MOLTIPLICAZIONE EGIZIA

Papiro di Ahmes 1650 AC

MOLT (A, B) $A \geq B$

```
p=0
while (A>0) {
    if A è dispari   p=p+B      (n+1)
    A=A/2   cost.
    B=B*2   cost.
}
print p
```

si esegue <u>n</u> volte

sono eseguite una sola volta. (tempo costante

A	B	P
25	13	0
12	26	13
6	52	13
3	104	13
1	208	$104+13=117$
0	416	$117+208=\underline{\underline{325}}$

$$
\begin{array}{r}
25\cdot \\
13 \\
\hline
2.5\,{}^{75}
\end{array}
$$

$\underline{\cdot\ 325}$

Correttezza

$$A \cdot B = \begin{cases} \frac{A}{2} \cdot 2B & A \text{ pari} \\ \frac{A}{2} \cdot 2B + B & A \text{ dispari} \end{cases}$$

proprietà ripetuta in ogni ciclo.

↳ questo algoritmo si usa nei computer moderni (ALU)

$$25 = 16 + 8 + 1$$
$$= 2^4 + 2^3 + 1$$
$$= (11001)_2$$

$$13 = 8 + 4 + 1 = 2^3 + 2^2 + 1$$
$$= (01101)_2$$

$$A/2 = (01100)_2 = (12)_{10}$$

shift destro

$$2 \cdot B = (011010)_2 = 26$$ shift sx

$$
\begin{array}{r}
011010 \; + \\
100111 \\
\hline
1000001
\end{array}
$$

tempo proporzionale al
di cifre binarie

n = cifre binarie

costo somma = n+1 op.
elementari

COSTO IN TEMPO

in funzione di n

$n = \#$ cifre di A e B

↓

dimensione del problema

$n*(n+1)$ + costante

$\simeq n^2$

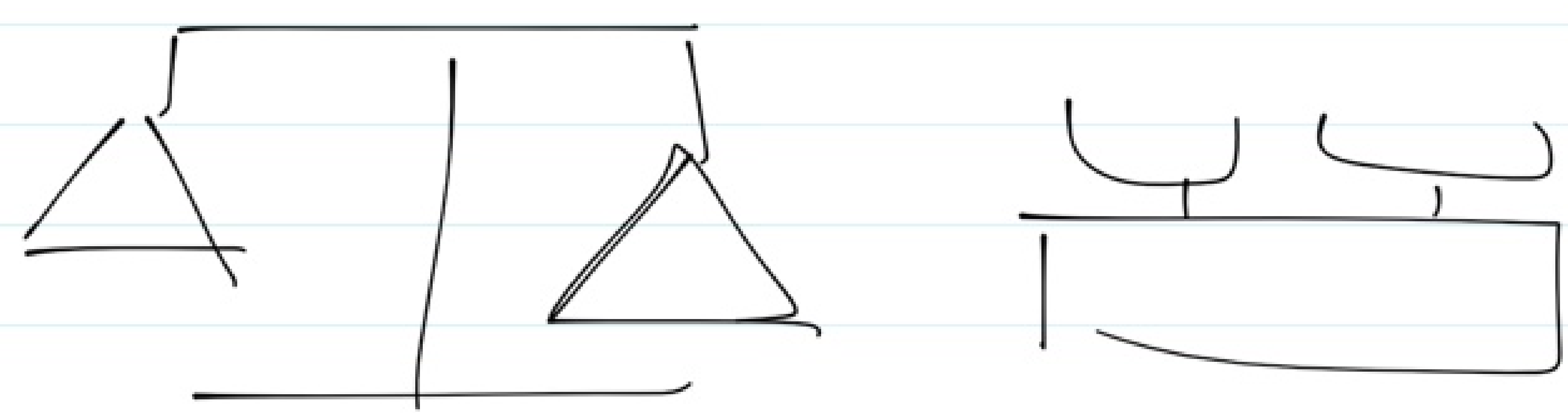
algoritmo quadratico

```
    1011
    0101
--------
    1011
  0000-
 1011--
--------
  110111
```

n^2 op. elementen

IL PROBLEMA delle 12 monete

12 monete identiche
una può essere falsa
(+ leggera, o più pesante)
→ individuare (se c'è) la moneta falsa, e stabilire se è + leggera o + pesante
con 3 pesate



1 : 2

$<$: $\underbrace{1L, 2P}_{2}$

$=$: $\underbrace{3L, 3P, \dots, 12L, 12P, 0}_{21}$

$>$: $\underbrace{1P, 2L}_{2}$

saranno almeno 3 (3 + 1 = 4

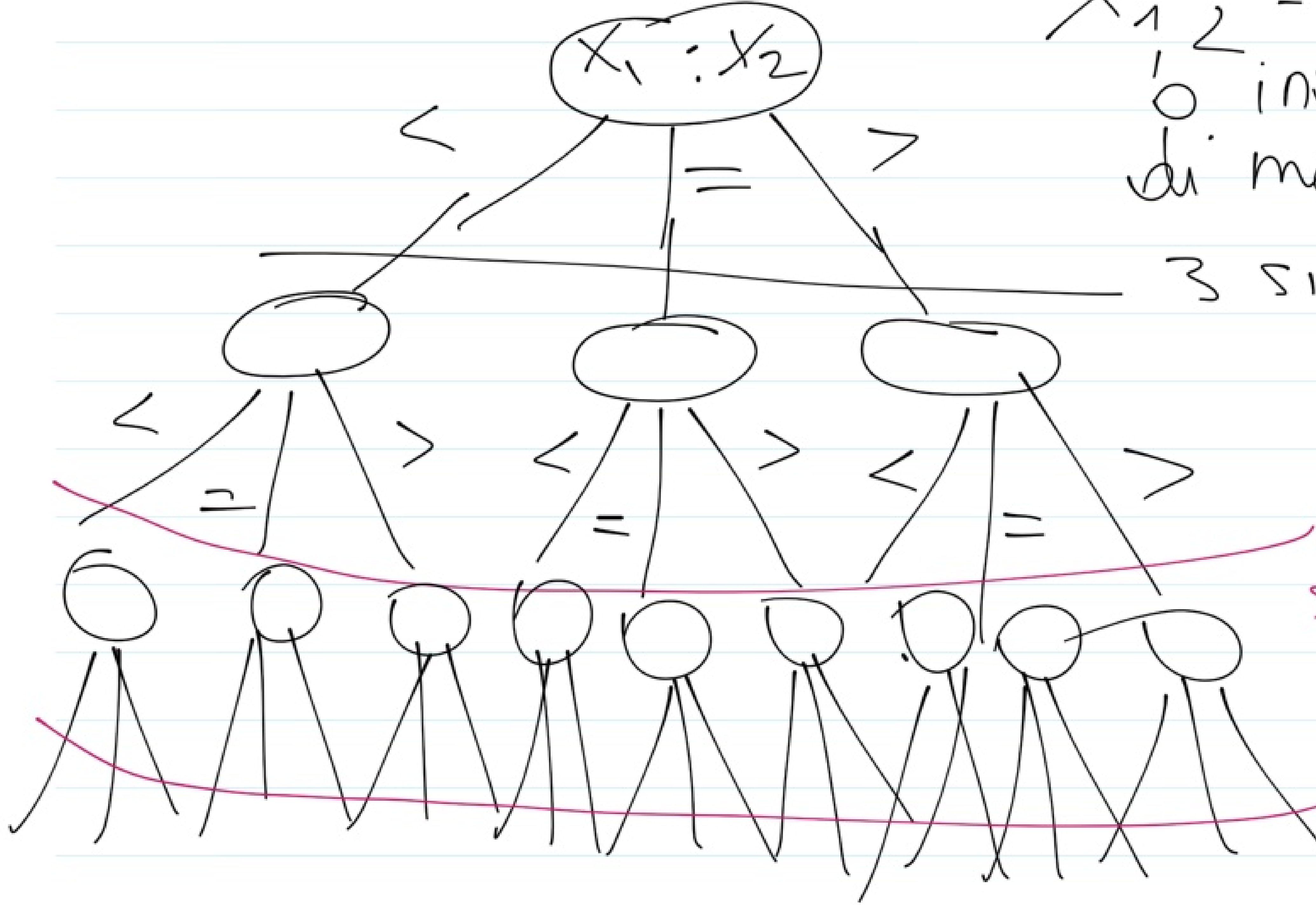
$\underbrace{1L, 1P, 2L, 2P, \dots, 12L, 12P, 0}$

25 casi possibili

soluzioni = 25

(sono i casi possibili che si possono presentare)

$x_{1,2}$ = monet o insiemi di monete



3 situation.

9 situazio

27

$3 \cdot 3 \cdot 3$

$3^i \geq 25$

$\Rightarrow i \geq 3$

i = # pesate

servono almeno
3 pesate
→ limite inferiore
x il problema

1, 2, 3, 4, 5, 6 : 7 ; 8, 9, 10, 11, 12

<

1L, ---, 6L
7P, —, 12P

12 casi

=

$\boxed{0}$

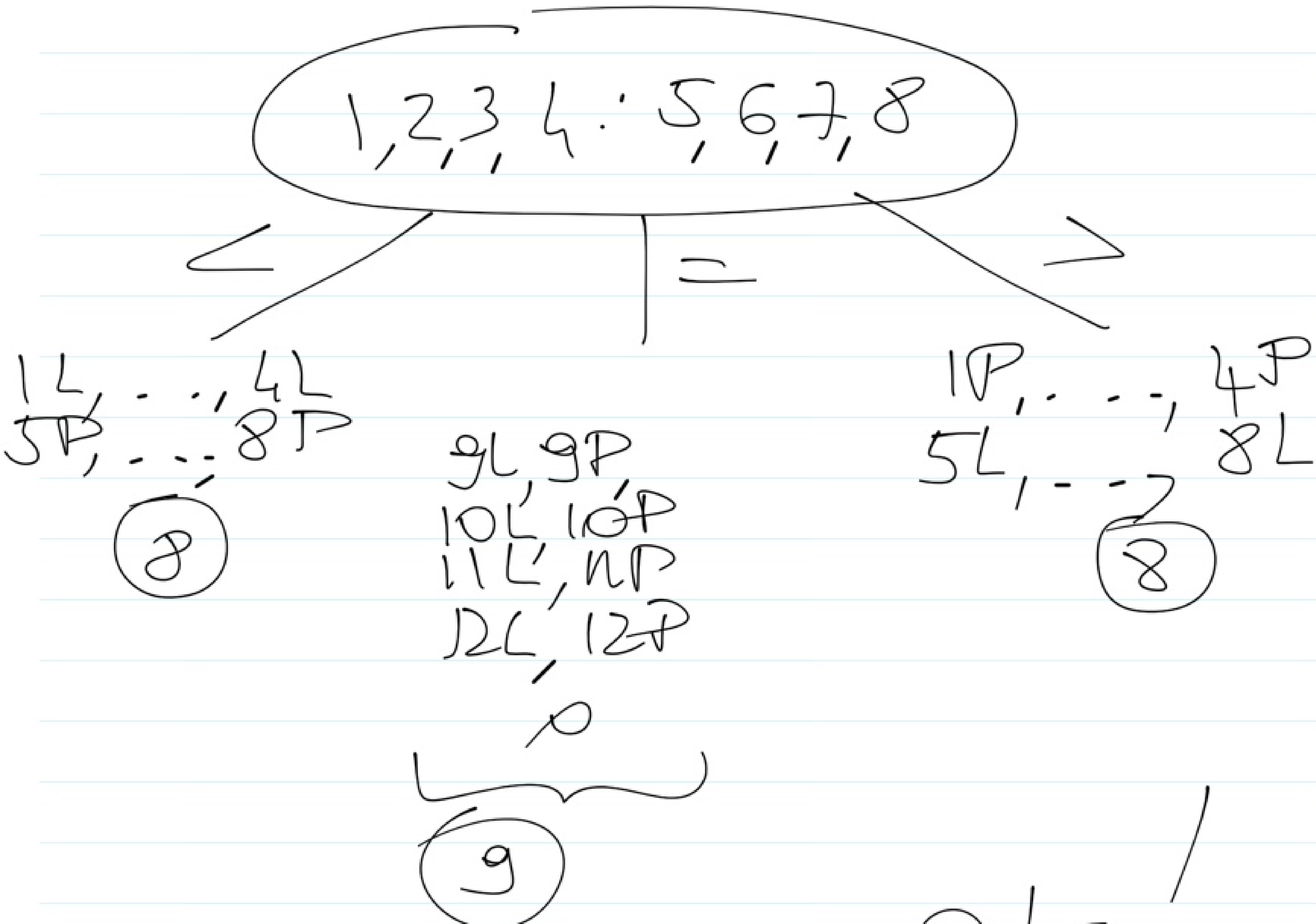
1 caso

>

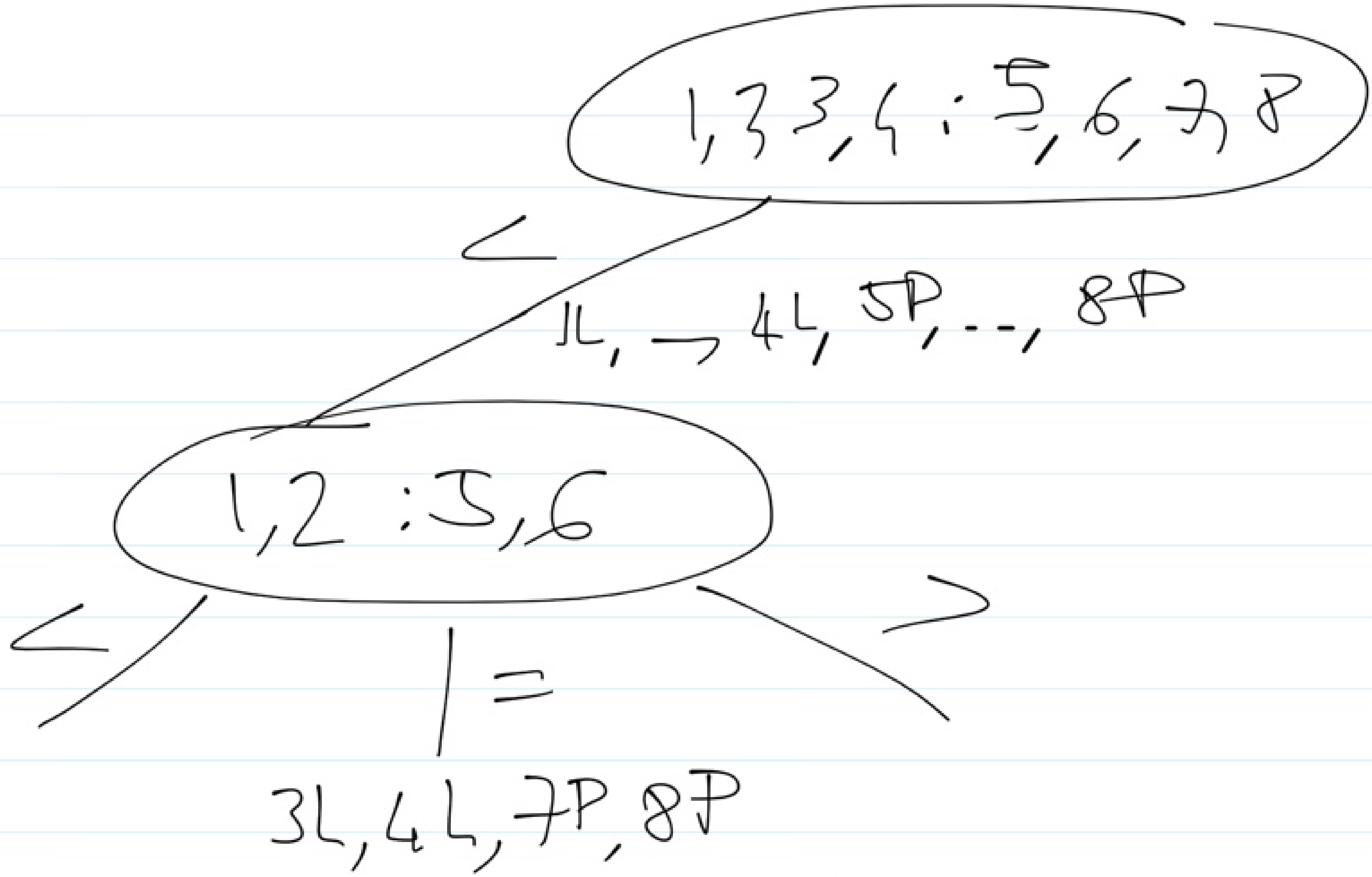
1P, ..., 6P
7L, ..., 12L

12 casi

NON VA BENE


1,2,3,4 : 5,6,7,8
<
=
>
1L, ..., 4L, 5P, ..., 8P
8
9L, 9P, 10L, 10P, 11L, 11P, 12L, 12P
p
9
1P, ..., 4P, 5L, ..., 8L
8

Ok!



Non va bene
4 casi sono troppi
per una sola pesata

L

<

1, 2, 3 : 5, 6, 7

P

<

=

>

1L, 2L, 3L,
5P, 6P, 7P,

6

↓

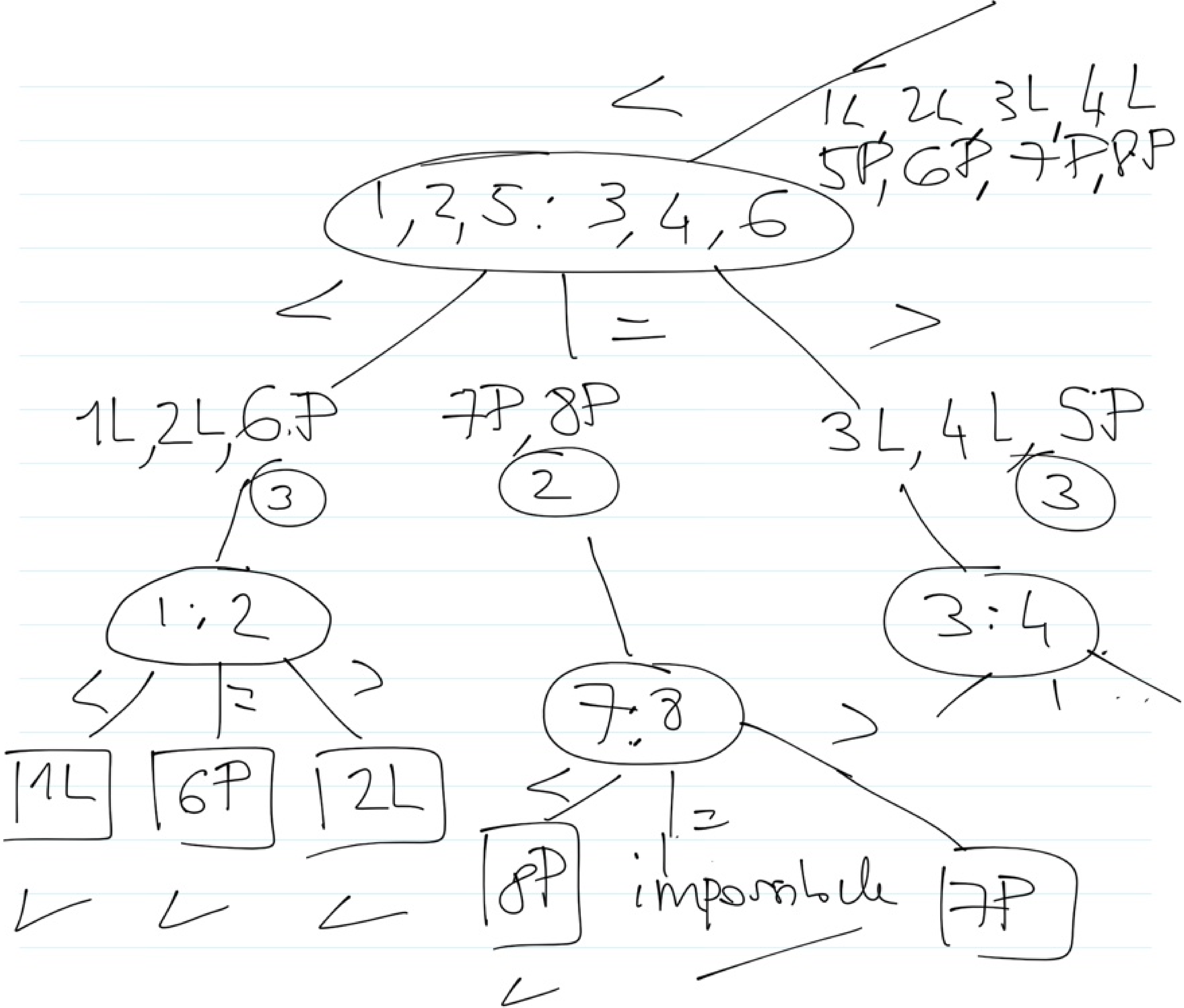
NO!

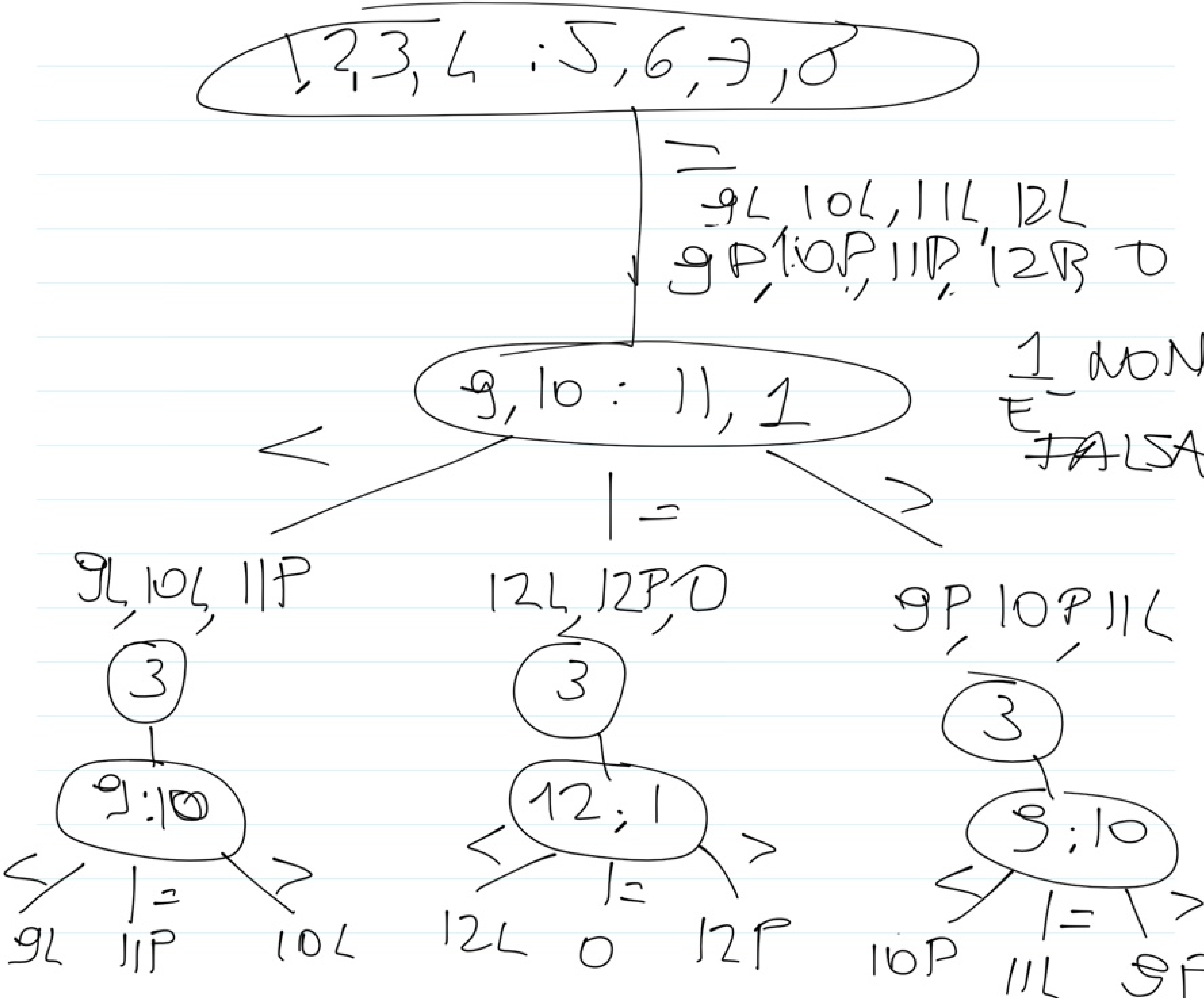
4L, 8P

OK.

2 casi

1 pesata


1L, 2L, 3L, 4L
5P, 6P, 7P, 8P
1, 2, 5 : 3, 4, 6
<
=
>
1L, 2L, 6P
7P, 8P
3L, 4L, 5P
3
2
3
1 : 2
7 : 8
3 : 4
1L
6P
2L
8P
impossible
7P


1, 2, 3, 4 : 5, 6, 7, 8
9L, 10L, 11L, 12L
9P, 10P, 11P, 12R, 0
1 NON
E ~~FALSA~~
9, 10 : 11, 1
9L, 10L, 11P
12L, 12P, 0
9P, 10P, 11L
3
3
3
9 : 10
12 ; 1
9 ; 10
9L
11P
10L
12L
0
12P
10P
11L
9P

Albero di decisione

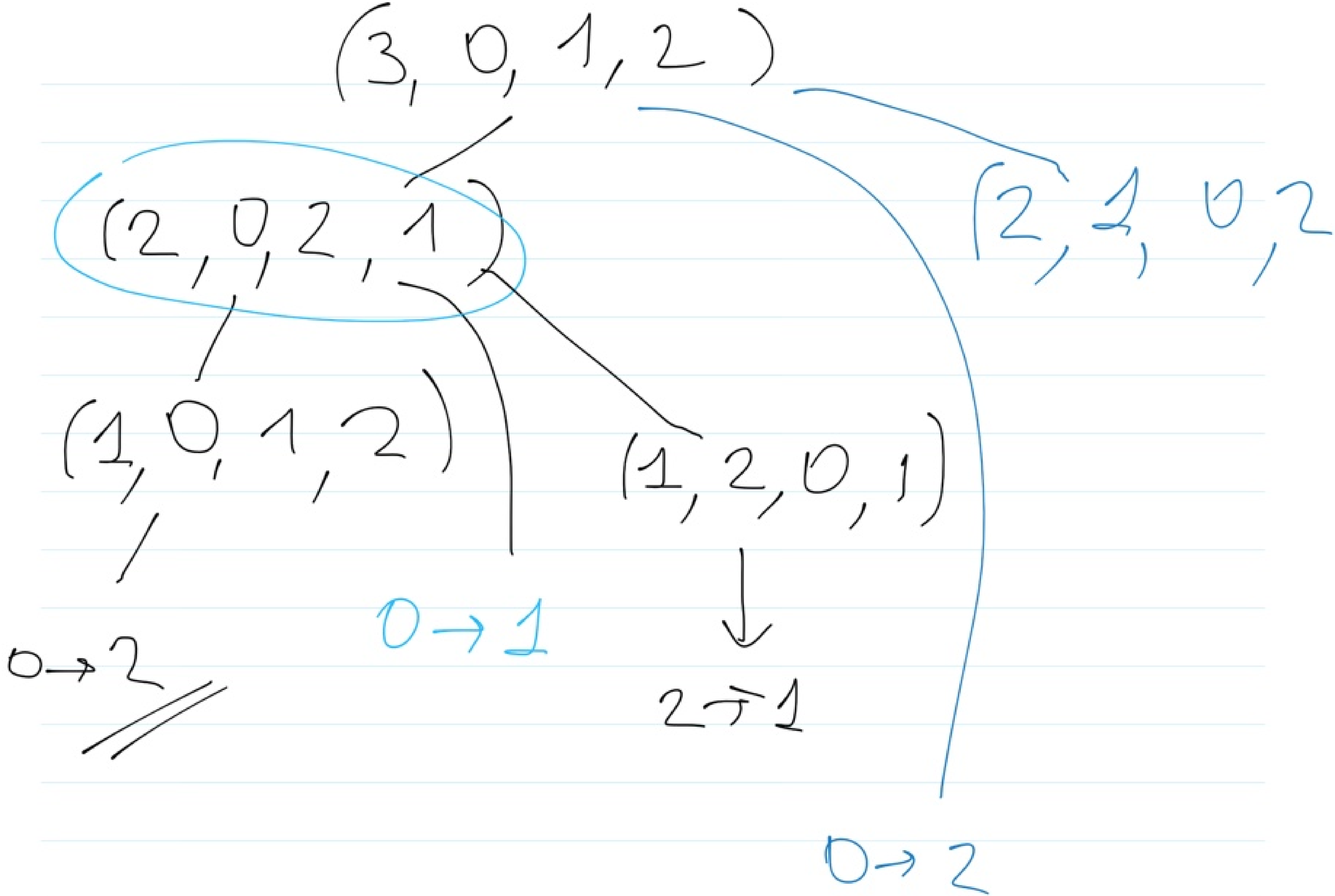
rappresenta le possibili esecuzioni dell'algoritmo

ALGORITMO OTTIMO

↳ se il suo costo è pari al limite inferiore stabilito per il problema

TorriHanoi (n, 0, 1, 2)

```
if (n==1)
    print 0 → 2;
else {
    TorriHanoi (n-1, 0, 2, 1)
    print 0 → 2
    TorriHanoi (n-1, 1, 0, 2)
}
```


(3, 0, 1, 2)
(2, 0, 2, 1)
(2, 1, 0, 2)
(1, 0, 1, 2)
(1, 2, 0, 1)
0 → 2
0 → 1
2 → 1
0 → 2

$(2, 1, 0, 2)$

$(1, 1, 2, 0)$ $(1, 0, 1, 2)$

$1 \to 2$

$1 \to 0$

$n = \#$ dischi

$M(n) = \#$ mosse

$\boxed{\text{Thm}}$ $M(n) = 2^n - 1$

Dim x induzione

base $n = 1$

$M(1) = 1 = 2^1 - 1 = 1$ ✓

Passo

$$M(n) = M(n-1) + 1 + M(n-1)$$

$$= 2M(n-1) + 1$$

$$\underset{i.i}{=} 2\left(2^{n-1} - 1\right) + 1 =$$

$$= 2^{n} - 2 + 1 = 2^{n} - 1$$

$$t = 2^{64} - 1 \text{ sec}$$

$$\simeq 585 \cdot 10^{9} \text{ anni}$$

n	5	10	...	20	30
t	31s.	17m		12g	1089a

C_1 $\quad 1 \text{ op/s}$

$t \qquad\qquad n_1 =$

$t = 2^{n_1}$

C_2 $\quad m \text{ op/s} \qquad n_2 = \#$ olisch

$t = \dfrac{2^{n_2}}{m}$

$2^{n_1} = \dfrac{2^{n_2}}{m}$

$\Rightarrow \quad 2^{n_2} = 2^{n_1} \cdot m$

$$n_2 = n_1 + \log m$$

$$t = 2^{64} \text{ sec}$$

op/sec	1	10	100	10^6	10^9
# discs	64	67	70	83	93

$$\frac{2^{64}}{m}$$


0
k pioli di appoggio
k+1

C_1 : 1 op/s

in tempo t

sposto n_1 dischi

$\#$ mosse $= 2^{n_1} (-1)$

$\Rightarrow \quad t = 2^{n_1}$

C_2 m op/sec

in ogni tempo t si spostano

n_2 dischi

$\Rightarrow$ # mosse $= 2^{n_2}$

secondi $= \dfrac{2^{n_2}}{m}$

$\Rightarrow$ $t = \dfrac{2^{n_2}}{m}$

$$t = 2^{n_1}$$

$$t = 2^{n_2}/m$$

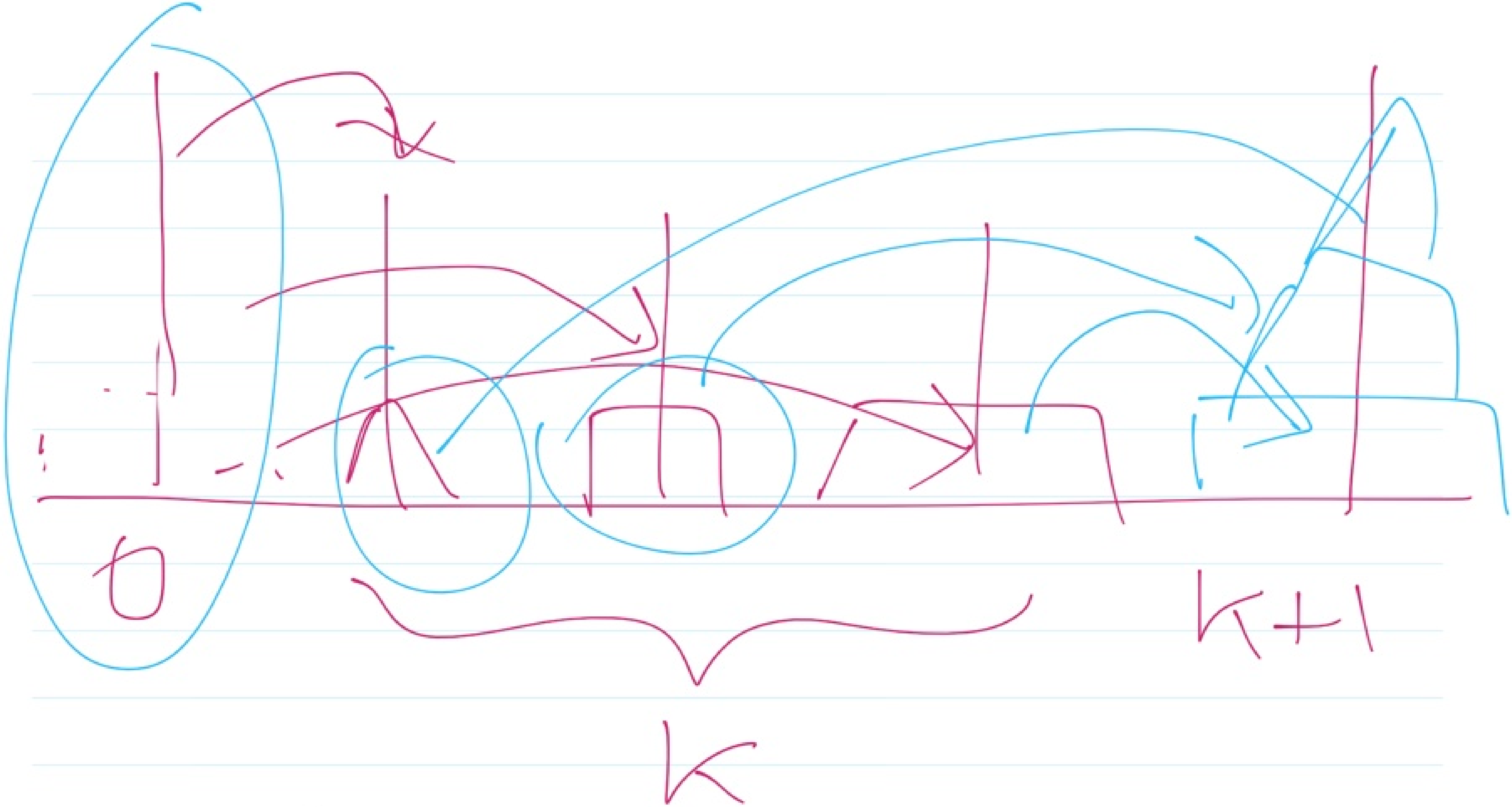
$$\Rightarrow \quad 2^{n_2} = m \cdot 2^{n_1}$$

$$n_2 = n_1 + \log m$$

$\Rightarrow$ usando calcolatore m volte più veloce si possono spostare, a pari tempo, solo $\log m$ dischi in più

$$t = 2^{64} s$$

op/sec	1	10	10^6	...	10^9
# dischi	64	67	83		93



$k \mid n$

$THG(n,k)$ $\langle n>0,\ k \mid n\rangle$

```
for (i = 1; i ≤ k; i++)
    TorriHanoi(n/k, 0, k+1, i);
for (i = k; i ≥ 1; i--)
    TorriHanoi(n/k, i, 0, k+1);
```

\# MOSSE

$$M(n,k) = 2k \cdot M\left(\frac{n}{k}\right) =$$

$$= 2k\left(2^{n/k} - 1\right)$$

$n \geq 2$

$$k = \left\lceil \frac{n}{\log n} \right\rceil$$

$$\frac{n}{\log n} \leq k \leq \frac{n}{\log n} + 1$$

$$M(n,k) = 2k\left(2^{n/k} - 1\right) =$$

$$\leq 2\left(\frac{n}{\log n} + 1\right)\left(2^{n/(n/\log n)} - 1\right) =$$

$$2\left(\frac{n}{\log n} + 1\right)\left(n - 1\right) \leq$$

$$\leq 2(n+1)(n-1) =$$

$$2(n^2-1) < 2n^2$$

$$M\left(n, \left\lceil \frac{n}{\log n} \right\rceil\right) < 2n^2$$

C_1 $\quad$ 1 op/s

$$\rightarrow t = 2n_1^2$$

C_2 $\quad$ m op/s

$$\rightarrow t = \frac{2n_2^2}{m}$$

$$\cancel{2}n_1^2 = \frac{\cancel{2}n_2^2}{m} ,$$

$$n_2^2 = n_1^2 \cdot m$$

$$\Rightarrow \quad n_2 = n_1 \cdot \sqrt{m}$$

op/sec	1	100	10^6	10^9
# dischi	64	640	64000	$\simeq 2 \cdot 10^6$

$t \simeq 2$ ore e $1/4$

Algoritmo polinomiale

$\exists$ costante $c > 0$, t.c.

passi elementari eseguiti dall'algoritmo è $\leq n^c$

per ogni istanza di input

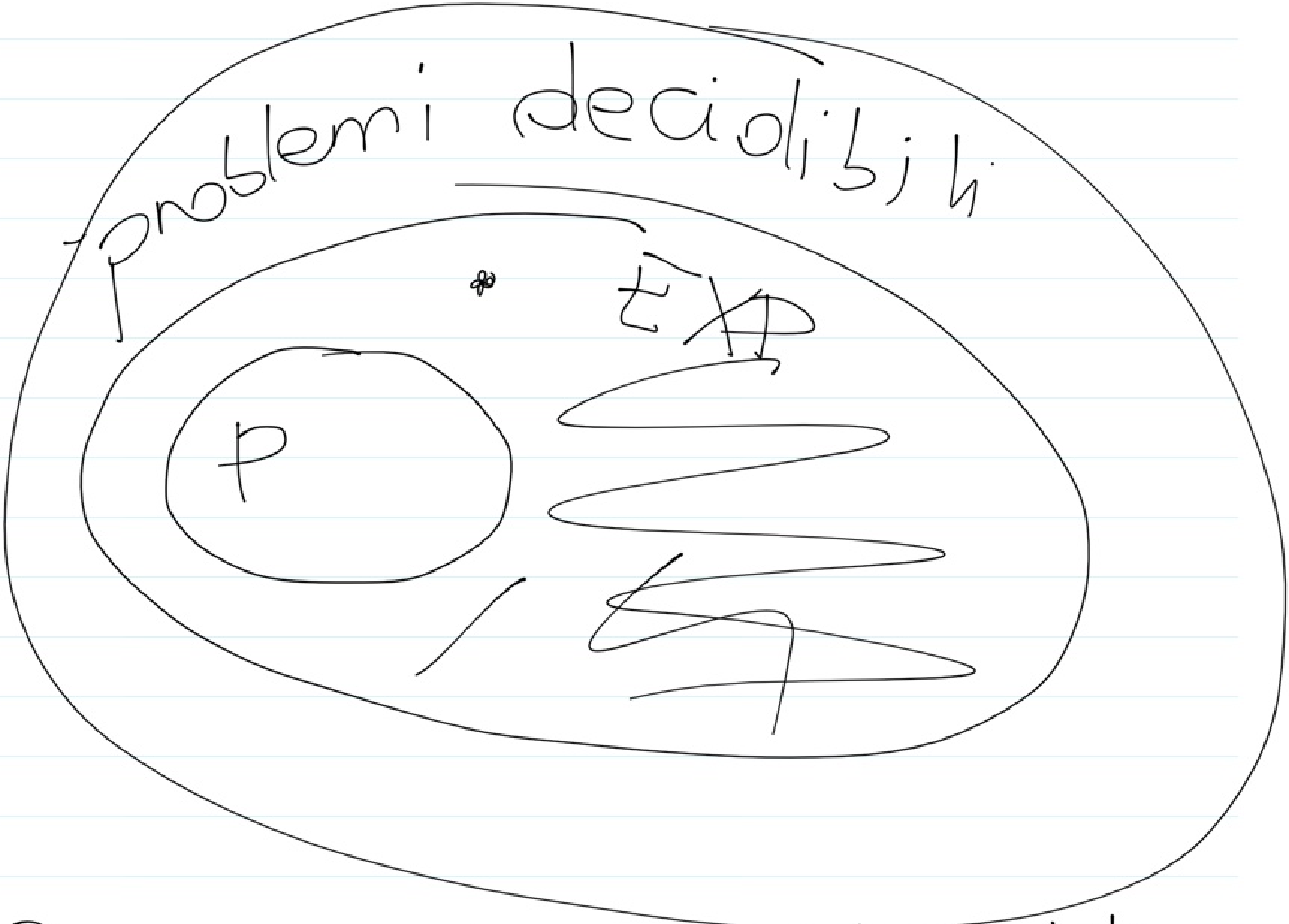
di dimensione n.

$z \in \mathbb{Z}$

si scrive con

$\lfloor \log_2 z \rfloor + 1$ bit

$+ 1$ bit di segno



P = insieme dei problemi trattabili
(ammettono alg. polinomiale)

EXP = problemi che ammettono algoritmi risolutivi di costo al più esponenziale nella dimensione dell'input

$$P \subseteq EXP$$

$EXP \setminus P$ = insieme dei problemi intrattabili.

sequenze binarie
lunghe n

000
001
010
011
100
101
110
111

$\hookrightarrow 2^n$

$$S = \{ s_0, s_1, \ldots, s_{n-1} \}$$

$$S' \subseteq S$$

b-esimo bit sequenza

$\longleftrightarrow$

elemento di indice b

$$A[b] = 1 \iff s_b \in S'$$

$S = \{S_0, S_1, S_2\}$

$\emptyset, \{S_0\}, \{S_1\}, \{S_2\}, \{S_0, S_1\}, \{S_0, S_2\}$

000 100 010 001 110 101

$\{S_1, S_2\}$ $\{S_0, S_1, S_2\}$

011 111

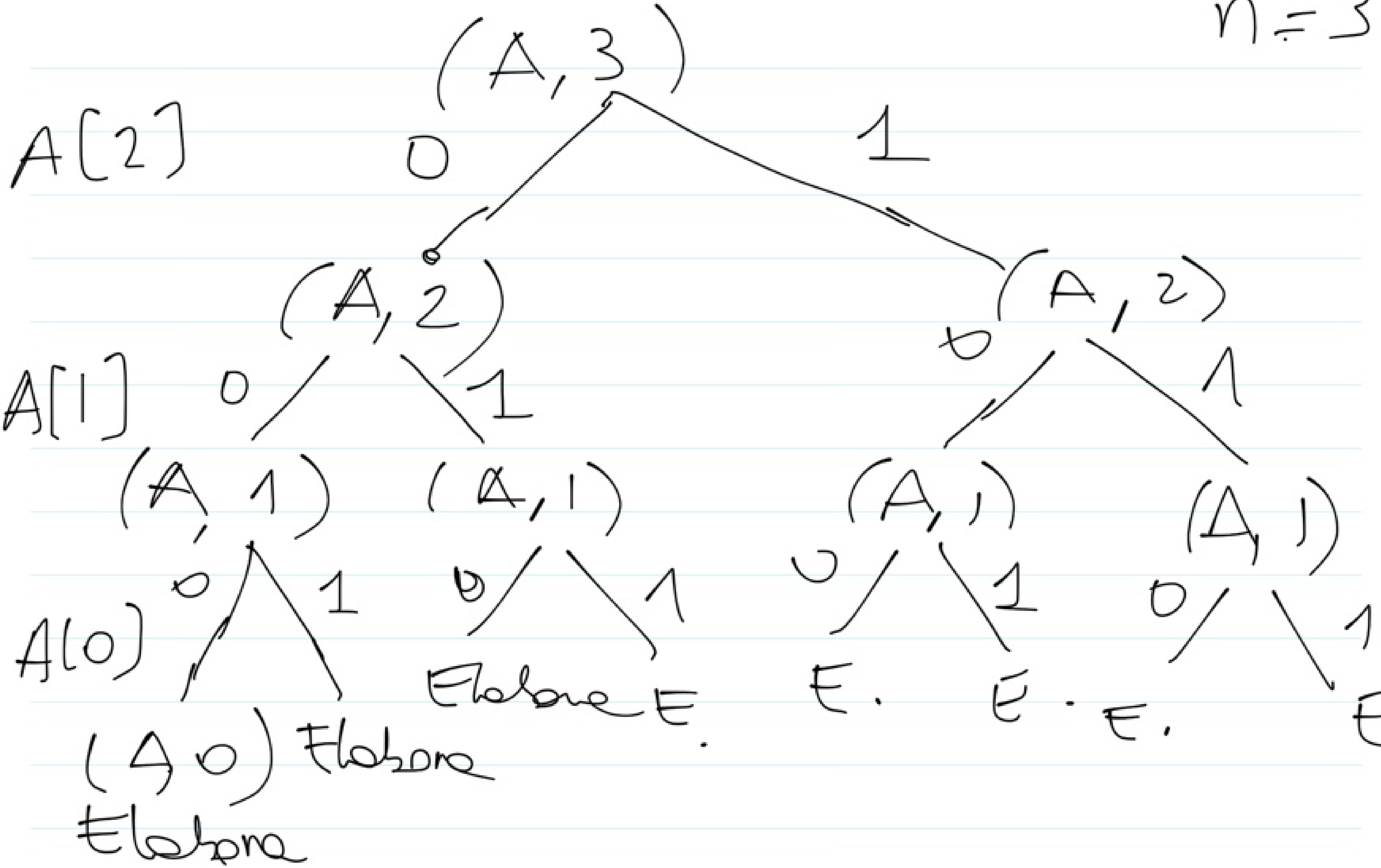
```
GeneraBinarie(A, b)
// i primi b bit di A sono da generare

if(b==0) Elabora(A);
else{
    A[b-1] = 0
    GeneraBinarie(A, b-1);
    A[b-1] = 1
    GeneraBinarie(A, b-1);
}
```

chiamata iniziale

```
GeneraBinarie(A, n)
```

$n = 3$



000 100 010 110 001 101 011 111

PARTITION

P = insieme di <u>n</u> interi <u>positivi</u>

di somma pari 2S

trovare un sottoinsieme di P di

somma S, se esiste

$P = \{2, 3, 4, 1\}$

$2S = 10$ $\{2, 3\}$

$P = \{9, 7, 5, 4, 1, 2, 3, 8, 4, 3\}$

$2S = 46$ $P' = \{5, 4, 1, 2, 8, 3\}$

<u>Elabora(P, S, A)</u>

somma = 0

for (i=0; i<n; i++) {

if (A(i)==1) somma=somma+P(i)

}

if (somma==S) : STAMPA "TROVATO"

: TERMINA

Costo Elabora $\simeq$ n

n

$n! = $ pemutation

abc	abc	abc
abc	acb	cba
bac	cab	bca

GeneraPermutazioni (A, p)
// i primi p elementi di A sono da permutare

```
if (p==0) Elabora(A)
else {
  for (i=p-1; i≥0; i--) {
      scambia(i, p-1);
      GeneraPermutazioni(A, p-1);
      scambia(i, p-1);
  }
}
```

Prima chiamata: GeneraP. (A, n)

\# operazioni $\sim$ $n!$ costo(Elabora)

CICLO HAMILTONIANO

$G = (V, E)$

V: insieme di vertici

E: insieme di archi

↳ coppie di vertici

$$E \subseteq V \times V$$

$n = |V|$

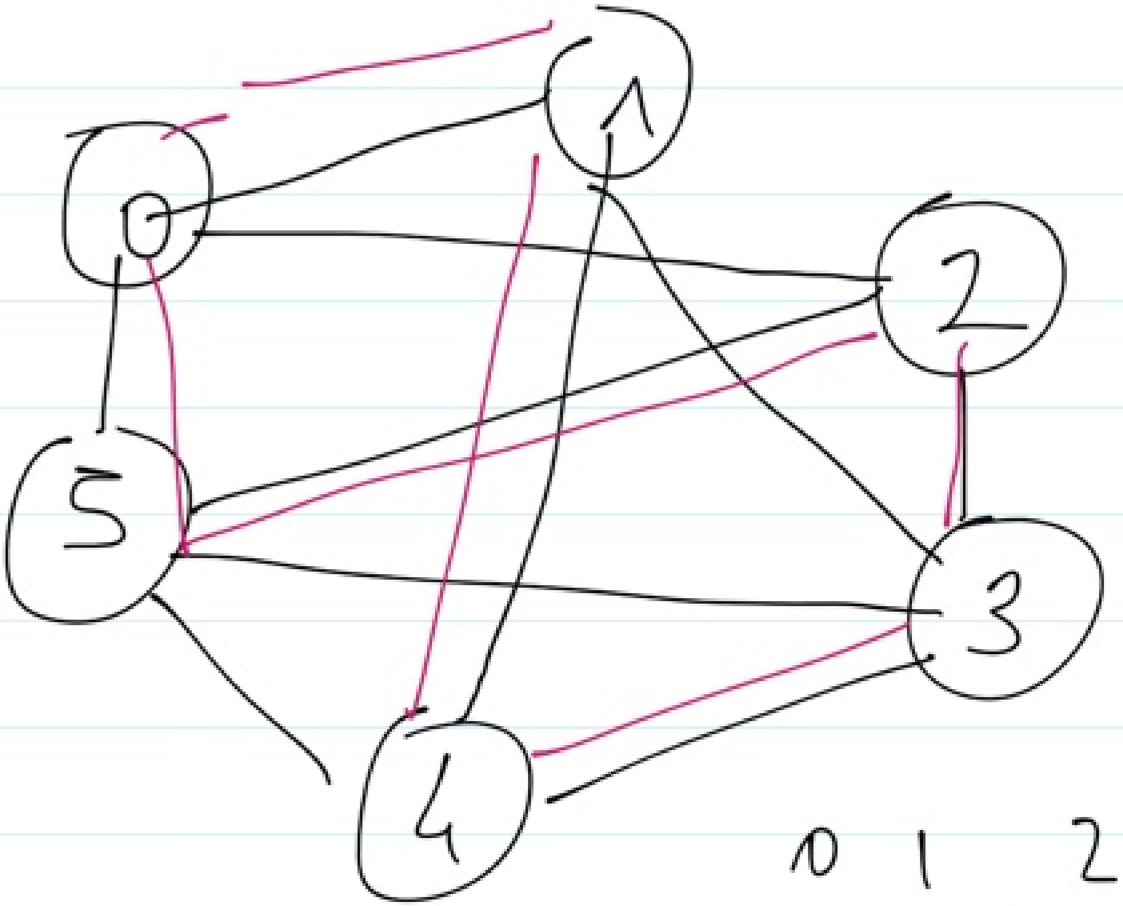
rappresentiamo i vertici con gli interi da 0 a $n-1$

Matrice di adiacenza

$M \quad n \times n$ (binaria)

$M[i][j] = 1 \iff (i,j) \in E$

$M[i][j] = 0 \iff (i,j) \notin E$



0, 1, 4, 3, 2, 5

0 1 2 3 4 5

	0	1	2	3	4	5
0		1	1			1
1	1			1	1	
2	1			1		1
3		1	1		1	1
4		1		1		1
5	1		1	1	1	

Ciclo hamiltoniano

Esiste un percorso che inizia e termina sullo stesso vertice (CICLO) e passa da ciascun vertice una ed una sola volta?

A permutazioni dei vertici

la permutazione in A è un ciclo hamiltoniano se

$\forall \; 0 \leq i < n-1$

$$M\left[\underbrace{A[i]}_{\text{riga}}, \underbrace{A[i+1]}_{\text{colonna}}\right] = 1$$

e

$$M\left[A[n-1], A[0]\right] = 1$$

Elabora(A, M)

```
trovato = true;
i = 0
while (trovato && i < n-1) {
    if (M[A(i), A(i+1)] == 0)
                    trovato = false;
    else i++;
}
if (trovato && M[A(n-1), A(0)] == 1) {
            STAMPA TROVATO CICLO
              TERMINA
}
```

trattabili

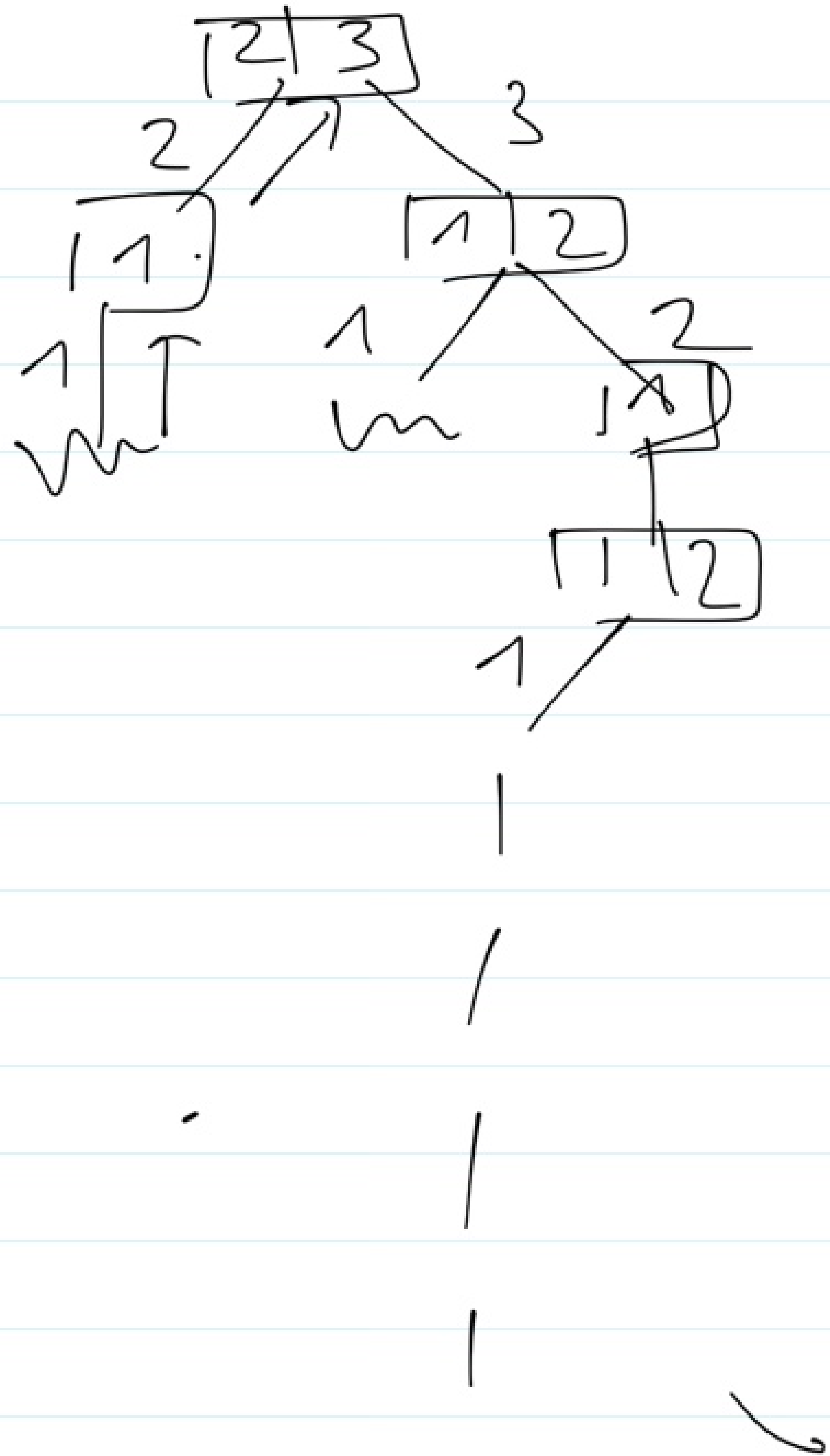
problemi

presumibilmente intrattabili

intrattabili

4	2	3	1
1	3	4	2
2	4	1	3
3	1	2	4

4	3	2	1
1	2	4	3
			2

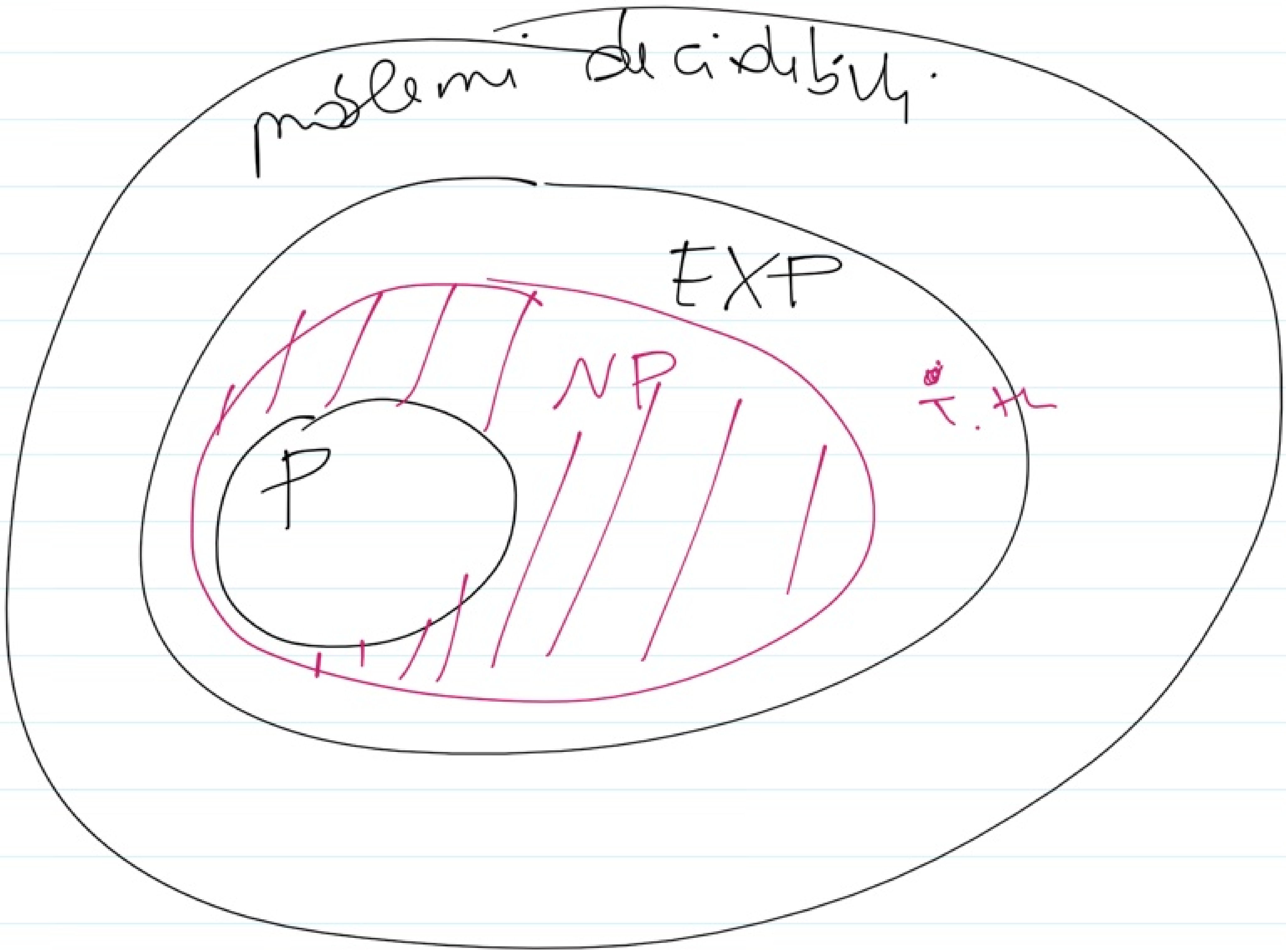


NP

classe dei problemi "verificabili" in tempo polinomiale

$$P \subseteq NP$$

$$\boxed{P \subset NP \quad ??}$$


problemi decidibili
EXP
NP
P

Certificato Polinomiale

i problemi in NP ammettono delle sequenze binarie (CERTIFICATI) t.c chi ha una soluzione per una istanza di input di un problema in NP può convincerci fornendoci un certificato che ci permetta di verificare in TEMPO POLINOMIALE l'esistenza di una qualche soluzione

|Certificato| $\not=$

il certificato deve essere polinomiale nelle dimensioni dei dati di ingresso classe P

classe NP

Problemi decisionali

si chiede l'esistenza di una soluzione che ~~gode~~ di certe proprietà (anziché determinarla)

Problemi decisionali

- la difficoltà di un problema è più presente nella versione decisionale
- tutto il tempo è speso per il calcolo

Π problema decisionale

$S = \{0, 1\}$

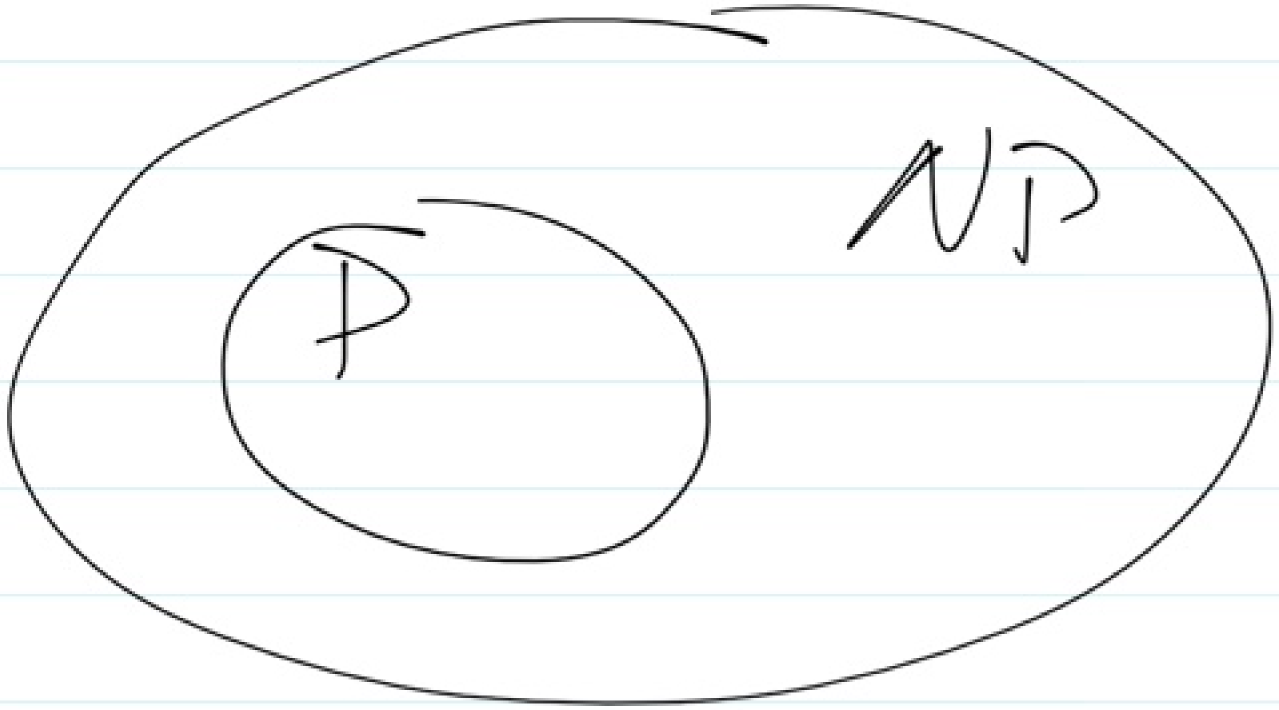
X: istanza di input

$\Pi(x) = 1$

x : istanza accettabile
(positiva)

$\Pi(x) = 0$

x : istanza non accettabile
(negativa)



$\Pi \in NP$

Π è t.c.

1) $\forall$ istanza accettabile x $(\Pi(x) = 1)$ di lunghezza n esiste un CERTIFICATO y di lunghezza polinomiale in n

2) $\exists$ un algoritmo V di verifica polinomiale in n, applicabile a ogni coppia (x, y)

1) e 2) $\Longrightarrow$

l'algoritmo V verifica il problema Π in tempo polinomiale

NP

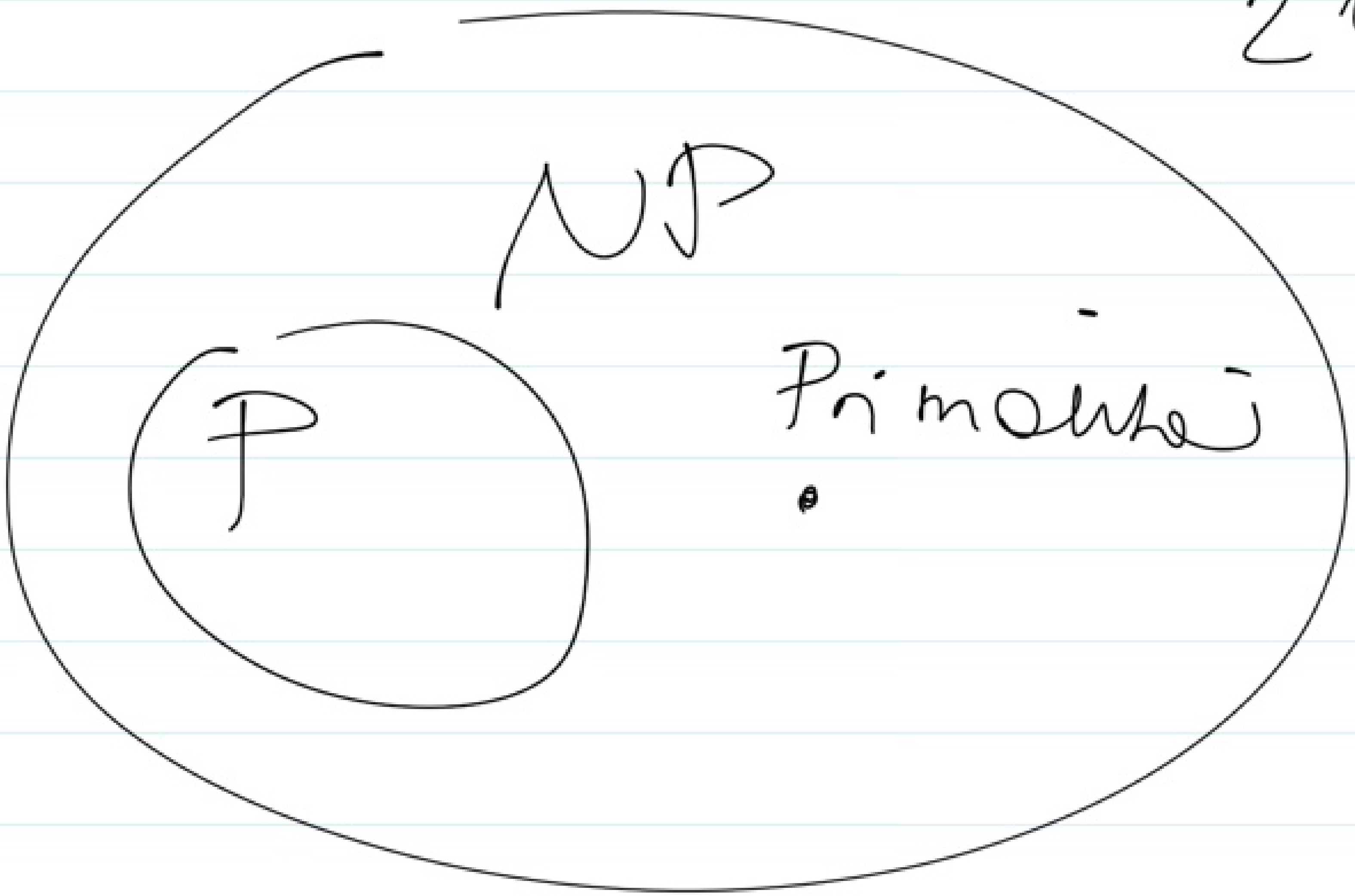
classe dei problemi VERIFICABILI IN tempo polinomiale

$\boxed{P \subseteq NP}$

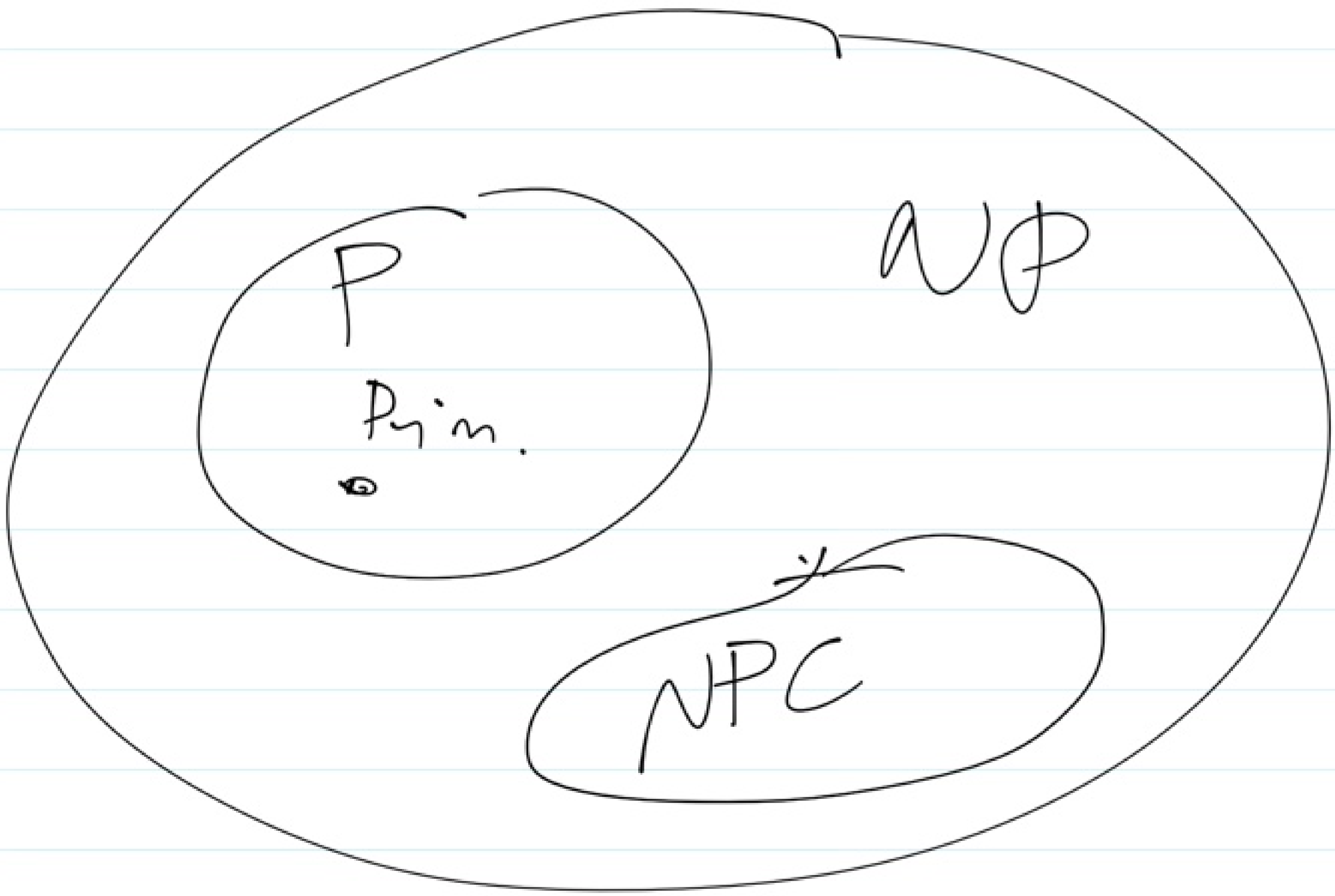
$\Pi \in P$

$\forall$ istanza x, costruisco un certificato y vuoto, e definisco un algoritmo di verifica $V(x, y)$ che ignora y e risolve direttamente Π in tempo polinomiale.

2002



dal 2003



NPC = problemi NP completi

RAM

Random Access Machine

(Macchina ad accesso diretto)

- processore di calcolo
- memoria di dim. illimitata
  (dati e programma da eseguire)

processore

- Unità Centrale di Elaborazione
- 2 registri
  - Contatore di programma → prossima istruzione da eseguire
  - Accumulatore → operazioni elementari

Operazioni elementari

- op. aritmetiche: +, −, *, /.
- op. di confronto: <, >, =
- op. logiche: ∧, ∨, ¬
- op. di trasferimento: lettura e scrittura.

accumulatore ⟷ memoria

- op. di controllo: salti condizionati e non condizionati tempo corrente di esecuzione

(SPAZIO)

Costo in tempo di un algoritmo su una istanza di input:

\# di operazioni elementari eseguite

→ # celle di memoria utilizzate durante l'esecuzione (oltre a quelle x l'input)

Complessità di un algoritmo:

si esprime come una funzione della dimensione n dei dati di ingresso,

notazione ASINTOTICA

in ordine di grandezza trascurando le costanti moltiplicative e i termini di ordine inferiore

$$3n^2 + \cancel{2n} + \cancel{5} + \cancel{\log n}$$

costo cresce come n^2

$$\propto$$

(proporzionale)

$$n^2$$

Complessità al caso pessimo

costo massimo su tutte le possibili istanze di dim. n

↳ limite superiore al tempo di esecuzione su qualsiasi input

$$\boxed{\text{Complessità al caso medio}}$$

si considera la media del costo su tutte le istanze di dim. n

VALUTAZIONE AL CASO PESSIMO DI ALCUNI COSTRUTTI

1. Singole operazioni aritmetiche/logiche e di assegnamento

   ↳ costo costante

(2) <u>if</u> (guardia) {blocco 1}
<u>else</u> {blocco 2}

costo: costo(guardia) +

max{ costo(blocco 1),
costo(blocco 2) }

③ **for**

for(i=0; i<m; i++) {corpo}

t_i = costo (corpo) all'i-sima iterazione del ciclo for

Costo: proporzionale a

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

4) WHILE, DO-WHILE

while (guardia) { corpo }

do { corpo } while (guardia)

m = # volte in cui la guardia è vera

t_i' = costo (guardia) all'iterazione i del ciclo

t_i = costo (corpo) all'iterazione i

COSTO TOTALE $\leq \sum_{i=0}^{m} (t_i + t_i')$

5) Chiamata a funzione

costo : costo del corpo della funzione
+
costo per il calcolo degli argomenti passati alla funzione

6) costo di un blocco di istruzioni e costrutti

somma dei costi delle singole istruzioni e dei costrutti

ESEMPIO

Ric(a, k)

```
i=0
indice=-1
while(i<n && indice==-1){
    if(a(i)==k) indice=i;
    else i++;
}
return indice;
```

c_3

costi costante c_2

costo costante c_1

While: quando vale $n+1$ volte

corpo: n volte

$$\text{Costo} = C_1 + (n+1)\, C_3 + n\, C_2$$

$$= C_1 + C_3 n + C_3 + n\, C_2$$

$$= n\,(C_2 + C_3) + C_1 + C_3$$

Costo è proporzionale a n

$$\Theta(n)$$

NOTAZIONE ASINTOTICA

O, Θ, Ω, o, ω

Knuth

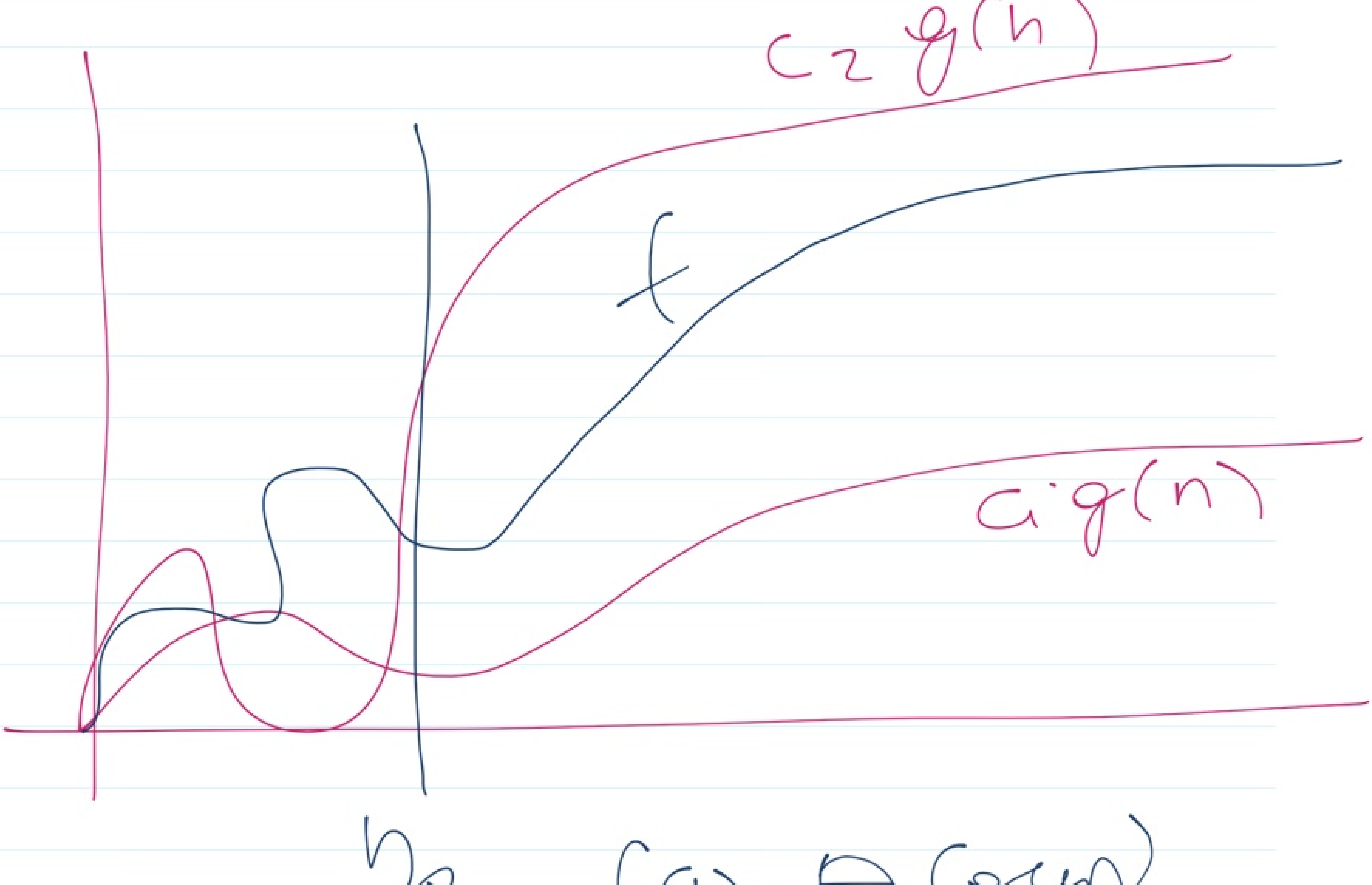
NOTAZIONE Θ $[\simeq]$

(limite asintotico stretto)

$\Theta(g(n)) = \{ f(n) : \exists\, n_0, c_1, c_2$ t.c.

$\forall n \geq n_0, \; 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 g(n) \}$

$f(n)$ "cresce come" $g(n)$, a meno di fattori costo



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

$$f(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$$

$$c_1 = \frac{1}{14}, \quad c_2 = \frac{1}{2}, \quad n_0 = 7$$

$$f(n) = \sum_{i=0}^{d} a_i \cdot n^i$$

$$a_d > 0$$

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

$$f(n) = 3n^2 + 3\log n$$

$$f(n) = \Theta(n^2)$$

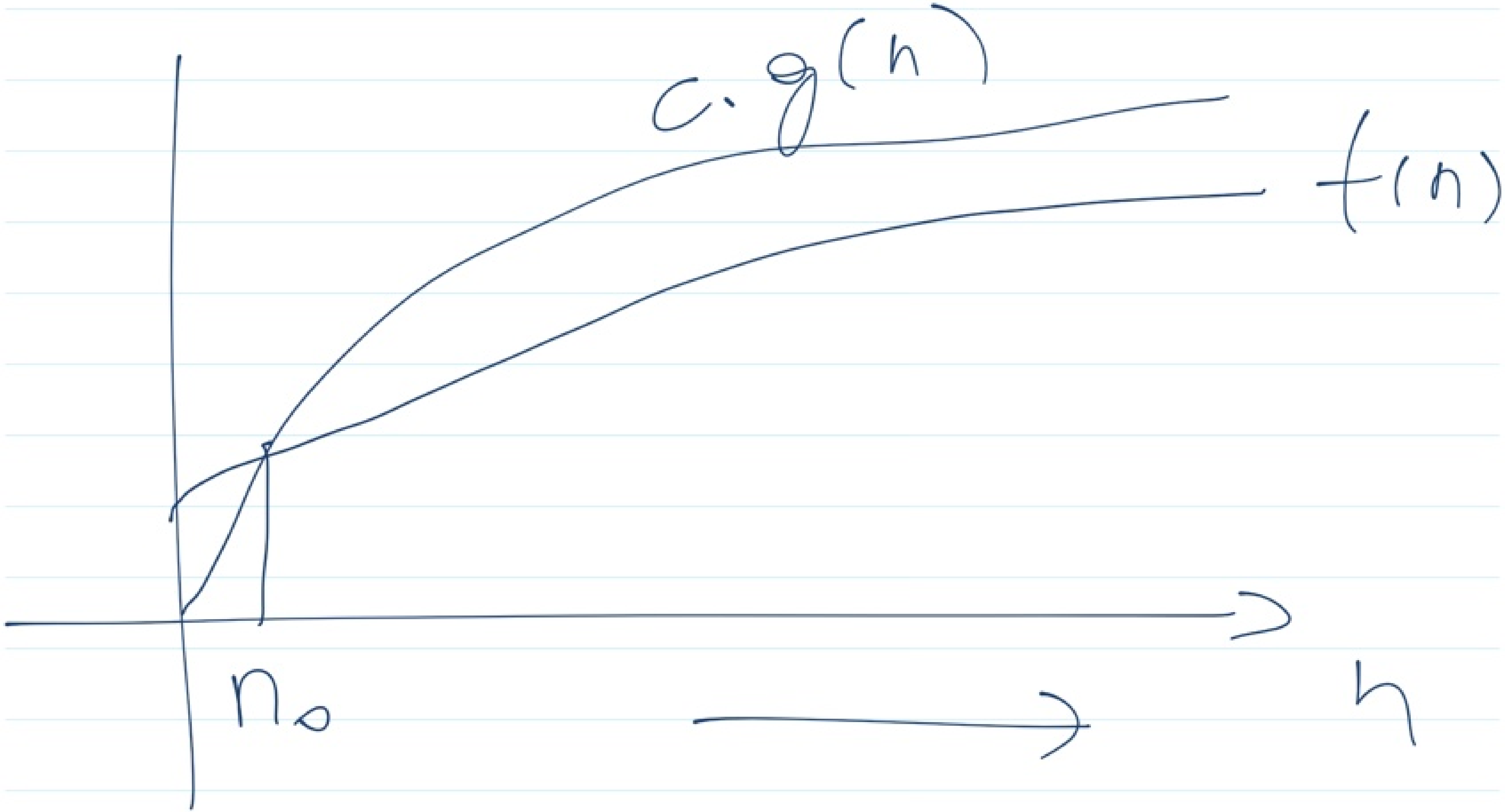
$$\Theta(\cancel{3}n^2) = \Theta(n^2)$$

NOTAZIONE O $(\leq)$

limite asintotico superiore

$O(g(n)) = \{ f(n) \mid \exists c, n_0 > 0$ t.c.
$\forall n > n_0$
$0 \leq f(n) \leq c \cdot g(n) \}$

$f(n) = O(g(n))$ se $f(n)$ cresce al più come $g(n)$ a meno di un fattore costante



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

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

$$= O(n^3)$$

$$\neq O(n^2)$$

$$= \Theta(n^2 \log n)$$

$$\neq \Theta(n^2)$$

$$\neq \Theta(n^3)$$

$$f(n) = an^2 + bn + c \qquad a > 0$$

$$= O(n^2)$$

$$= O(n^k) \quad \forall k \geq 2$$

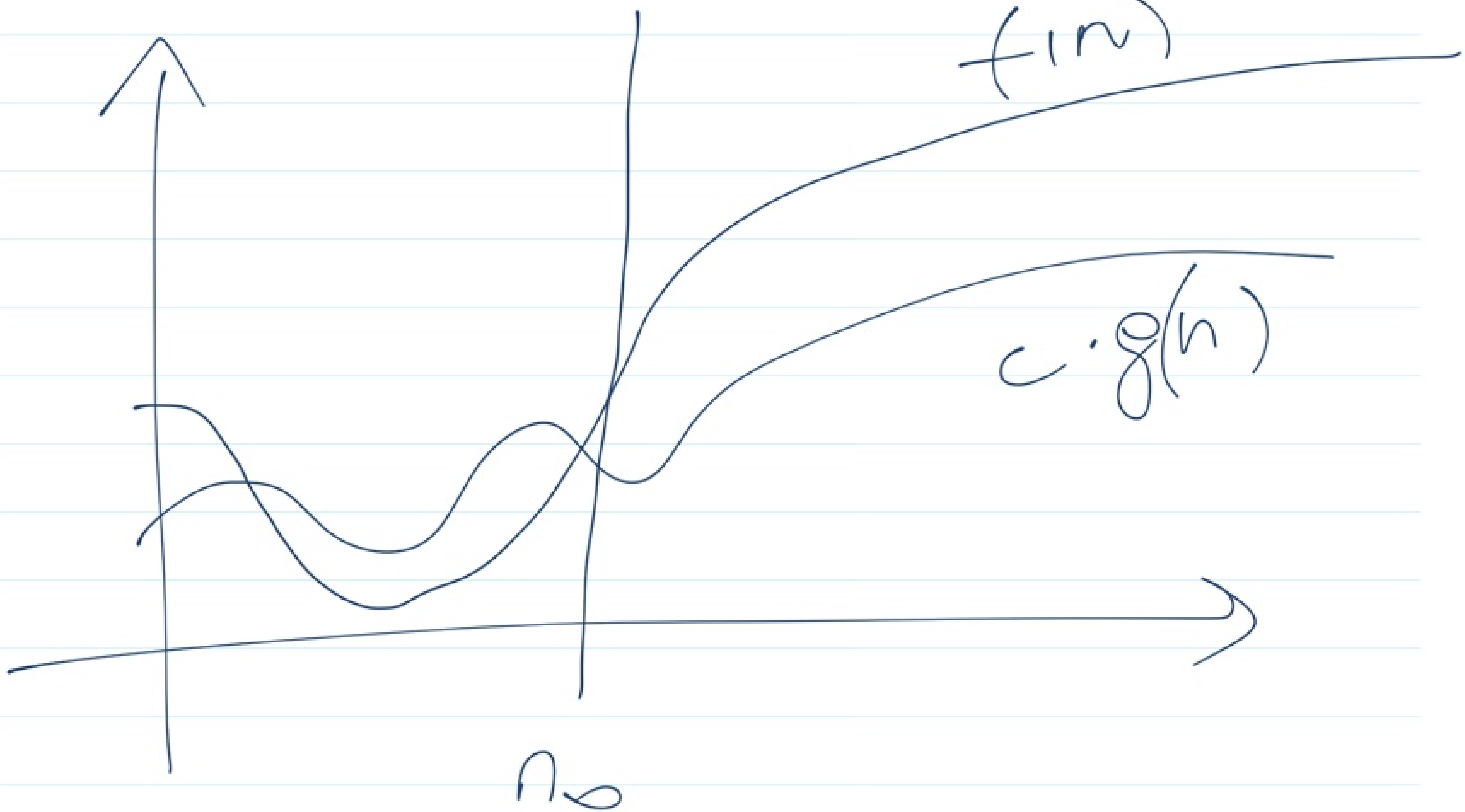
$$\neq O(n)$$

NOTAZIONE Ω $\gtrsim$

limite asintotico inferiore

$$\Omega(g(n)) = \{ f(n) \mid \exists\, c, n_0 > 0 \text{ t.c. } \forall n \geq n_0 \quad 0 \leq c\,g(n) \leq f(n) \}$$

$f(n)$ cresce almeno come $g(n)$, a meno di un fattore costante


f(n)
c·g(n)
n_0

$$
\begin{aligned}
an^2 + bn + c &= \Omega(n^2) \\
&= \Omega(n) \\
&= \Omega(\log n) \\
&\neq \Omega(n^3) \\
&\neq \Omega(n^2 \log n)
\end{aligned}
$$

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

$$f(n) = O(g(n)) \quad \text{AND}$$

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