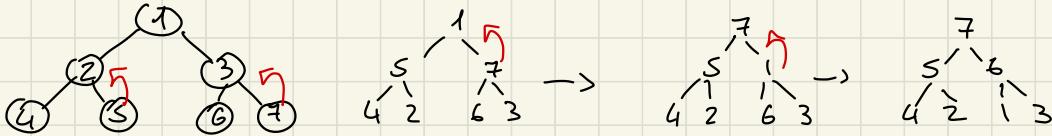




Esercizio da esame: (1-1)

Si inseriscono le chiavi $\{1, 2, \dots, 7\}$ in un max heap inizialmente vuoto, nell'ordine dato. Dire se è vero, giustificando: i 4 elem. più piccoli sono nelle foglie dell'heap.



È vero che i 4 elem. più piccoli stanno nelle foglie, ma solo dopo aver bilanciato l'heap. Nell'ordine dato, no.

Un heap di N nodi ha complessità $O(n)$

Esercizio esame (3-1)

Spiegare il concetto di grandezza di una funzione precisando il significato di $O(f(n))$ e $\Omega(f(n))$

La complessità di un algoritmo viene espressa sotto forma di funzione e rappresentata su un grafico come tale: questo si fa per avere un'idea concreta della complessità.

Il significato di $O(f(n))$ è "di ordine non superiore a" ed indica il limite assintotico superiore di una funzione. $O(f(n)) \leq c \cdot g(n)$ con $c > 0 \in \mathbb{R}$ date due funz $f(n)$ e $g(n)$.

$\Omega(n)$ significa "non inferiore a" e $\Omega \leq c \cdot g(n) \leq f(n)$

Es es 1) dire quale dei seguenti algoritmi è stabile e ne lavora in loco: quick, merge, heap, counting.

Quicksort: non è stabile, ne lavora in loco visto che esegue swap di elementi dentro alla struttura dati di input.

Mergesort: è stabile, NON è in loco visto che richiede $O(n)$ spazio aggiuntivo per eseguire il merge.

Heapsort: non è stabile, ne lavora in loco perché sviluppa le operazioni dentro alle strutture dati.

Counting Sort: è stabile, ma NON è in loco perché lo bisognerebbe usare più array per svolgere le operazioni.

Es es 2) Valutare il numero di scambi fatti da 1) bubble sort e 2) selection sort nel caso ottimo e nel caso peggiore.

1) Il bubble sort ordina due elenchi contemporaneamente e li ordina se c'è bisogno, poi si muove in avanti e ripete.
Nel caso migliore ha complessità $O(n)$ ed esegue $O(1)$ scambi.
Nel caso peggiore ha complessità $O(n^2)$ ed esegue $O(n^2)$ scambi.

2) Il selection sort scorre un array e seleziona il primo el. key, continua a scorrere e se ne trova uno più piccolo lo imposta come key. Quando non c'è più uno più piccolo lo mette al primo posto e procede così fino ad ordinare l'array.
Nel caso migliore ha complessità $O(n^2)$ ed esegue $O(n)$ scambi.
Nel caso peggiore ha complessità $O(n^2)$ ed esegue $O(n)$ scambi.

Es es 3) Quale è la complessità del quick sort quando viene fatto operare su vettori di elementi distinti ordinati in modo crescente? $O(n^2)$, $\Theta(n^2)$, $O(n \log n)$?

Il quicksort è un algoritmo divide et impera: divide l'array in due sub array, gli elementi bassi e quelli alti.

Dopo di che ordina l'array ricorsivamente:

- 1) seleziona un valore pivot
- 2) riordina l'array di modo che gli elementi minori del pivot vadano prima e quelli maggiori dopo. Finito il procedimento il pivot è al posto giusto.
- 3) Ripete ricorsivamente ad altri sub array.

In questo caso la complessità sarebbe quello del caso medio e ottimale $\rightarrow O(n \log n)$ e non quella peggiore $O(n^2)$.

Es es 4) Determinare l'ordine di grandezza del costo in tempo in termini di operazioni aritmetiche della seguente f.

```
int pippo (int n) { → c
    int i; → c
    if (n <= 3) return 1;   c(1) + (1 + O(1)) = c(1)
    else if (n > 333) {   ↓
        i = n/2; → log n
        return 3 * pippo (i) + pippo (i) + n * i; ((1))
    else return pippo (n-3) + 9;   c(1)
    }
}
```

$O(c) \cdot O(\log n) \rightarrow O(\log n)$

$$\begin{cases} 1 & n \leq 3 \\ T(n-3) & 3 < n \leq 333 \\ 2T(n/2) & n > 333 \end{cases}$$

Es relazione di ricorrenza:

Assumiamo che n sia una potenza di 3

$$T(n) = \begin{cases} 1 & \text{se } n=1 \\ 9 \cdot T(n/3) + n & \text{se } n>1 \end{cases}$$

$$T(n) \rightarrow 9T(n/3) + n \quad n \text{ potenza 3}$$

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

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

$$9T\left(\frac{n}{9}\right) = n + 9\left(\frac{n}{27}\right)$$

$$\text{Quindi } n + 9T\left(\frac{n}{3}\right) = 2n + 27T\left(\frac{n}{9}\right) = 3n + 81T\left(\frac{n}{27}\right) \dots =$$

$$= i \cdot n + 9^i T\left(\frac{n}{3^i}\right) \text{ bisogna iterare l'indice } \frac{n}{3^i} = 1$$

$$\frac{n}{3^i} = 1 \rightarrow n = 3^i \rightarrow i = \log_3 n \text{ volte}$$

$$\text{Quindi } T(n) = 9^{\log_3 n} + \log_3 n \cdot n =$$

? ✓

$$= 9^{\log_3 n} + \log_3 n^2 = \log_3 n^2 = n^2$$

ma comunque

$$\log_3 n = \log_9 n \cdot \log_3 9 \rightarrow 9^{\log_9 n \cdot \log_3 9} = n^2 \rightarrow \text{quindi } \Theta(n^2)$$

$$9^{\log_9(x) \cdot \log_3 9} \rightarrow 9^{\log_9(x) \cdot \log_3 3^2} \text{ usando } \log_a(a^x) = x$$

$$9^{\log_9(x) \cdot 2} \rightarrow 9^{2 \log_9 x} \rightarrow 9^{\log_9 x^2} = \cancel{9^{\log_9 x^2}} \rightarrow x^2$$

Es 3-2) Dato un intero k e un array v di interi positivi ordinati, si proponi un algoritmo efficiente che verifichi se esistono indici i ; tali che $v[j] - v[i] = k$. Analizzare il costo dell'algor.

→ :

1	2	3	4	5	6	7	8	9	10	11
i	j	i				j				
$i+1$			$v[j] - v[i] = k$							

int trova_k(v[], int k) {

for ($i=0, i \leq v[\text{dim}]$) {

 for ($j=i+1, j \leq v[\text{dim}]; j++$) { → ha costo $O(n^2)$

 if ($v[j] - v[i] == k$) printf "trovato";

 return $v[j]$; return $v[i]$; return k ;

 else $i++$;

}

}

Es 3-3) Siano date le rel. di ricorrenza

con $H(1) = \Theta(1)$, $W(1) = \Theta(1) \rightarrow$ trovare l'ordine

asintotico di $A(n) = H(n) \cdot W(n)$

quindi $\Theta(1)$ per $n=1$

$H(n) \left\{ \begin{array}{l} 2W\left(\frac{n}{8}\right) + \Theta(1) \text{ per } n > 1 \\ 2W(n) + \Theta(1) \text{ altrimenti} \end{array} \right.$

$2W\left(\frac{n}{8}\right)$ è come $2T\left(\frac{n}{8}\right) \rightarrow$ iterazione binaria $\frac{n}{8^i} = 1$

$$\log_8 n = \log_{64} n \cdot \log_8 64$$

$$2^i = \log_8 n \quad A(n) = 2^{\log_8 n} \quad n = 2^{3i} \quad \downarrow \text{volte}$$

$$2^{\log_8 n} = 2^{\log_{64} n \cdot \log_8 64} \xrightarrow{2^{\log_a(b^x)} = b^{\log_a b}} = 2^{\log_{64} n \cdot 2} = 2^2 \log_{64} n = 2 \log_{64} n^2 = 2 \log_{64} n^2 = \sqrt[3]{n^{1/3}}$$

$$\begin{cases} H(n) = W(n/2) \\ W(n) = 2H(n/4) + \Theta(1) \end{cases}$$

$$\begin{aligned} \Delta W(n) &= 2W\left(\frac{n}{8}\right) + \Theta(1) \\ &= 2W\left(\frac{n}{8}\right) + \Theta(1) \end{aligned}$$

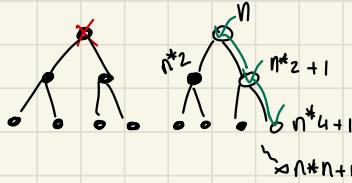
$$\begin{aligned} / \quad \frac{n}{8^i} &= 1 \rightarrow n = 8^i \\ \hookrightarrow i &= \frac{1}{3} \log_2 \end{aligned}$$

volte

↓

$\Theta(n^{1/3})$

Esercizio 3-4) T = albero binario contenente valori interi, rappresentato con puntatori ai figli. Progettare un alg. che verifichi se esiste in T una comune radice popola di soli nodi contenenti il valore 0. Si calcoli la complessità di tale algoritmo.



che complessità \rightarrow
 $O(\log n)$

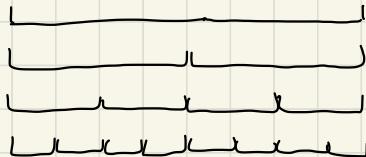
```
void searchPath(int arr[], int n) {
    int i = 1;
    if (arr[i] != 0) return false;
    while (i <= n) {
        if ((i * 2 <= n) && (arr[i * 2] == 0))
            i = i * 2;
        else if ((i * 2 + 1 <= n) && (arr[i * 2 + 1] == 0))
            i = i * 2 + 1;
        else return false;
    }
    return true;
}
```

Esercizio 6) Date una sequenza di n interi, ci interessano le coppie (a_i, a_{i+1}) con $a_{i+1} = a_i + 1$. Ad esempio, nella seq $\{1, 3, 4, 5, 13, 14, 14, 7, 7, 8, 7\}$ le coppie di interesse sono $(3, 4), (4, 5), (13, 14), (7, 8)$. Scrivere un alg. divide et impera per calcolare il numero di coppie di questo tipo presenti nella sequenza. Calcolarne la complessità.

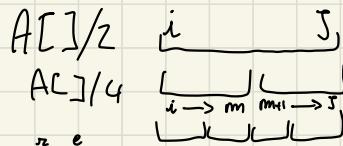
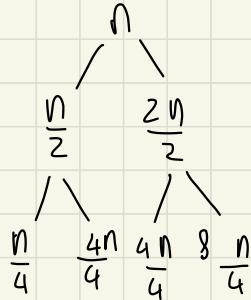
```
void countsucc(A[], i, j) {
    if (i < j) {
        m = (m - j) / 2;
        count(A, i, m);
        count(A, j, m);
        succ(A, i, j + i, m);
        for (i + 1 → j)
            if (a[i] == a[i - 1] - 1)
                succ++;
        succ++;
    }
    return succ;
}
```

$A[1, 2, 4, 6, 7 \dots]$

$O(\log n)$



Ninide et impresa



```
void findcouple(A[], i, j){  
    if(j > i){  
        m = (i+j)/2  
        cercasucc(A[], i, m){  
            for(i=0; i < m; i++){  
                if(A[i] == A[i+1]){  
                    return A[i], A[i+1];  
                }  
            }  
            cercasucc(A[], m+1, j){  
                for(m+1; m+1 < j, m+1++){  
                    if(A[m+1] == A[m+1+1]){  
                        return A[m+1], A[m+2];  
                    }  
                }  
            }  
        }  
    }  
}
```

costo $O(n \log n)$

NUOVO ANNO !

LEZIONE 13

Esercizio 1

Distributore bibite \rightarrow 3 monete 10 cent 20 cent bibite solo se 50 cent
 cifre > 50 cent \rightarrow sputa monete

f_1 = aspetta \rightarrow inserimento f_4 .

f_2 = ok

f_3 = troppo \rightarrow sputa

$A = \{10\text{m}, 20\text{m}, 50\text{m}\}$

$B = \{A \quad R \quad E$
 $\quad \text{ancora, restituisci, emetti}\}$

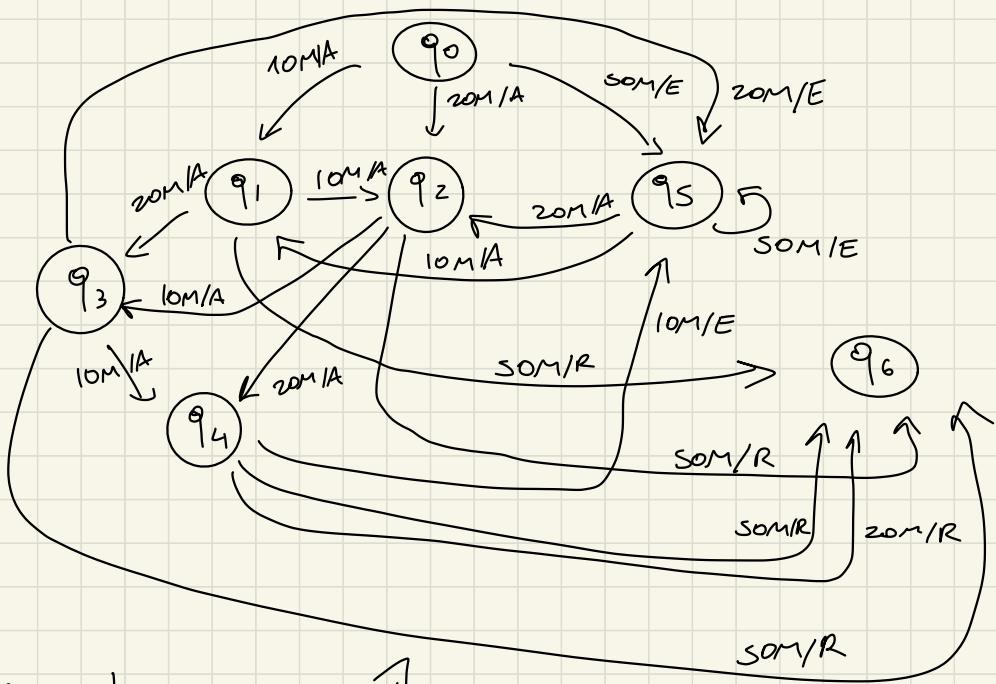
A	A	A	A	A	E	R
0	10	20	30	40	50	>50
q_0	q_1	q_2	q_3	q_4	q_5	q_6

Stati finali e iniziali $q_i = q_0$
 $q_f = q_5$

(G)	10M	20M	50M
q_0	q_1	q_2	q_5
q_1	q_2	q_3	q_6
q_2	q_3	q_4	q_6
q_3	q_4	q_5	q_6
q_4	q_5	q_6	q_6
q_5	q_1	q_2	q_5
q_6	X	X	X

(F)	10M	20M	50M
q_0	A	A	E
q_1	A	A	R
q_2	A	A	R
q_3	A	E	R
q_4	E	R	R
q_5	A	A	E
q_6	R	R	R

X \rightarrow torna al stato prec.



• Grafo di transizioni \Updownarrow

Esercizio 2

Alfabeto $A = \{a, b, c, \$\}$, linguaggio $L \subseteq A^*$

Stringhe $\in L$ tipo $t_1 \rightarrow Q + t_2 / b + t_2$
 $t_2 \rightarrow \$ / C + \text{stringa} \xrightarrow{+} t_2$
 $\$ \text{ termina}$

Devo creare un automa che definisce se una stringa appartiene a L
di formato da t_1 e t_2 .

A $\{a, b, c, \$\}$

B $\{\text{una appartenente a } L \text{ NP, appartenente a } L +, LT_1, \text{ appartenente a } L t_2 LT_2\}$

Q $\{q_0, q_1, q_2, q_3, q_4, q_5\}$
A A A A EB EG

q_0 offre simbolo

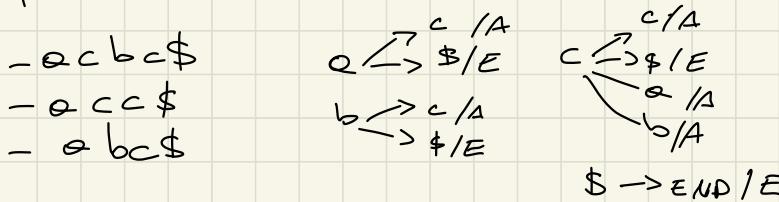
q_1 $a \rightarrow$ appartenere \rightarrow vuole $T_2 (\$, c)$

q_2 $b \rightarrow$ appartenere \rightarrow vuole $T_2 (\$, c)$

q_3 $c \rightarrow$ appartenere \rightarrow vuole $\$, c, T_1(a, b) + T_2 (\$, c)$

q_4 $\$ \rightarrow$ appartenere e termina $\in L$

$q_5 \rightarrow$ terminazione $\notin L$



$$q_i = q_0$$

$$q_f = q_4, q_5$$

tabelle degli stati:

tabelle uscite

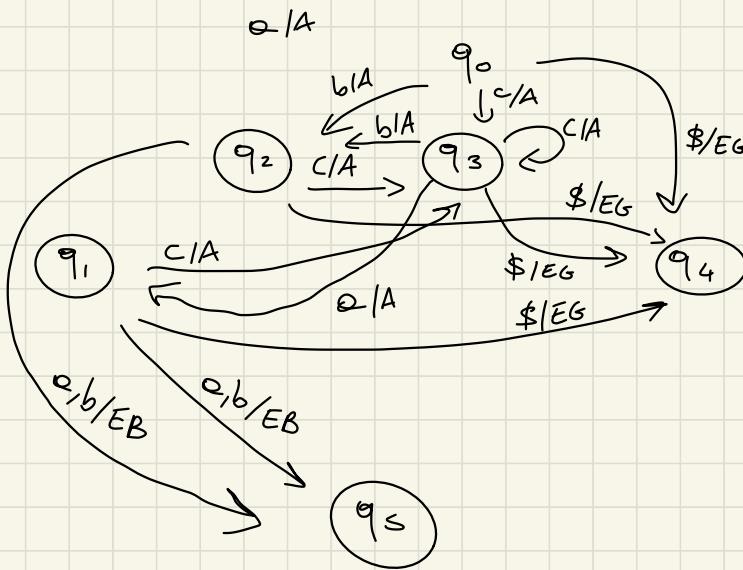
G

	a	b	c	\$
q ₀	q ₁	q ₂	q ₃	q ₄
q ₁	q ₅	q ₅	q ₃	q ₄
q ₂	q ₅	q ₅	q ₃	q ₄
q ₃	q ₁	q ₂	q ₃	q ₄
q ₄				
q ₅				

F

	a	b	c	\$
q ₀	A	A	A	EG
q ₁	EB	EB	A	EG
q ₂	EB	EB	A	EG
q ₃	A	A	A	EG
q ₄				
q ₅				

grafo di transizione



- a c b c \$ \rightarrow q₁ q₃ q₂ q₄ $\in \mathcal{L}$

- a c c \$ \rightarrow q₁ q₃ q₃ q₄ $\in \mathcal{L}$

- a b c \$ \rightarrow q₁ q₅ $\notin \mathcal{L}$ batch!

ESERCIZIO 3

Scrivere comportamento MT in grado di incrementare di 5 unità un numero scritto in base 10 sul nostro, scrivendo gli elenchi delle quintupla. Scrivere la nostra rice funzionale.

$$b > 3 \rightarrow 0$$

$\langle A, Q, P, q_i, q_f \rangle$

$$A = \{\phi, 1, 2, 3, 4, 5, 6, 7, 8, 9, \emptyset\}$$

P =	\emptyset	q_0	S	q_1	D
1	q_0	6	q_1	D	D
2	q_0	7	q_1	D	D
3	q_0	8	q_1	D	D
4	q_0	9	q_1	D	D
5	q_0	\emptyset	q_1	D	D
6	q_0	1	q_1	D	D
7	q_0	2	q_1	D	D
8	q_0	3	q_1	D	D
9	q_0	4	q_1	D	D

$$Q = \{q_0, q_1\} \quad 2 \text{ stat.}$$

Conf: initial \rightarrow O_C in g_0

q_i : $q_0 = N$. letto con sonnetto

$q_f: q_i = n$ somatos & 5

TLS legge \rightarrow incrementa \rightarrow
si sposta a destra

M. fuscivole

Esercizio 4

Descrivere il comportamento di uno undt in grado di incrementare di 1 unità un num. scritto in base 2.

$$A = \{\emptyset, 1, 2\}$$

$$Q = \{q_0, q_1\} \quad 2 \text{ stati}$$

$q_i \rightarrow$ n. non incrementato

$q_f \rightarrow$ n. incrementato $f > 2 \rightarrow \infty$

Conf. iniziale = DC in 0

P:	\emptyset	q_0	1	q_1	0
1	q_0	2	q_1	D	
2	q_0	\emptyset	q_1	D	

	0	1	2
q_0	1 q_1 , D	2 q_1 , D	0 q_1 , D
q_1			

TLS \rightarrow legge \rightarrow incremento
 \rightarrow si sposta a destra

LEZIONE 14

Esercizio 1

Dimostrare che valgono le seguenti relazioni:

$$1 \ (5n+2) \cdot \log(n) = O(n \log n)$$

$$2 \ (5n+2) \log(n) = O(n^2)$$

$$3 \ 5n^3 + 3n + 8 = O(n^3)$$

$$4 \ 5n^3 + 3n + 8 \log(n) = O(n^3)$$

$$5 \ 5n^2 + \sin(n) = O(n^2)$$

$$6 \ 2^n + n^{2000} = O(2^n)$$

1) Devo dimostrare che $\exists \alpha > 0$, $\exists n_0 \in \mathbb{N}$ t.c. $\forall n > n_0 \quad f(n) \leq \alpha \cdot n \log n$
limite

Considero $(5n+2) \log n \leq \alpha \cdot \log n \iff \alpha \geq 1 +$

$$\lim_{n \rightarrow +\infty} \frac{(5n+2) \log n}{n \log n} \rightarrow \lim_{n \rightarrow +\infty} \frac{5n+2}{n} = \lim_{n \rightarrow +\infty} \frac{5 + \frac{2}{n}}{\frac{n}{n}} = 5$$

2) con limite

$$\lim_{n \rightarrow +\infty} \frac{(5n+2) \log n}{n^2} = 0 \rightarrow \text{la relazione non vale}$$

3) con limite

$$\lim_{n \rightarrow +\infty} \frac{5n^3 + 3n + 8}{n^3} = \lim_{n \rightarrow +\infty} \frac{\cancel{n^3}(5 + \frac{3}{n^2} + \frac{8}{n^3})}{\cancel{n^3}} = 5$$

4) Con limite

$$\lim_{n \rightarrow +\infty} \frac{5n^3 + 3n + 8 \log n}{n^3} = \lim_{n \rightarrow +\infty} \frac{n^3 \left(5 + \frac{3}{n^2} + \frac{8 \log n}{n^3}\right)}{\sqrt[n]{n^3}} = 5$$

5) Con limite

$$\lim_{n \rightarrow +\infty} \frac{5n^2 + 5 \sin n}{n^2} = \lim_{n \rightarrow +\infty} \frac{5n^2 + \overset{0}{\cancel{5 \sin n}}}{n^2} = 5$$

6) Con limite

$$\lim_{n \rightarrow +\infty} \frac{2^n + n^{2000}}{2^n} = \frac{2^{-n} \cdot (2^n + n^{2000}) + 1}{1} = 1$$

Esercizio 2) Dimostrare che valgono le seguenti relazioni

$$1) (5n+2) \log n = \Theta(n \log n)$$

$$2) 5n^3 + 3n + 8 = \Theta(n^3)$$

$$3) 5n^3 + 3n + 8 \log n = \Theta(n^3)$$

$$4) 5n^2 + 5 \sin(n) = \Theta(n^2)$$

$$5) 2^n + n^{2000} = \Theta(2^n)$$

$$\lim_{n \rightarrow +\infty} \frac{f(n)}{g(n)} = l \neq 0$$

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

$$1) \lim_{n \rightarrow +\infty} \frac{(5n+2) \log n}{n \log n} = 5 \neq 0 \text{ vale}$$

$$2) \lim_{n \rightarrow +\infty} \frac{5n^3 + 3n + 8}{n^3} = 5 \neq 0 \text{ vale}$$

$$3) \lim_{n \rightarrow +\infty} \frac{5n^3 + 3n + 8 \log n}{n^3} = 5 \neq 0 \text{ vale}$$

$$4) \lim_{n \rightarrow +\infty} \frac{5n^2 + \sin(n)}{n^2} = \lim_{n \rightarrow +\infty} n^2 \left(5 + \frac{\sin(n)}{n^2} \right) = 5 \neq 0 \text{ vale}$$

$$5) \lim_{n \rightarrow +\infty} \frac{2^n + n^{2000}}{2^n} = 1 \neq 0 \text{ vale}$$

Esercizio 3

Sia $f(n) = 10 \cdot n^2 \cdot \log(n)$ dire cosa è vero e cosa no

1. $f(n)$ è di ordine non superiore a
- n^3 **vero**
 - n^2 **falso**
 - $n^2 \cdot \log n$ **vero**
 - $n \cdot \log n$ **falso**
- $f(n)$ è di ordine uguale a
- n^3 **falso**
 - n^2 **falso**
 - $n^2 \cdot \log n$ **vero**
 - $n \cdot \log n$ **falso**

$$\lim_{n \rightarrow +\infty} \frac{10 \cdot n^2 \log(n)}{n^2 / n^3 / n^2 \cdot \log n / n \cdot \log n} \quad \left. \begin{array}{l} \text{dim.} \\ \text{dim.} \end{array} \right\}$$

Esercizio 4

$$3 \cdot (2^n + n) = \mathcal{O}(2^n) \quad \rightarrow \lim_{n \rightarrow +\infty} \frac{6^n + 3n}{2^n} = 3 \quad \text{vero}$$

$$(n+1)(3n^2 + 4) = \mathcal{O}(n^2) \rightarrow \lim_{n \rightarrow +\infty} \frac{(n+1)(3n^2 + 4)}{n^2} = \frac{3n^3 + 4n + 3n^2 + 4}{n^2} = \frac{\text{falso}}{\infty} = \infty$$

$$(n+1)(3n^2 + 4) = \mathcal{O}(n^3) \rightarrow \text{vero} = 3$$

Esercizio 5

Dim.

1) $O(2 \log(n)) = O(\log(n))$ 2 è costante, quindi non conta
 a) se $f(n) \in O(2 \log(n))$ allora $f(n) \in O(\log n)$
 b) se $f(n) \in O(\log n)$ allora $f(n) \in O(2 \log n)$

2) $\lim_{n \rightarrow +\infty} \frac{f(n)}{\log n} = l \in \mathbb{R}$ quindi $\lim_{n \rightarrow +\infty} \frac{f(n)}{\log(n)} = l \in \mathbb{R}$

2) algoritmo analogo

3) " "

4) " "

5) " "

LEZIONE 15

Esercizio 1

Considerate le funz. in c dire costo restituisce f_1, f_2, f_3
 Calcolare costo temporale (O) di f_1, f_2, f_3

```
int f1(int n) {
    int i = 1; C
    int s = 0; C
    while (i <= n) {
        s = s + 2 * i; dipende da n
        i = i + 1; O(n)
    }
    return s; C
}
```

$O(n)$

```
int f2(int n) {
    int s = 0; C
    for (int i = 1; i <= n; i++) {
        s = s + 2 * i; dipende da n
    }
    return s; C
}
```

$O(n)$

int f3(int n) {
 C return
 > [n * (n + 1);]
 OUT $O(1)$

$n \rightarrow 1 = 2 \rightarrow 4$
 $n \rightarrow 2 = 6 \rightarrow 12$ finiti
 $n \rightarrow 3 = 12 \rightarrow 20$
 $n \rightarrow 4 = 20 \rightarrow 8$

$O(1)$

Fanno tutte e 3 lo stesso cosa. Generano un numero che portenolo da 2 viene incrementato di $c+2$ a ogni ciclo per n

Esercizio 2

Quale algoritmo f? Calcola costo.

void f(int n) { c }

```

for (int i = 0; i <= n; i++) { O(n)
    for (int j = 0; j <= n; j++) { O(n) } O(n^2)
        printf("%d %t", i+j) ↑ c
    printf("\n"); c
}
return;
}
  
```

$O(n^2)$

Esercizio 3

Same

void g(int v[], int n) { c }

```

for (int i = 0; i < n/2; i++) { O(n/2)
    int temp = v[i]; c ↓
    v[i] = v[n-1-i]; c O(n)
    v[n-1-i] = temp; c
}
return c
}
  
```

Esercizio 4 Same

int h(int n) {

```

if (n < 0) return -1; } c → caso ottimo O(1)
  
```

caso ottimo $O(1)$

```

int f = 1 c
for (int i = 2; i <= n; i++) { O(n)
    f = f * i; c
}
return f; c
}
  
```

caso peggiore $O(n)$

LEZIONE 18 - Master theorem

Mt: Siano $f(n)$ e $g(n)$ due funzioni decrescenti, siano $u \in v \in \mathbb{N}$ con $u \geq 1$, $v \geq 2$ t.c. $f(n) = u \cdot f(\lceil \frac{n}{v} \rceil) + g(n)$

Siano $b = \inf \{ h : g(n) = O(n^h) \}$
 $\alpha = \log_v(u)$ ($u = v^\alpha$)

Allora

N.B.: Usato soprattutto per gli alg. ricorsivi.

- | | |
|-----------------------------------|-----------------|
| 1) $f(n) = O(n^\alpha)$ | se $\alpha > b$ |
| 2) $f(n) = O(g(n))$ | se $\alpha < b$ |
| 3) $f(n) = O(g(n) \cdot \log(n))$ | se $\alpha = b$ |

2) Se regolarità
3)

\downarrow
 $\forall c \in \mathbb{R}, c > 1$ si ha che
 $g(c \cdot n) \geq c^b \cdot g(n)$

Esempio 1

$$f(n) \text{ e } g(n) \text{ funz. t.c. } f(n) = 4 \cdot f\left(\frac{n}{2}\right) + g(n) \quad u = 4$$

$$v = 2$$

$$\alpha = \log_2 4 = 2$$

• $g(n) = 5 \cdot n^2$
In questo caso $g(n) = \Theta(n^2)$ quindi $b = 2$, $\alpha = b$
 $f(n) = O(g(n) \log(n)) = O(n^2 \log(n))$ $\log_2 4 = 2$
 $\rightarrow 3^{\circ}$ caso

• $g(n) = 3\sqrt{n}$ in questo caso $g(n) = \Theta(n^{1/2})$, quindi $b = \frac{1}{2}$, $\alpha > b$
 $f(n) = O(n^\alpha) = O(n^2)$ $\alpha = 2 (\log_2 4)$

1° caso $\rightarrow f(n) = O(n^\alpha) = O(n^2)$

- $f(n) = n^3$ in questo caso $g(n) = \Theta(n^3)$ $b=3$ $a=2$ $a < b$

quindi caso 2 $\rightarrow O(f(n)) = O(n^3)$

$$f(n) = \Theta(n^3)$$

Esempio 2

$$f(n) = 3 \cdot f\left(\frac{n}{3}\right) + g(n) \quad \text{con } u=3, v=3 \quad \alpha = \log_3 3 = 1$$

- $f(n) = 5 \cdot n^2 \rightarrow O(n^2) \quad g(n) = \Theta(n^2) \quad b=2, a=1 \quad a < b$

Quindi caso 2 $O(g(n)) = \Theta(n^2)$

Il $\log n$ è $O(n^\epsilon)$ $\forall \epsilon > 0$, lo possiamo sempre vedere così. Il $\log n$ cresce meno rispetto

- $f(n) = 2 \cdot \log n \quad g(n) = O(n^\epsilon) \quad \epsilon > 0, b=0 \quad \text{quindi } a > b$
 l'inf di questo insieme sarà 0 $\uparrow \quad a=1$

Quindi caso 1 $O(g(n^\epsilon)) = O(n)$

$$n = O(n) \quad \log(n) = O(n^\epsilon) \rightarrow O(n^{1+\epsilon}) \quad \text{l'inf sarà 1} \quad \downarrow$$

- $f(n) = n \log(n) \quad g(n) = O(n^{1+\epsilon}) \quad \text{e quindi } b=1, a=1, a > b$

Quindi 3 caso $O(g(n) \cdot \log(n)) \rightarrow O(n \log^2 n)$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$n \cdot \log(n) \cdot \log(n) \rightarrow n \cdot \log^2(n)$$

Esercizi da PDF L18

es 1a) $f(n) = f\left(\frac{n}{2}\right) + n \quad u=1 \quad v=2 \quad \alpha = \log_2 1 = 0 \quad b=1$

Caso 2 $a < b \rightarrow f(n) = O(n)$

$$g(n) = n$$

$$\text{es 1 b)} \quad f(n) = f\left(\frac{n}{2}\right) + \log(n) \quad u=1 \quad v=2 \quad \alpha = \log_2 1 = 0$$

$\log n$

↑

$$g(n) = \log(n) \rightarrow O(n^\varepsilon) \rightarrow b=0 \quad \text{caso } 3 \quad \alpha=b \quad O(g(n) \cdot \log(n))$$

$$f(n) = O(\log^2(n))$$

$$\text{es 1 c)} \quad f(n) = 2 \cdot f\left(\frac{n}{2}\right) + n \quad u=2 \quad v=2 \quad b=1$$

$$\log_2 2 = 1 \quad \alpha = 1$$

Quindi $\alpha = b \geq \text{caso}$

$$O(g(n) \cdot \log(n)) \rightarrow f(n) = O(n \log(n))$$

$$\text{es 1 d)} \quad f(n) = 2 \cdot f\left(\frac{n}{2}\right) + \log(n) \quad u=2 \quad v=2 \quad \alpha = 1$$

$$g(n) = O(n^\varepsilon) = \varepsilon - 0 \quad b=0$$

Quindi caso 1 $\alpha > b$

$$O(n^\alpha)$$

$$f(n) = O(n)$$

tutte queste relazioni si possono dimostrare anche senza master theorem, però bisogna fare una dim. opposta ogni volta
 Può servire come alternativa per essere sicuri del Master theorem.
 ↳ per induzione.

[dim alternative...]

