA IN RISALITA

```
10)   if q = last then        → SE VERO, STIAMO TORNANDO ALLA PRIMA
11)       sr ← Top(Sr)
12)       p ← q+1             ADESSO SIAMO IN RISALITA → CALL DA FeT
13)       call ← T
      else
14)       (Sq, Sr) ← (Pop(Sq), Top(Pop(Se)))
15)       last ← r
```

Algorithm Find(A, d, p, r)

```
1)  if r < p then
2)      return ⊥
    else
3)      q ← ⌊(p+r)/2⌋
4)      if A[q] = d then
5)          return T
        else
6)          d ← Find(A, d, p, q-1)
7)          d ← d ∨ Find(A, d, q+1, r)
8)          return d
```

VERSIONE ITERATIVA Algorithm 12 Find (A, d, p, r)

```
   Stack Sr, Sq
1) call ← T
2) while (call v ¬ is Empty (Sr)) do
3)     if call then
4)         if r < p then
5)             ret ← ⊥
6)             (call, last)
           else
7)             q ← ⌊(p+r)/2⌋
8)             if A[q] = d then
9)                 ret ← T
10)                (call, last) ← (⊥, r)
               else
11)                (Sr, Sq) ← (Push(Sr, r), Push(Sq, q))
12)                r ← (q - 1)
       else
13)        r ← Top(Sr)
14)        if last ≠ r then
15)            z ← ret
16)            q ← Top(Sq)
```

17) $s_d \leftarrow Push\ (s_2, d)$

18) $p \leftarrow (q+1)$

19) $call \leftarrow T$ ↳ riga 6 e 7

else

20) $d \leftarrow Top\ \&\ Pop\ (s_2)$

21) $d \leftarrow d \vee ret$

22) $ret \leftarrow d$

23) $(s_1, (s_2, r)) \leftarrow (Pop\ (s_1), Top\ \&\ Pop\ (s_2))$

24) $last \leftarrow r$

25) return ret

GRAFI

SONO UN INSIEME

$G = \langle V, E \rangle$

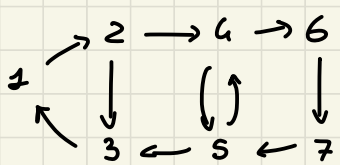
V → INSIEME

E → RELAZIONE BINARIA $E \subseteq V \times V$

↓ OVVERO RELAZIONE TRA 2 NODI

$\uparrow\downarrow = \updownarrow$

Es.



$(\{1,2,\ldots,7\}, \{(1,2),(2,3),(2,4),(4,5),(5,4),(4,6),(6,7),(7,5),(5,3),(3,1)\})$

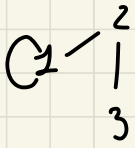
1.

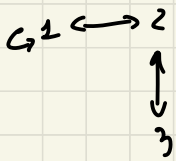
$G = \langle V, E \rangle \quad E \in \binom{V}{2}$

$E \in 2^V \quad |E| \leq 2$

2.

$\langle \{1,2,3\}, \{\{1\}, \{1,2\}, \{2,3\}\} \rangle$ → NON ORIENTATO MATEMATICAMENTE

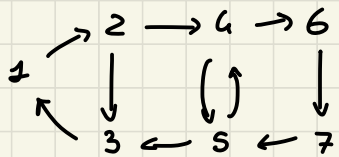




$\langle \{1,2,3\}, \{(1,1),(1,2),(2,1),(2,3),(3,2)\} \rangle$

↳ ORIENTATO

Es. di prima



$\langle \{1,2,\ldots,7\}, \{(1,2),(2,3),(2,4),$
$(4,5),(5,4),(4,6),$
$(6,7),(7,5),(5,3)$
$(3,1)\} \rangle$

2 NOVEMBRE

2 NODI SONO ADIACENTI SE ESISTE UN ARCO TRA DI LORO

SEQ. FINITE SEQ. INFINITE

$\tilde{\pi} \in Path(G) \subset V^{+} \cup V^{\omega}$

$\forall i \in [0, |\tilde{\pi}|) \quad ((\tilde{\pi})_i, (\tilde{\pi})_{i+1}) \in E$ → SE LA VEDO COME COPPIA DEVE ESSERE UN ARCO

$\pi \in Cycle(G) \subseteq Path(G)$

$(\tilde{\pi})_0 = (\tilde{\pi})_{|\tilde{\pi}|-1}$

$\int st(\tilde{\pi}) \quad last(\tilde{\pi})$

$\tilde{\pi} \in SimplePath(G) \subseteq Path(G) \qquad \forall i, j \in [0, |\tilde{\pi}|) \quad (\tilde{\pi})_i \neq (\tilde{\pi})_j$ $i \neq j$

$SPath(G)$

UN GRAFO SI DICE CONNESSO QUANDO DA UN VERTICE POSSO ANDARE AD UN ALTRO e ←

È FORTEMENTE CONNESSO QUANDO È ANCHE ORIENTATO

NON ORIENTATO → O CI SONO TUTTE LE COMBINAZIONI DI FRECCE
O NON HA FRECCE

COMPONENTE MASSIMALE CONNESSA

È UN SOTTO GRAFO INDOTTO CONNESSO CHE NON È CONTENUTO STRETT. IN UN ALTRA COMPONENTE CONNESSA PIÙ GRANDE

RELAZIONE DI EQUIVALENZA

$$\mathcal{G} = \langle V, E \rangle$$

$\subseteq$ → INCLUSIONE DI SOTTOGRAFO

$$\mathcal{G}' = \langle V', E' \rangle$$

$$V' \subseteq V \wedge E' \subseteq E$$

$\subseteq_I$ → INDOTTO

$$\mathcal{G}' \subseteq_I \mathcal{G} \iff V' \subseteq V \wedge E' = E \cap (V' \times V')$$

CHIUSURA TRANSITIVA e RIFLESSIVA

$R^* \subseteq D \times D \qquad R^i \qquad R^0 = \{(d, d) \mid (d, d) \in D \times D\} \qquad R^{i+1} = R \circ R^i$

$R^1 \triangleq R$

$R_1, R_2 \subseteq D \times D \qquad R_1 \circ R_2 \triangleq \{(d_1, d_2) \in D \times D \mid \exists d \in D.\ (d_1, d) \in R_1 \wedge (d, d_2) \in R_2\}$

$$R^* \triangleq \bigcup_{i \geq 0} R^i$$

6 DICEMBRE

- $\begin{cases} Reach(G) \triangleq \epsilon^* \\ Reach^+(G) \triangleq \epsilon^+ \end{cases}$ $\forall v,u \in V \; (v,u \in Reach(G) \iff \exists \tilde{\imath} \in Path(G)$

$fst(\tilde{\imath}) = v \wedge last(\tilde{\imath}) = u$

DISTANZA TRA VERTICI DI UN GRAFO

$$\delta(v,u) \triangleq \min_{\tilde{\imath} \in Path(v,u)} (|\tilde{\imath}| - 1)$$

BIANCO : ? Scoperto

GRIGIO : Scoperto $\wedge$? VISITATO

NERO : VISITATO

```
Algorithm BFS (G, v)
1)  Init (G, c)
2)  (Q, c(v)) ← (Enqueue (Q, v), gr)
3)  while (Q is Empty (Q)) do
4)      (Q, v) ← Head & Dequeue (Q)
5)      for each u ∈ Adj[v] do
6)          if c(u) = bn
7)              (Q, c(u)) ← Enqueue (Q, u), gr)
8)      c(v) ← nr
```

```
Algorithm Init (G, c)
1)  for ever v ∈ V do
2)      c(v) ← bn
```

```
Algorithm BFS (G, v)
1)  (c, d, p) ← Init (G)
2)  (Q, c(v), d(v)) ← (singleton Queue (v), gr, 0)
3)  while Q is Empty (Q) do
4)      (Q, v) ← Head & Dequeue (Q)
5)      for each u ∈ Adj[v] do
```

```
Algorithm Init (G)
1)  for each v ∈ V do
2)      (c(v), d(v), p(v)) ← (bn, ∞, ⊥)
3)  return (c, d, p)
```

6) if c(u) = bm then

7) (Q, c(u), d(u), p(u)) ← (Enqueue(Q, u), gr, d(v) + 1, v)

8) c(v) ← mz

return (c, d, p)

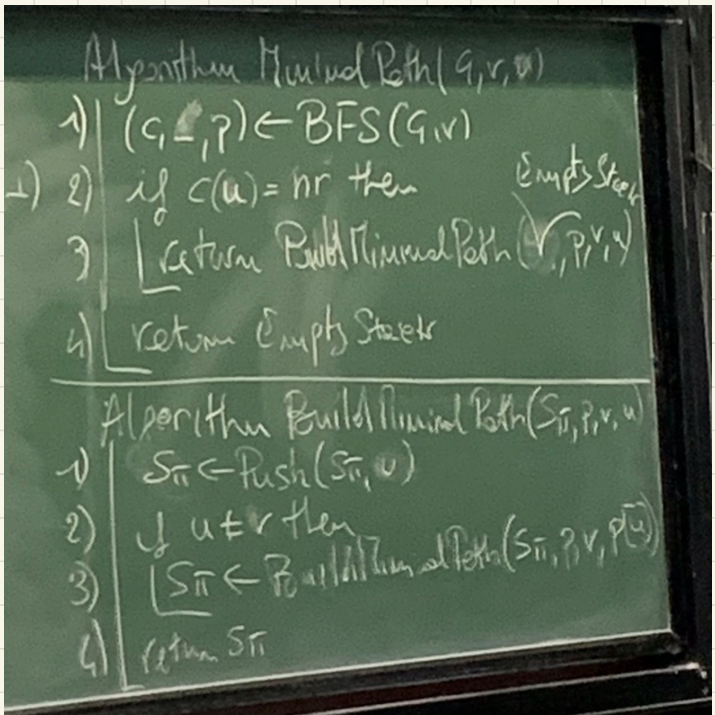
Algorithm MinimalPath(G, v, u)

1) (c, _, p) ← BFS(G, v)

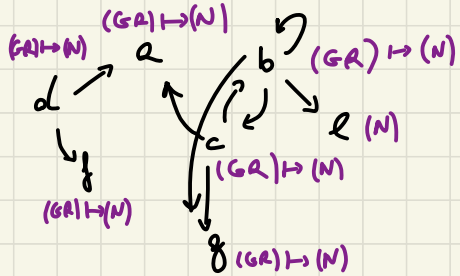
2) if c(u) = mz then

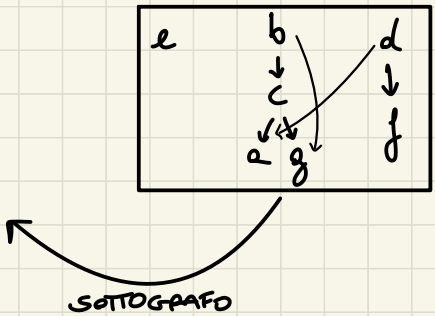
3) return BuildMinimalPath(Empty stack, p, v, u)

4) return Empty Stack



13 DICEMBRE





ALGORITMO DFS (G)

```
1) (c, p, t) ← Init(G)
2) for each v ∈ V do
3)     if c[v] = bn then
4)         (c, p, d, f, t) ← DFS(G, v, c, p, d, f, t)
```

Algoritmo DFS (G, v, c, p, d, f, t)

```
1) (c[v], d[v], t) ← (GR, t, t+1)        (c[v], d[v]) ← (gr, t)   t ← t+1
2) for each w ∈ Adj[v] do
3)     if c[w] = bn
4)         p[w] ← v
5)         (c, p, d, f, t) ← DFS(G, w, c, p, d, f, t)
6) (c[v], f[v], t) ← (N, t, t+1)
```

return (c, p, d, f, t)

TEOREMA DELLA STRUTTURA A PARENTESI DELLE DFS

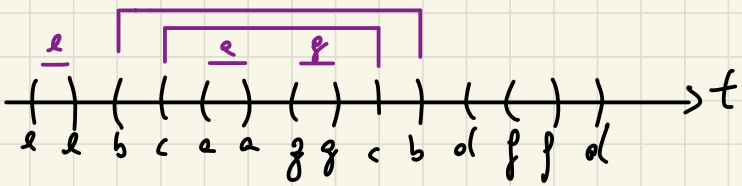
$\forall v, w \in V, v \neq w$ UNA DELLE SEGUENTI PROPRIETÀ È VERA

1a) $d[v] < d[w] < f[w] < f[v]$

1b) $d[w] < d[v] < f[v] < f[w]$

2a) $d[v] < f[v] < d[w] < f[w]$

2b) $d[w] < f[w] < d[v] < f[v]$



15 DICEMBRE

DFS, VISITA IN PROFONDITÀ

IN UN NODO IN UN PERCORSO ACICLICO LO POSSO VISITARE TANTE VOLTE QUANTI SONO GLI ARCHI ENTRANTI

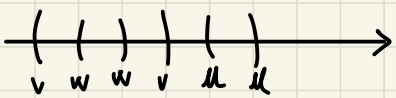
IL GRIGIO PERMETTE DI IDENTIFICARE I CICLI

PER LA TERMINAZIONE SONO SUFFICIENTI IL BIANCO E IL GRIGIO/NERO

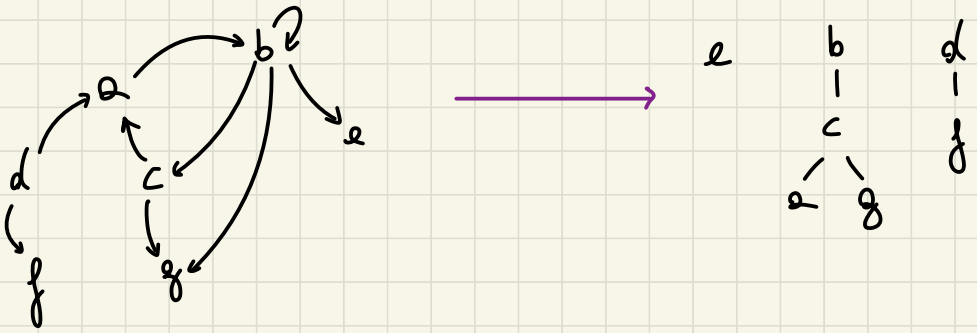
É FONDAMENTALE COLORARE IL MOMENTO DI INIZIO VISITA

LEMMA SULLA FORESTA DI VISITA

$$\mathcal{G} = \langle V, \{(p[v], v) \mid v \in V, p[v] \neq \perp\} \rangle$$



FORESTA DI VISITA DEL GRAFO



TEOREMA PERCORSO BIANCO

$\forall v, w \in V$ LE SEGUENTI CONDIZIONI SONO EQUIVALENTI

1) w SIA DISCENDENTE DI v IN $\mathcal{F}$ (OSSIA $\exists \pi \in Path(\mathcal{F}, v), last(\pi) = w$)

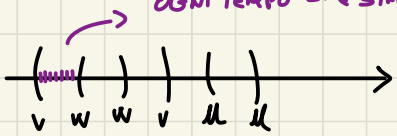
2) al tempo $d[v]$ $\exists \hat{\pi} \in Path(\mathcal{F}, v), last(\hat{\pi}) = w, \forall i \in (0, |\hat{\pi}|]$

$c(\hat{\pi}|_i) = bm$

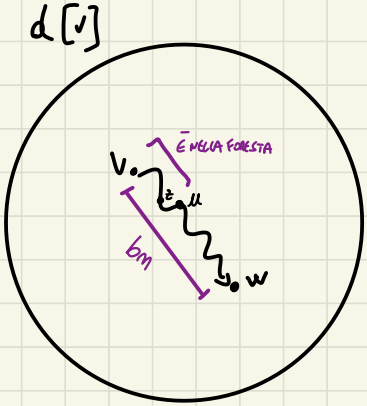
↳ COROLLARIO DEL LEMMA

DIMOSTRAZIONE

"1 $\Rightarrow$ 2"



"2 $\Rightarrow$ 1" X ASSURDO (SUPPONIAMO 2 VERO e 1 FALSO)



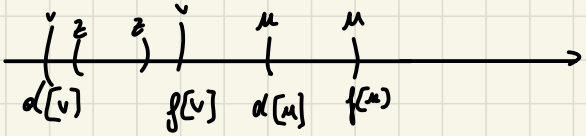
SE 1 È FALSA, LUNGO IL PERCORSO CI DEVE ESSERE UN NODO "u" CHE NON STA NELLA FORESTA

↓

u SARÀ ALMENO UN PASSO DOPO v

$u \neq v$

u X ASSURDO NON È DISCENDENTE DELLA FORESTA CADE LA "1" PERCHÉ ERA UN "$\Leftrightarrow$" SE E SOLO SE



PER IL LEMMA DELL'ALTRA VOLTA

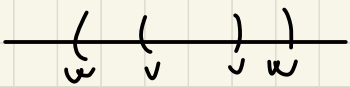
MA z È DISCENDENTE E AVREBBE DOVUTO ATTRAVERSARE u, QUINDI:

$$2 \Rightarrow 1 \quad \checkmark$$

Algoritmo DI VISITA DI PROFONDITÀ (Acyclic (G))

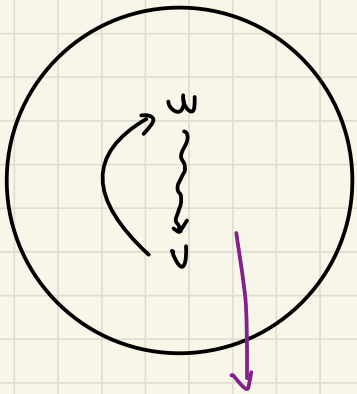
```
1) c ← Init(G)        SBIANCA IL GRAFO
2) for each v ∈ V do
3)     if c[v] = bn then
4)         acyclic ← Acyclic(G, v)      VARIABILE BOOLEANA
5)         if ¬ acyclic then
6)             return 1
7)     return T
```

Algoritmo Acyclic (G, v)




```
1) c[v] ← GR
2) for each w ∈ Adj[v] do
3)     if c[w] = bn then
4)         acyclic ← Acyclic (G, w)
5)         if ¬ acyclic then
6)             return ⊥
7)     else if c[w] = GR then
8)         return ⊥
9) return ⊤
```

$d[w] < d[v]$



C'È UN CICLO, QUINDI NON PUÒ ESSERE ACYCLIC

$c[v] \leftarrow NR$

PRIMA DI OGNI return $c[v]$ LO SETTO A NERO

È $O(|G|)$

ORDINAMENTO TOPOLOGICO

def.
SIA G UN GRAFO

SEQUENZA DI VERTICI

UN ORDINAMENTO TOPOLOGICO PER G È UNA PERMUTAZIONE ($\pi \in Perm(V)$) DEI NODI DI G : $\forall (v,w) \in E$ v COMPARE PRIMA DI w IN π ASSIA
$\exists i,j \in [0,n) : [\pi]_i = v \, , [\pi]_j = w$ e $i < j$

9

CI PUÒ ESSERE UN ORDINAMENTO TOPOLOGICO SE È CICLICO.

NO

TEOREMA

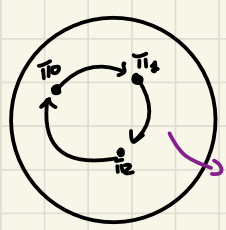
SIA G UN GRAFO FINITO 1 2

G È ACICLICO $\iff$ ESISTE UN ORDINAMENTO TOPOLOGICO PER G

dim.

"$\Leftarrow$" X ASSURDO

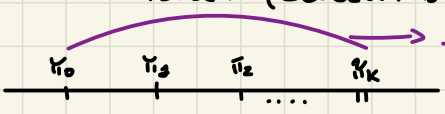
G



$\tilde{\pi}$ = CICLO SEMPLICE

GRAFO COMPLETO → clique

$\hat{\pi}$ → ORD. TOPOL. (SEQUENZA DI VERTICI)



→ SONO UGUALI → IN UNA PERMUTAZIONE OGNI VERTICE PUÒ ESSERE RIPETUTO 1 SOLA VOLTA

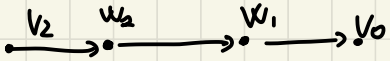
dim. X ASSURDO ←

"$\Rightarrow$"

$\left.\begin{array}{l} G \\ \text{aciclico} \\ \wedge \\ G \\ \text{finito} \end{array}\right\} \Rightarrow \exists v \in V \,.\, \forall w \in V \quad (w, v) \notin E$

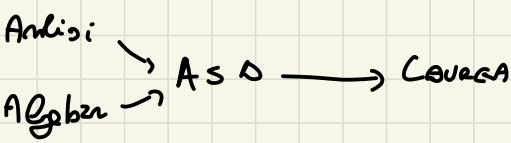
$\exists_v$ SENZA ARCHI ENTRANTI

LEMMA → SUPPONIAMO x ASS. CHE NON ESISTE NODO SENZA ARCO ENTRANTE



NON PUÒ CAPITARE CHE TUTTI I NODI ABBIANO ARCHI ENTRANTI

LEMMA → SE G È ACICLICO $G' \subseteq G \Rightarrow G'$ ACICLICO



Analisi , Algebra , ASD , LAUREA