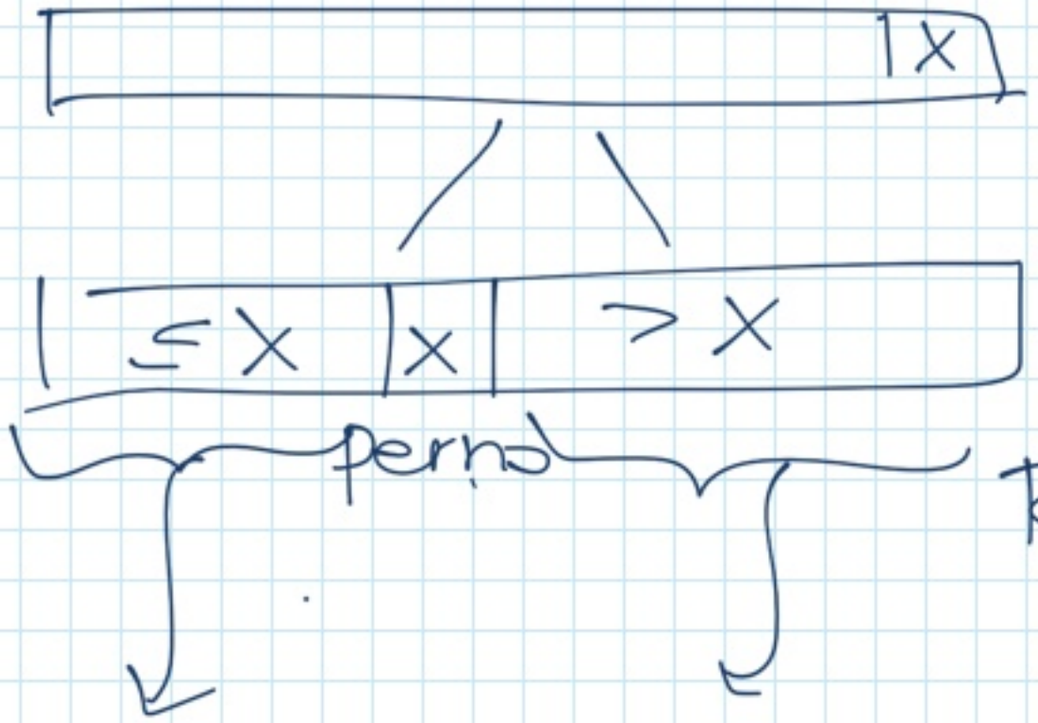
QUICK SORT (Hoare, 1962)

ottimo, ma solo al caso medio

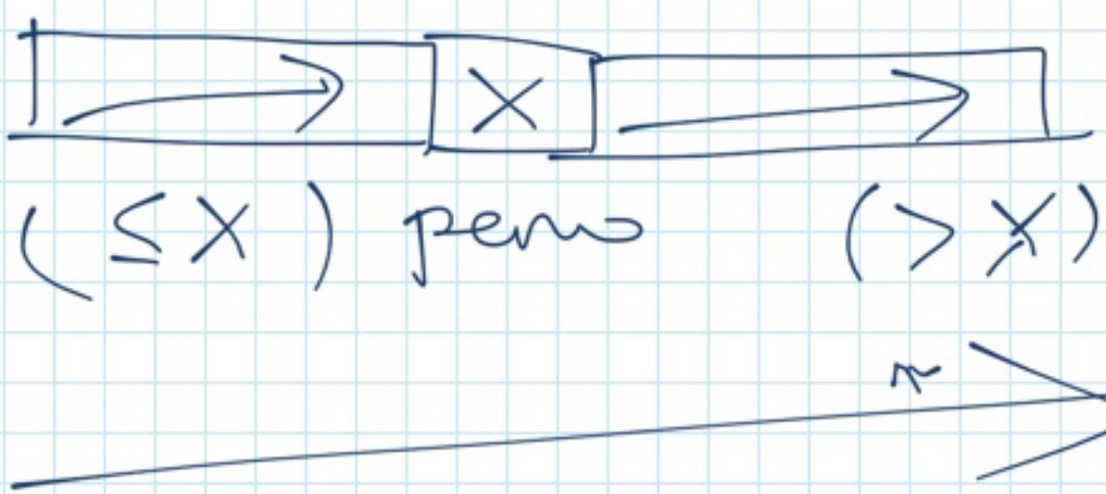
D & I



pivot

DIVISIONE

Ricorsione



Combinazione

vuota

n

divisa

```
QuickSort(a, sx, dx)
{ if (sx < dx) {          // ci sono almeno 2 elementi
```

divide

```
  - scegli px ∈ [sx, dx] come posizione del pivot   // pivot = a[px]
  perno = Distribuzione(a, sx, px, dx);
```

impera

```
  QuickSort(a, sx, perno-1);
  QuickSort(a, perno+1, dx);
}
```

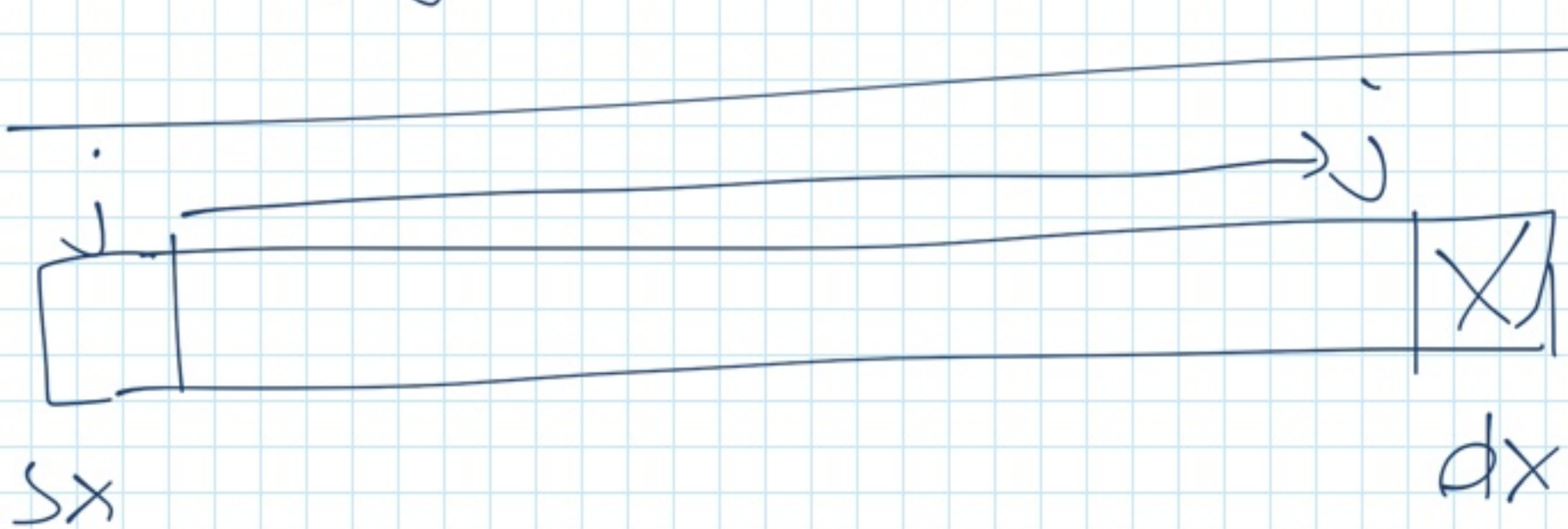
Dopo Distribuzione,

a[perno] = pivot

Scambio (a, i, j)

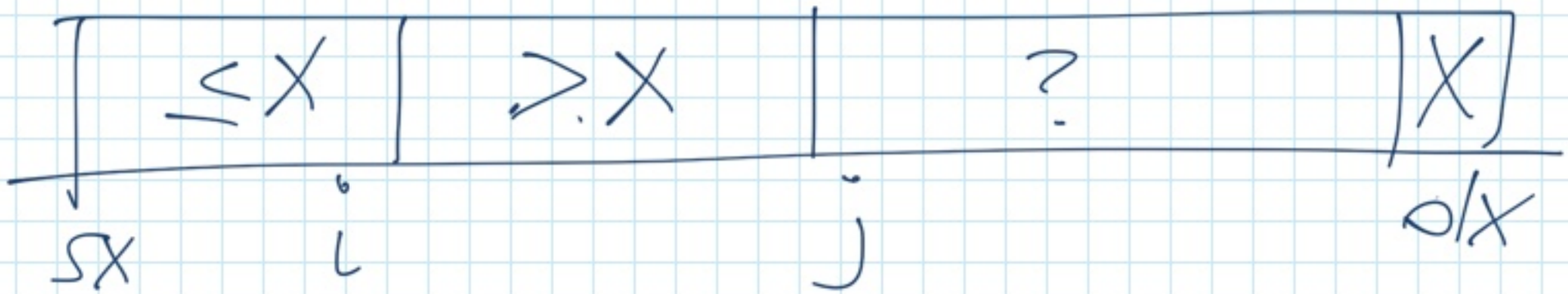
```
temp = a(i);
a(i) = a(j);
a(j) = temp;
```

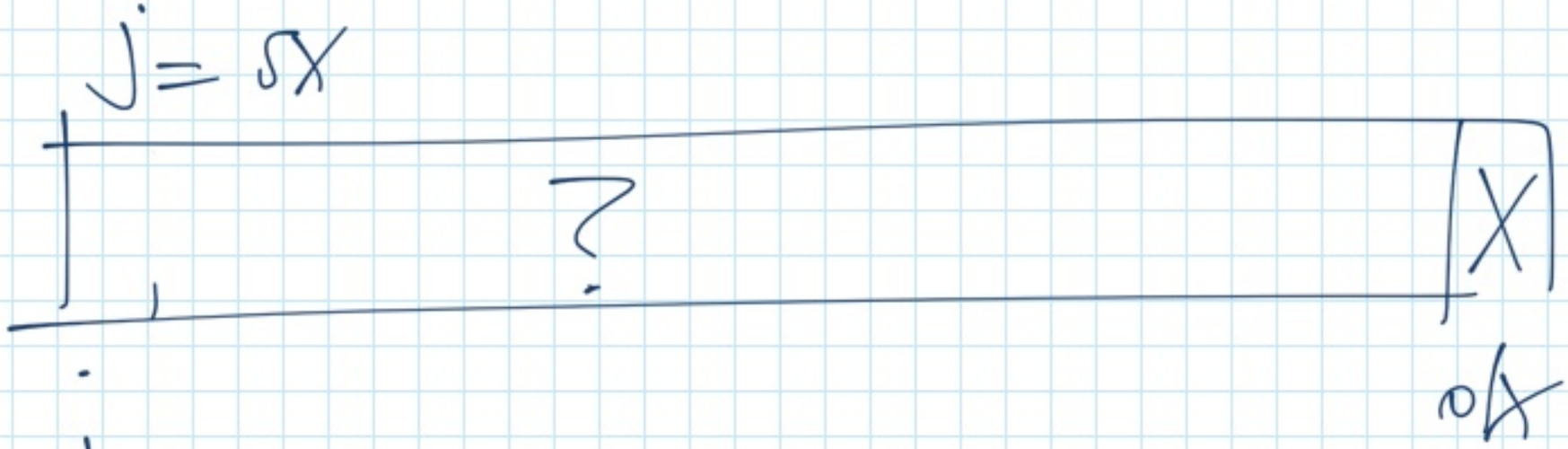
O(1)

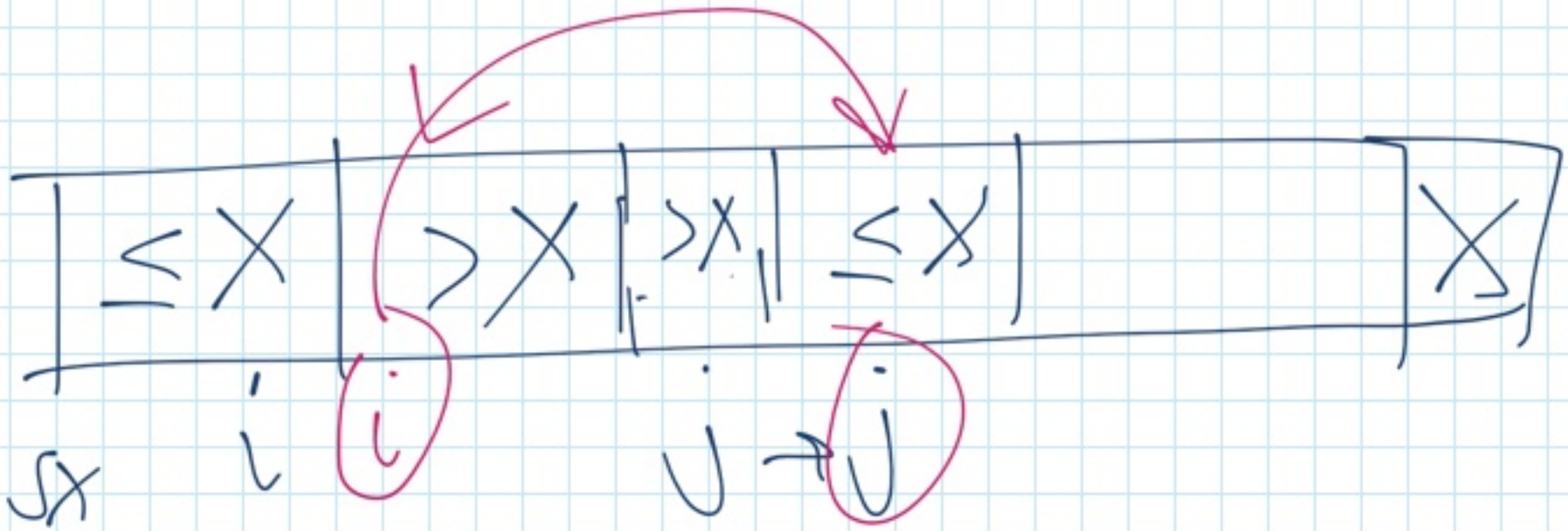


Scambio (a, sx, dx)

j-esima iterazione







a(j) > X

si incremento j

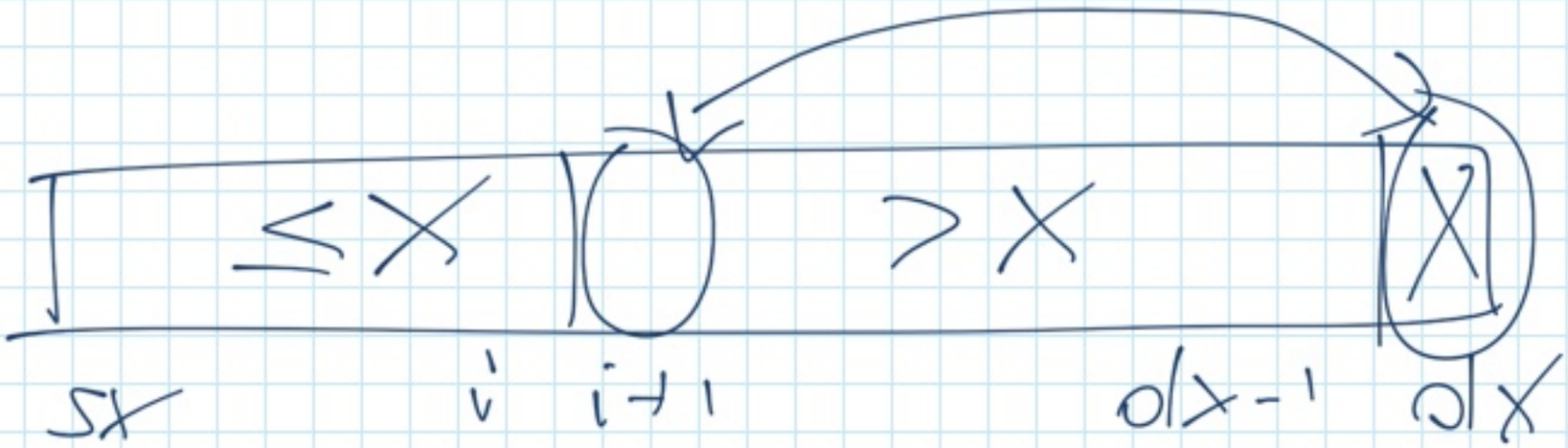
hasta temo l

se a(j) ≤ X

i++;

Distribuzione (a, sx, px, dx)

```
if (px ≠ dx) scambio (a, px, dx);
x = a[dx]   // x è il pivot
i = sx - 1;
for (j = sx; j ≤ dx - 1; j++) {
    if (a[j] ≤ x) {
        i++;
        scambio (a, i, j);
    }
}
i++;
scambio (a, dx, i);
return i;
```



Analisi Distribuzione

Costo in spazio $S(n) \geq O(1)$

n = # elementi in q $(sx - dx)$

$n = dx - sx + 1$

Costo in tempo

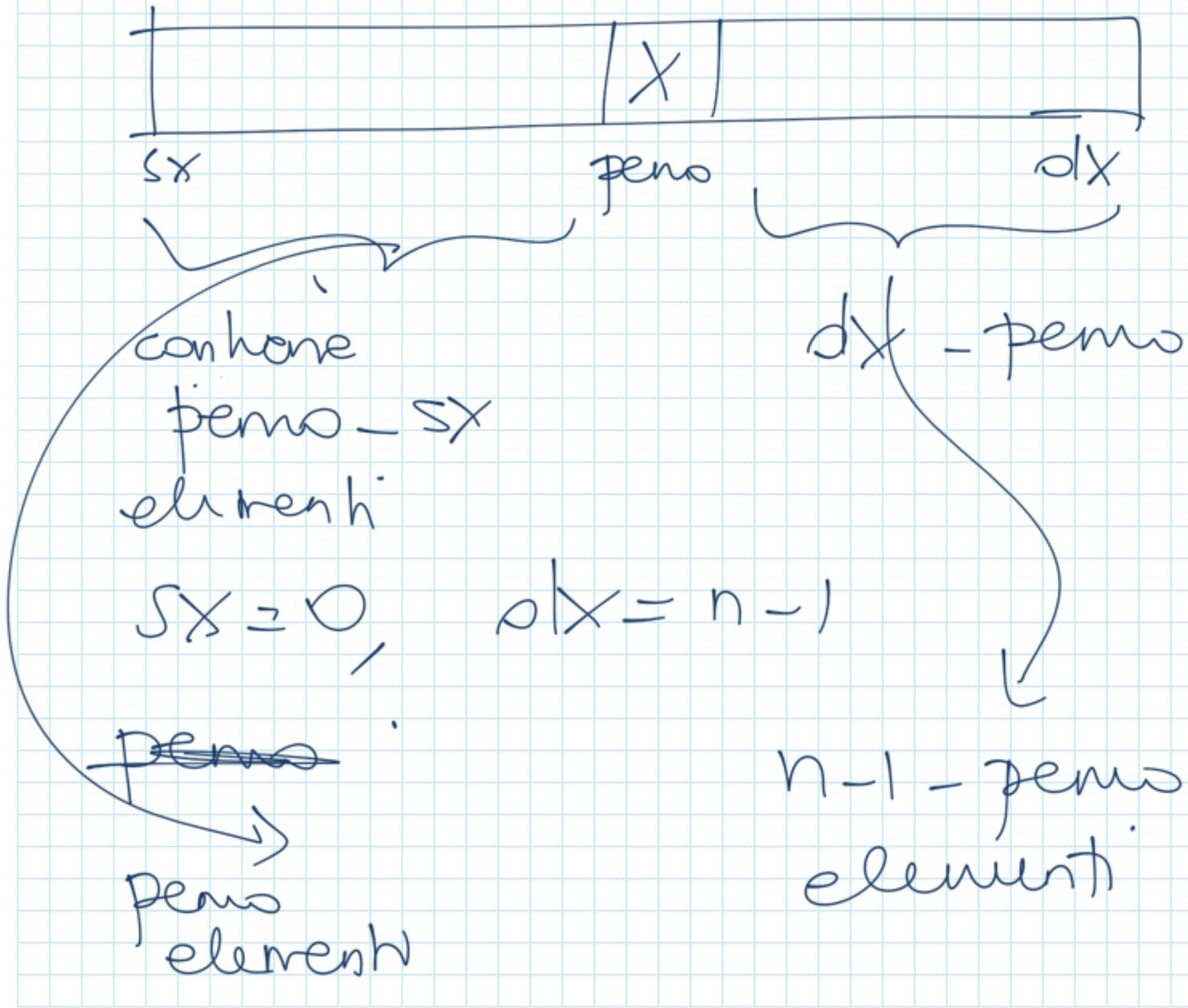
$T(n) = \Theta(n)$

(ad ogni iterazione del ciclo for si esegue un blocco costante)

Confronti

tutti gli elementi della porzione SX della partizione dx - 1 si confrontano (una ed una sola volta) con X. $\Rightarrow \Theta(n-1)$

Analisi del QuickSort



$$T(n) = T(\text{peso}) + T(n - 1 - \text{peso}) + \Theta(n)$$

costo della dichiarazione / chiusura della combinazione

CASO OTTIMO

partizione bilanciata

$$\text{peso} \simeq \frac{n}{2}$$

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

$$\leq 2T(n/2) + \Theta(n) = \Theta(n \log n)$$

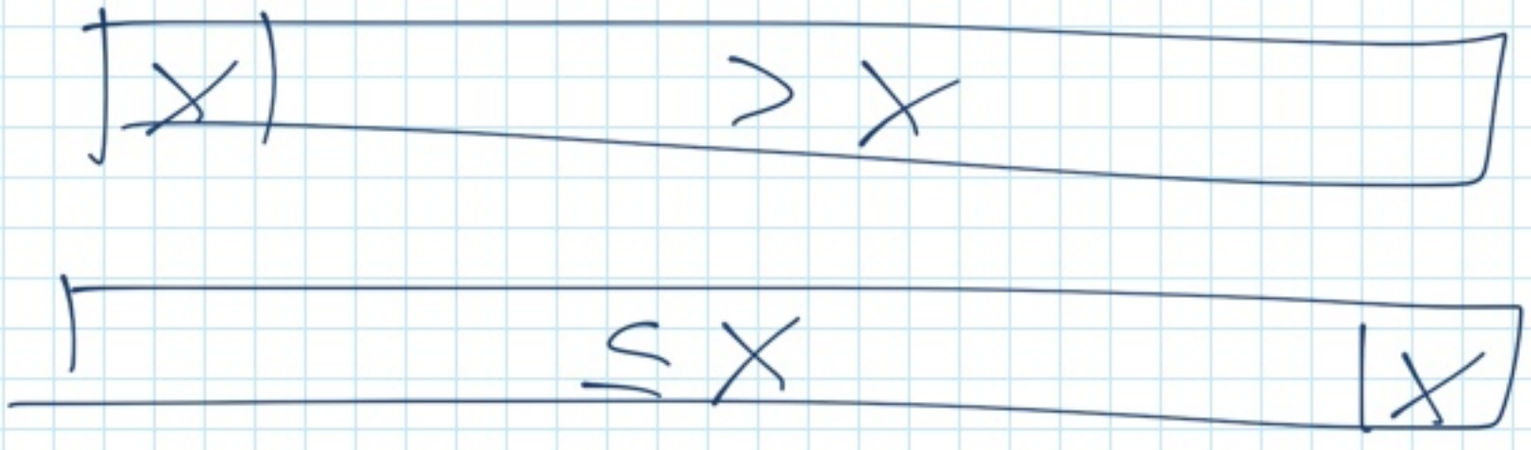
$$T(n) = O(n \log n)$$

CASO PESSIMO

perno = SX (0)

oppure

perno = dx (n-1)



$$T(n) = T(0) + T(n-1) + \Theta(n)$$

Cost

$$T(n) = T(n-1) + \Theta(n)$$

Ipotesi: ad ogni passo di ricorsione scegliamo come pivot il min (max) della porzione di array di lavoro.

$$T(n) = T(n-1) + \Theta(n)$$

$\exists\, c$ costanti

$$T(n) \leq T(n-1) + c \cdot n$$

$$\leq \left(T(n-2) + c \cdot (n-1)\right) + cn$$

$$\leq T(n-3) + c(n-2) + c(n-1) + cn$$

$$\dots$$

$$\leq T(n-i) + c \sum_{k=0}^{i-1} (n-k)$$

se $i = n-1$

$\Rightarrow$

$$T(n) \leq T(1) + c \sum_{k=0}^{n-2} (n-k)$$

$$T(n) \le T(1) + c\sum_{k=0}^{n-2}(n-k)$$

$$= O(1) + c\sum_{k=2}^{n} k$$

$$= O(1) + \left(\frac{n(n+1)}{2} - 1\right)c$$

$$= O(1) + \Theta(n^2)$$

$$T(n) = O(n^2)$$

Confronti al caso pessimo

Dim	# Confronti
n	$n-1$
$n-1$	$n-2$
$n-2$	$n-3$
$\vdots$	$\vdots$
2	1

$$\frac{n(n-1)}{2} = \binom{n}{2}$$

Analisi del caso medio

indipendente dall'ordine iniziale degli elementi in a, e basata sulla casualità

↳ versione randomizzata del QuickSort

↓ il pivot si sceglie a caso in a(sx ... dx)

Scelta del pivot
- casuale
- equiprobabile
- uniforme

⇒ ogni elemento in a(sx..dx) può essere scelto come pivot con pari probabilità

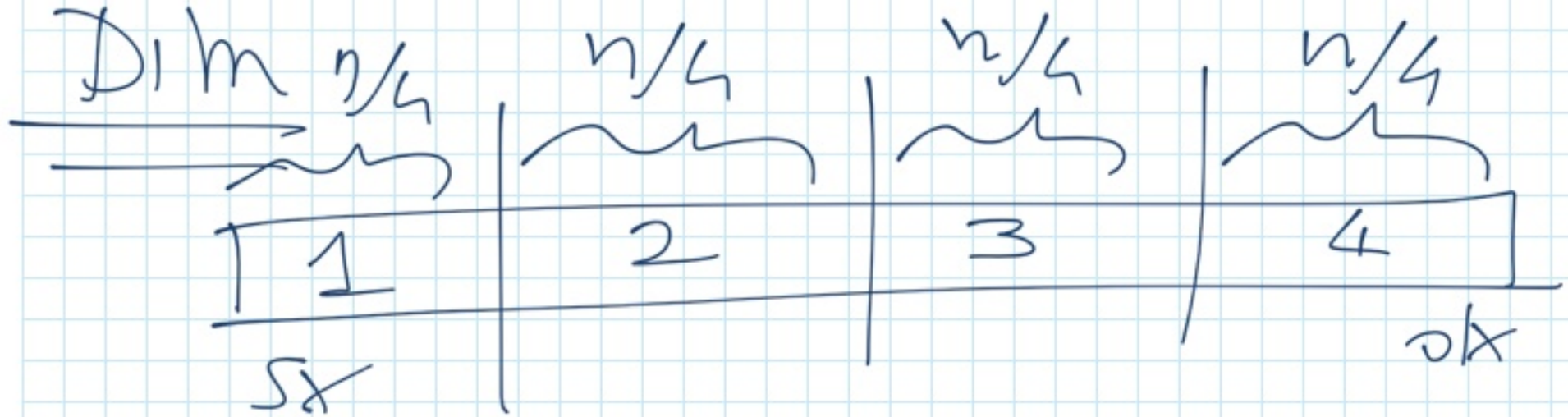
⇒ <u>perno</u> è distribuito uniformemente in [sx, dx)

RandomQuickSort (a, sx, dx)

```
if (sx < dx) {
    px = Random (sx, dx);
    pemo = Dishusjione (a, sx, px, dx)
    RandomQuickSort (a, sx, pemo - 1)
    RandomQuickSort (a, pemo + 1, dx)
}
```

TEOREMA

RandomQuickSort impiega tempo ottimo $O(n \log n)$ al caso medio per ordinare n elementi.



π

EVENTO "perno esterno"

la posizione perno ricade nella zona 1 o nella zona 4

Evento "perno interno"

perno ricade nella zona 2 o nella zona 3

"perno interno" e "perno esterno" sono eventi equiprobabili

$$Pr\{\text{perno interno}\} = Pr\{\text{perno esterno}\} = 1/2$$

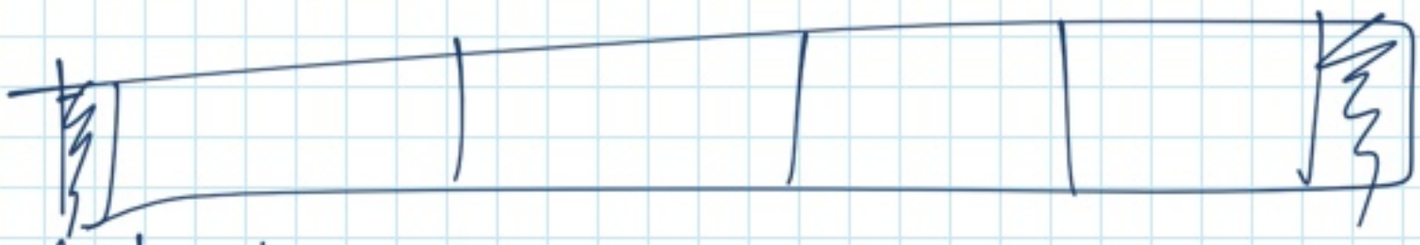
X = Costo del QS quando si verifica l'evento perno esterno

Y = Costo del QS quando si verifica l'evento perno interno

$$T(n) = \frac{1}{2}X + \frac{1}{2}Y$$

Costo medio

X ciclo esterno



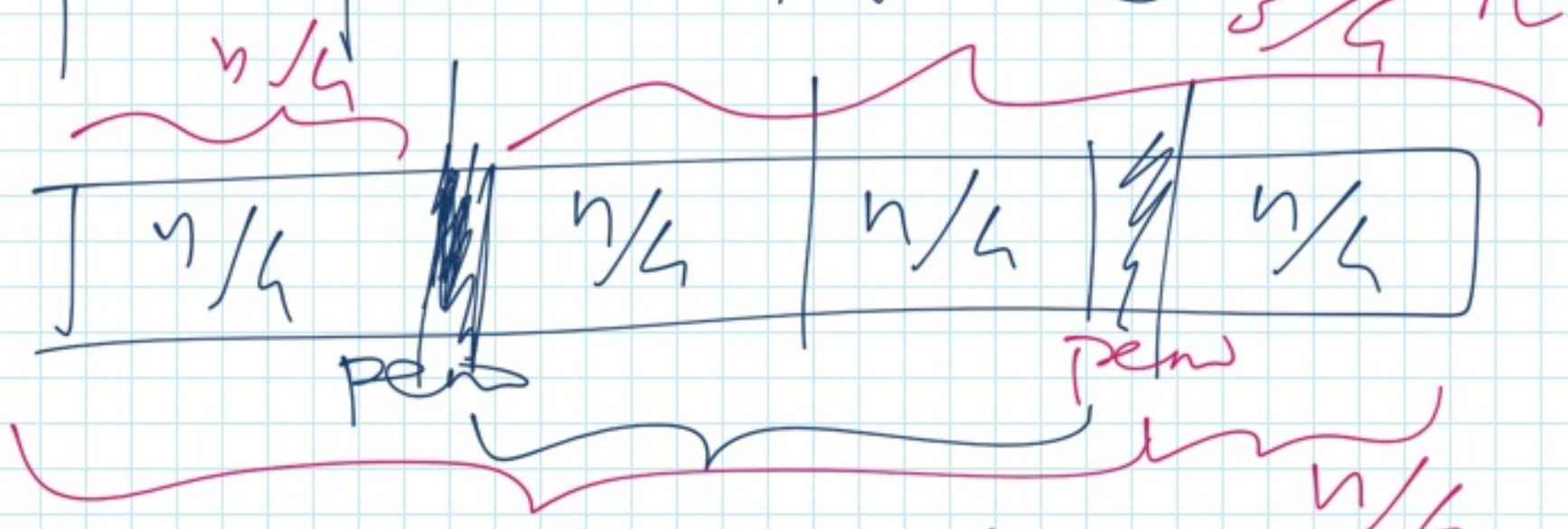
↑ ciclo interno favorevole

$X \leq T(n-1) + T(0) + \Theta(n)$

$X \leq T(n-1) + \Theta(n)$

$\boxed{X \leq T(n) + \Theta(n)}$

Y perno interno



$\frac{3}{4}n$

Caso + sfavorevole:

perno all'inizio delle zona 2

oppure

perno alla fine della zona 3

$$Y \geq T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) + \Theta(n)$$

$$T(n) = \frac{1}{2}X + \frac{1}{2}Y$$

$$\leq \frac{1}{2}\left(T(n) + \Theta(n)\right) +$$

$$\frac{1}{2}\left(T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) + \Theta(n)\right)$$

$$\leq \frac{1}{2}\left(T(n) + T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right)\right)$$

$$+ \underbrace{\frac{1}{2}\cdot\Theta(n) + \frac{1}{2}\Theta(n)}_{\leq c' \cdot n}$$

c' costante

$$T(n) \leq \frac{1}{2}\left[T(n) + T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right)\right]$$

$2^{\cdot}$ $+ c'n$) $\times 2$

$$2T(n) \leq T(n) + T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) + 2c'n$$

↳ $c \equiv 2c'$

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

livello 0 della ricorsione

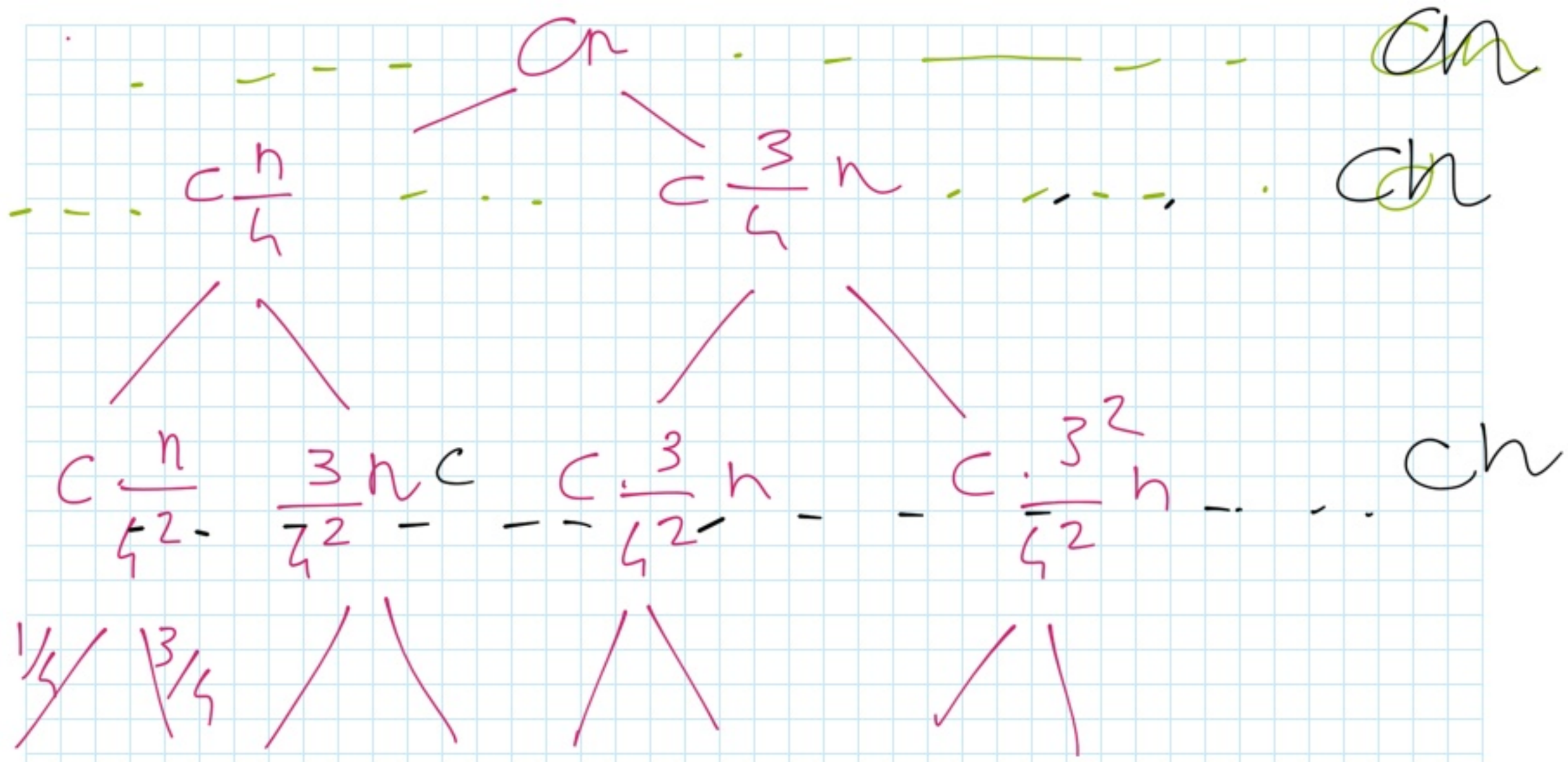
liv. 0 — cn

$1/4$ | $3/4$

liv. 1 — $T\left(\frac{n}{4}\right)$ — $c \cdot \frac{n}{4}$; $T\left(\frac{3}{4}n\right)$ — $c \cdot \frac{3}{4}n$ — cn

$1/4$ | $3/4$ | $1/4$ | $3/4$

$T\left(\frac{1}{4}\cdot\frac{n}{4}\right)$; $T\left(\frac{3}{4}\cdot\frac{1}{4}n\right)$; $T\left(\frac{1}{4}\frac{3}{4}n\right)$; $T\left(\frac{3}{4}\cdot\frac{3}{4}n\right)$



su ogni livello costa ch

livello i

contributo delle 2^i chiamate ricorsive

$$\begin{cases} \geq cn & \text{se nessun sottoproblema è terminale} \\ \leq cn & \text{se qualche chiamata ha raggiunto il caso base} \end{cases} \quad (1/4)$$

$$T(n) \leq c \cdot n \cdot (\#\text{livelli}) \quad (1)$$

livelli ?

livello più profondo si presenta seguendo il ramo 3/4 (conduce all'ultimo sottoproblema che raggiunge il caso base)

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

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

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

$$\log_{\frac{4}{3}}\left(\frac{4}{3}\right)^{i} = \log_{\frac{4}{3}} n$$

$$i = \log_{4/3} n$$

$$T(h) \leq ch \cdot \#livelli$$

$$\#livelli = \log_{\frac{4}{3}} n$$

$$\Rightarrow T(n) \leq ch \cdot \log_{\frac{4}{3}} n$$

$$\boxed{T(n) = O(n \log n)}$$