STABILITÀ di un algoritmo di ordinamento

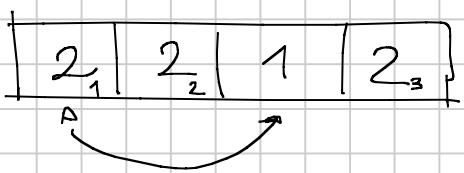
Un alg. di ord. è stabile se lascia invariato l'ordine degli elementi con lo stesso valore

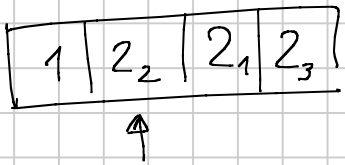
↳ importante in presenza di dati satelliti

STABILI: Insertion Sort ✓
Merge Sort ✓

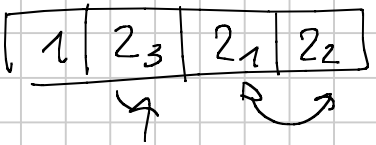
NON STABILI: Selection Sort, HeapSort, QuickSort

SELECTION SORT



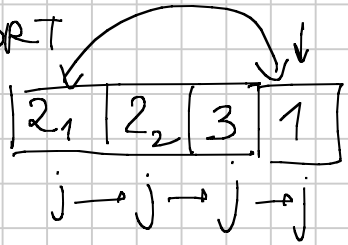


<

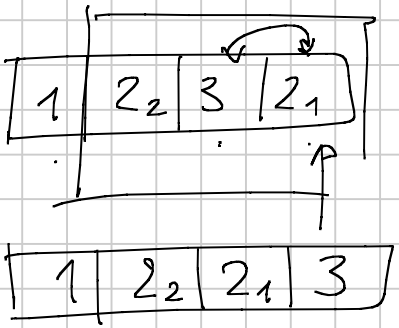


≤

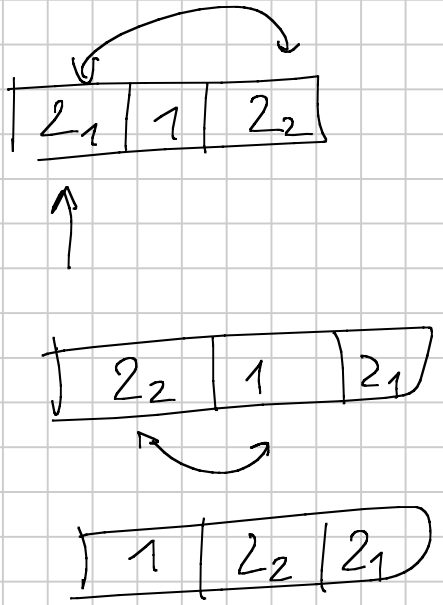
QUICK SORT

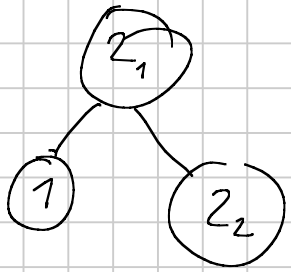


→

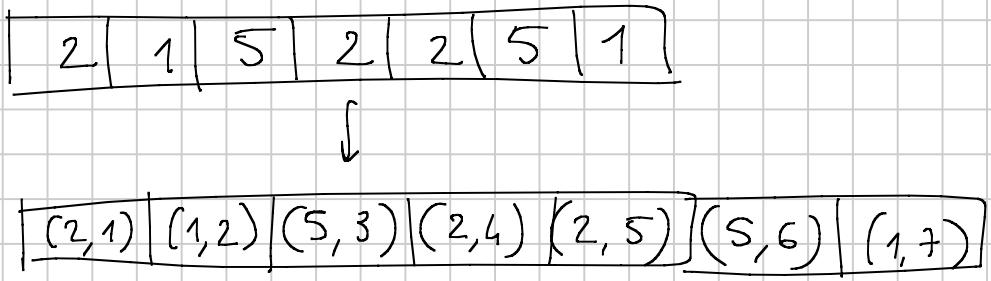


HeapSort





Tecnica x stabilità



$(2,1) < (2,4)$

$T(n)$ NON CAMBIA

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

per scrivere i, $i \leq n$ occorrono $O(\log n)$ bit.

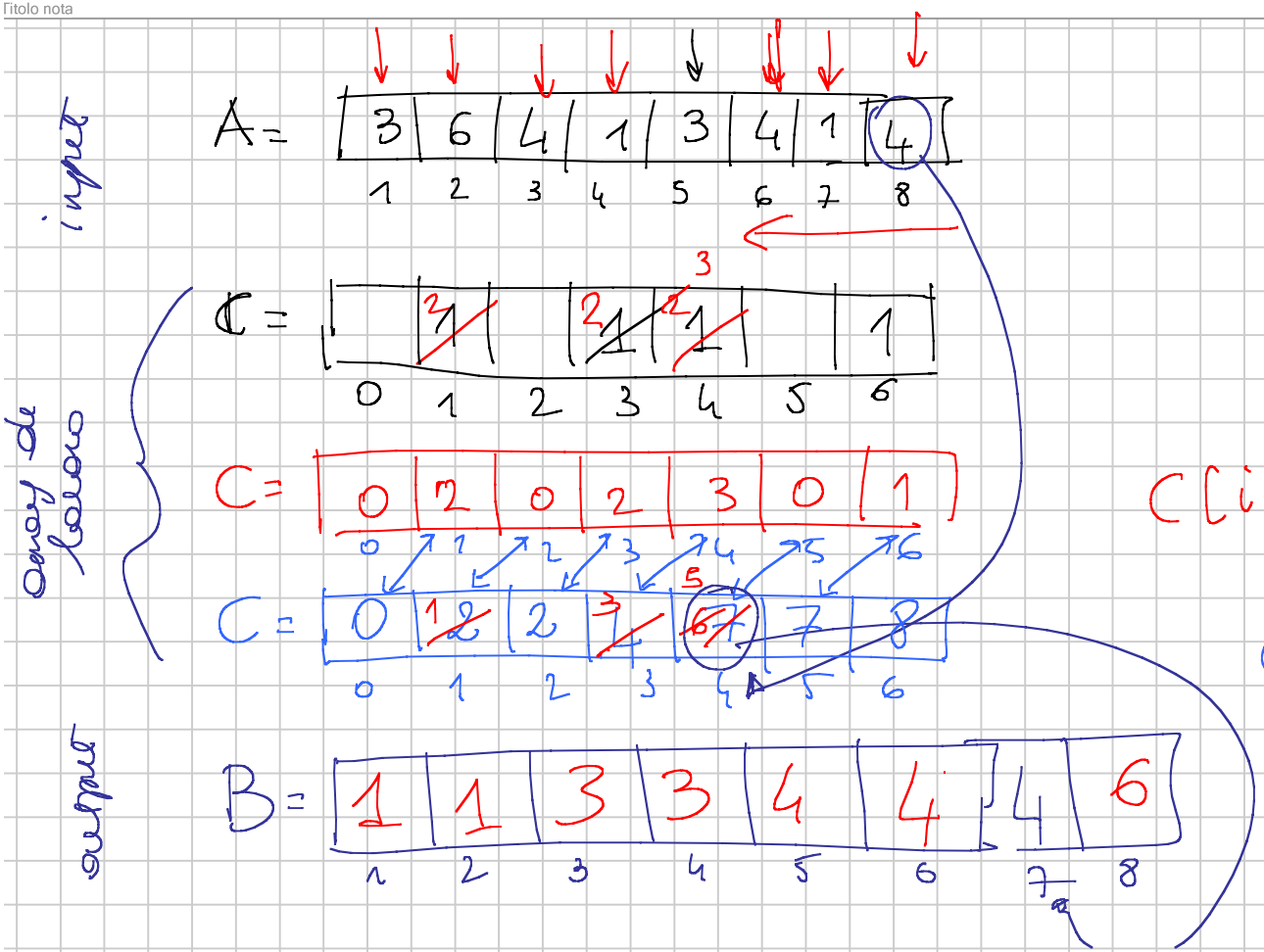
Ordinamenti in tempo lineare

NON ORDINANO X CONFRONTI
il limite inferiore $\Omega(n \log n)$ NON si applica.

COUNTING SORT

A: array di n interi

$\forall i \quad 0 \leq A[i] \leq k$



$k = 6 \qquad n = 8$

dim. $k+1$

$[0, k]$

$C[i]$ = # elementi di A di valore i

$C[i]$ = # elementi di A di valore $\leq i$

CountingSort (A, B, k)

```
C = nuovo array di dim k+1
for j = 0 to k                                   Θ(k)
    C[j] = 0
for i = 1 to n                                   Θ(n)
    C[A[i]]++
for j = 1 to k                                   Θ(k)
    C[j] = C[j] + C[j-1];
for i = n downto 1 {                             Θ(n)
    B[C[A[i]]] = A[i];
    C[A[i]]--;
}
```

// C[j] = # di elementi di A di valore = j

// C[j] = # elementi di A di valore ≤ j

Analisi

$$T(n,k) = \Theta(k) + \Theta(n) + \Theta(k) + \Theta(n) = \Theta(n+k)$$

$$\boxed{T(n,k) = \Theta(n+k)}$$

$k = O(n) \Rightarrow T(n,k) = \Theta(n)$ lineare!

(e stabile)

$S(n,k) = \Theta(n+k)$

non ordina in loco

Radix Sort

ordinamento per cifre, a partire dalla cifra meno significativa.

$d = \#$ cifre

329		720		720			329
457		355		329			355
657		436		436			436
839	→	457	→	839	→		457
436		657		355			657
720		329		457			720
355		839		657			839
↑		↑		↑			

È facile usare un ordinamento stabile.

RadixSort (A, d) d = # cifre

for i = 1 to d {

ordino A sulla cifra i usando un algoritmo di ordinamento stabile

}

i = 1 cifra meno significativa

i = d cifra più significativa

CORRETTEZZA (per induzione sul numero d di cifre)

BASE d = 1 ✓

IPOTESI INDUTTIVA

RS corretto sulle d − 1 cifre meno significative

PASSO: ordinamento rispetto alla cifra più significativa d.

d − 1 → d

2 casi

a, b due elementi dell'array

a_d = d-esima cifra + sign. di a

b_d = " " " di b

① $a_d \neq b_d$

↓ ordinare rispetto alla cifra d è corretto

infatti

$a_d < b_d$ ⇒ a prima di b nell'array ordinato

(corretto perché $a < b$)

$a_d > b_d$ ⇒ b prima di a,

Corretto perché $b < a$

① 99 < ② 200

(2) $a_d = b_d \Rightarrow$

l'ordinamento delle cifre è stabile.

a e b conservano l'ordine in cui erano nell'array dato dalle altre $d-1$ cifre.

↳ corretto x i.i.

□

Analisi di complessità

(ordinamento delle cifre con COUNTING SORT)

$d = \#$ cifre

(1) - base di rappresentazione costante (b)

- le cifre possono assumere b valori diversi $[0, b-1]$

valore massimo cifre $= b-1 = \Theta(1)$

$$T_{RS}(n) = \Theta(d(n+b)) = \Theta(dn)$$

($b \to \Theta(1)$)

se anche d è costante $\Rightarrow$ $T_{RS}(n) = \Theta(n)$

(2) i numeri da ordinare hanno valore $\leq k$
(non le singole cifre)

se $k = O(n)$ $\Rightarrow$ uso direttamente il Counting Sort sugli elementi da ordinare

$\boxed{k = \Omega(n)}$

$T_{RS}(n) = \Theta(d(n+b))$

$d \leq \lfloor \log_b k \rfloor + 1 = \Theta(\log_b k)$

$d = O(\log_b k)$ b = costante $\Rightarrow$ $d = O(\log k)$

$$T_{RS} = \Theta(d(n+b)) = O(n \cdot \log k)$$

$b = O(1)$
$d = O(\log k)$

$k = \Omega(n)$ ⇒ conviene Alg. di ordinamento per confronti

$k = O(n)$ ⇒ conviene il counting sort

Problema: troppe celle

(3) Usiamo una base b non costante (ordino per "gruppi" di celle, non per singole celle)

$b = n$

n numeri, di valore al più k, $k \geq n$

$$d \leq \lfloor \log_n k \rfloor + \cancel{1} = \Theta(\log_n k) = \Theta\left(\frac{\log k}{\log n}\right)$$

d: valore massimo delle celle $= n - 1$

il counting sort sulle celle (in base n) è lineare

$$T_{RS}(n) = \Theta(d(n+n)) = \Theta(dn)$$

$$d = O\left(\frac{\log k}{\log n}\right)$$

$$\Rightarrow \boxed{T_{RS}(n) = O\left(n \frac{\log k}{\log n}\right)}$$

È efficiente?

$k = \Theta(n)$ $\quad T_{RS}(n) = \Theta(n)$ (una sola cella Counting Sort)

$k = \Theta(n^t)$ $\quad t$ costante

$$T_{RS}(n) = O\left(n \cdot \frac{\log n^t}{\log n}\right) = O\left(n \cdot t \cdot \frac{\log n}{\log n}\right) \underset{t=O(1)}{=} \boxed{O(n)}$$

$k = 2^n$

$$T_{RS}(n) = O\left(n \frac{\log k}{\log n}\right) = O\left(n \frac{n}{\log n}\right) = O\left(\frac{n^2}{\log n}\right)$$

NON VA BENE
meglio un algoritmo ottimo per confronti

Ordinare
N numeri a 32 bit

$n = 10^6$

$2^{19} < n < 2^{20}$

$b = 2^{16}$

$\Rightarrow$ # cifre $:= 2$

$\Rightarrow$ 2 sole passate di Counting Sort ordinano in tempo lineare
10^6 numeri a 32 bit