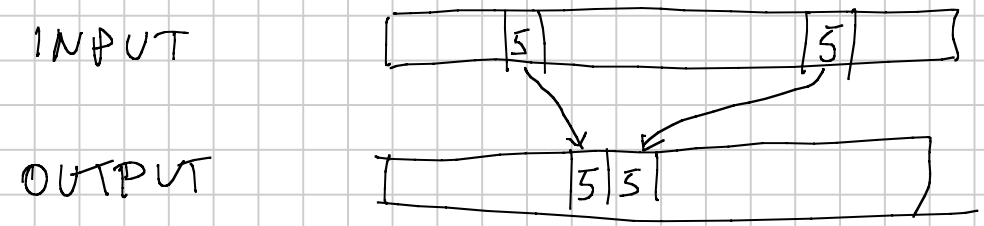
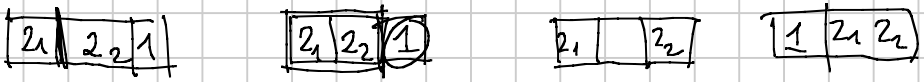
Algoritmo di Ordinamento si dice stabile se se si mantiene l'ordinamento iniziale degli elementi ripetuti



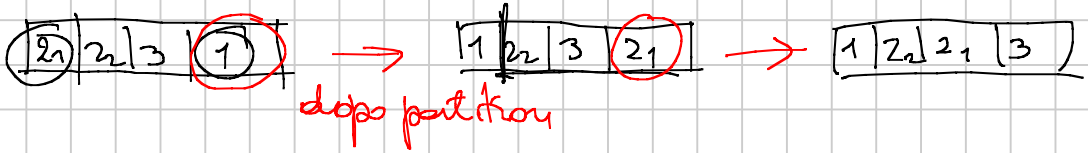
INSERTION SORT sì nella versione Cormen

A 

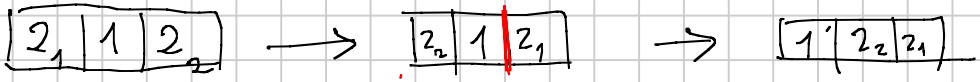
SELECTION SORT sì

MERGE SORT sì

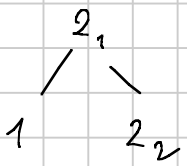
QUICK SORT no nella versione di Partitur Cormen

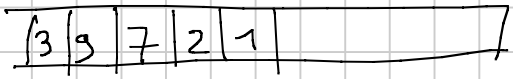


Heapsort no



HeapSort



INPUT 

$\Theta(n)$ tempo

3.1 9.2 7.3 2.4 9.5

Possono diventare stabili $S(n) = \Theta(n \log n)$ spazio

spazio etichette

- trasformazioni locali
- assegnando un'etichetta numerica a tutti gli el da ordinare

SORTING: limite inferiore $\Omega(n \log n)$ confronto tra elementi

SORTING IN TEMPO LINEARE

COUNTING_SORT(A) pre: $\forall i, A[i] \leq K$ $K = O(n)$

- alloca un array C di K el.
- in C si calcolano le frequenze di ciascun el. A[i]
- {si calcola in C quanti el. sono $\leq$ dell'el. A[i]} = somme prefisse
- si costruisce B[1..n] che contiene l'ordinamento sulla base di C

COUNTING_SORT (A, B, k): input: A di n el: $A[i] \le k$ $(k = O(n))$

"sia C [0..k] nuovo array;" output: B

$\Theta(k)$ for i = 0 to k C[i] = 0;

$\Theta(n)$ for j = 1 to n C[A[j]]++;

$\Theta(k)$ for i = 1 to k C[i] = C[i] + C[i-1];

$\Theta(n)$ for j = n downto 1 (for (j = 1 to n)

B[C[A[j]]] = A[j];

C[A[j]]--;

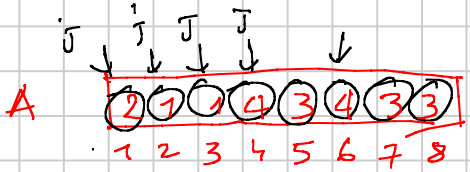
$T_{cs}(n) = \Theta(n+k)$

se $k = O(n)$

$T_{cs} = \Theta(n)$

NO stabilito

$S_{cs}(n) = \Theta(k+n)$



k = 4

C | 0 | 2 | 1 | 3 | 2 | ⇒

0 1 2 3 4

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

1	1	2	3	3	3	4	4	B
1	2	3	4	5	6	7	8	

COUNTING_SORT (A, K) se $K = O(n \log n)$?

$T_{CS}(n) = \Theta(n + n\log n) = \Theta(n \log n)$

è conveniente usarlo se K è $o(n \log n)$

↑

o piccolo

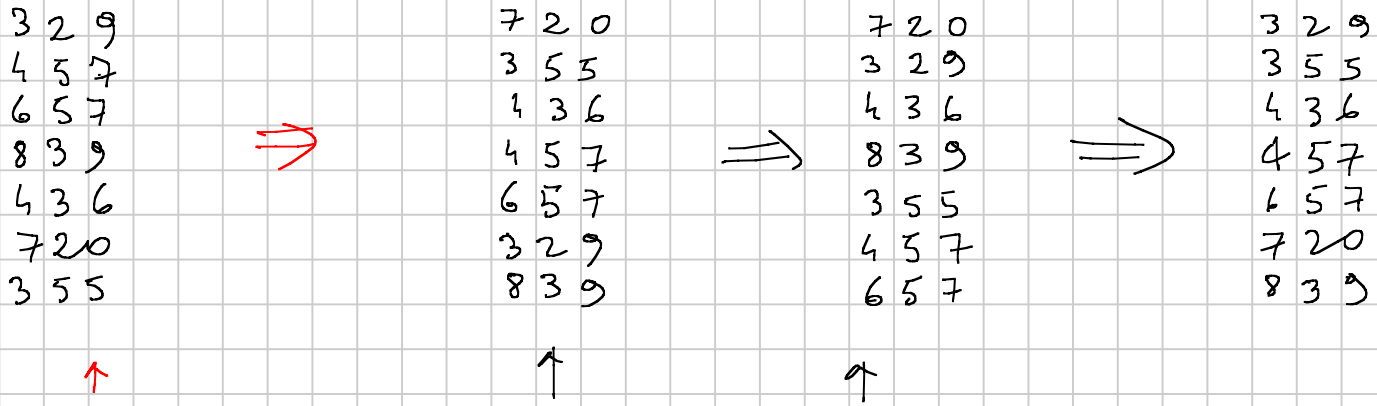
È stabile nella versione del Cormen

RADIX SORT

basato su confronti di cifre (non di elementi).

Si esegue un ordinamento STABILE sulla singola cifra a partire dalla meno significativa.

d = numero cifre n = numero ll.



RADIX-SORT (A, d)
for i = 1 to d
    "ordino A sulla cifra d
    usando ordinamento stabile"

usiamo COUNTING-SORT perché max value = t e usiamo

$\Theta(d(n+t))$ se d è costante

cifre decimali

RADIX SORT richiede tempo lineare

Correttezza di RADIX-SORT

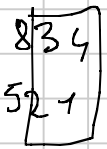
Si dimostra per induzione sulle cifre da ordinare $1 \leq i \leq d$

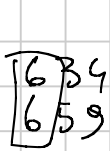
caso base: $i = 1$ $\Rightarrow$ sorting stabile vero

passo induttivo $1, \dots, i-1 \rightarrow i$.

$i \; | \; i-1 \dots 1$

- Se due cifre in posizione i sono diverse l'ordine tra le due cife al passo i è indipendente da quello che è successo prima. vero
- Se due cifre in posizione sono uguali $\rightarrow$ l'ordinamento rimane invariato da quello delle cifre precedenti — vero





Titolo nota 17/03/2016

Caso: $A[i] < k$ RADIX-SORT funziona bene ?

$$d \leq \lfloor \log_{10} k \rfloor + 1 = \Theta(\log k)$$

$$T_{RS} = \Theta(d(n+10)) = \Theta(n \log k)$$

NO $T_{CS}(n) = \Theta(n+k)$

se $k = O(n)$ Counting Sort è migliore

se $k = \Omega(n)$ Merge Sort $\Theta(n \log n)$ è migliore

Radix Sort $\Theta(n \log k)$ $\Omega(n)$

IDEA: invece di usare cifre singole si usano gruppi di cifre.

$\longleftrightarrow$ cambiare base

$$\log_n K = \frac{\log K}{\log n}$$

n numeri $\forall i \quad A[i] < K \qquad 1 \le i \le n$

Considero i numeri in base n

$n = 100$ numeri, ognuno $< 10\,000$ 8356 ogni cifra è $O(n)$

$$d \le \left\lfloor \log_{\underset{100}{n}} K \right\rfloor + 1 = 2 \text{ nell'esempio}$$

$$T_{RS} = \Theta(dn + d\underbrace{t}_{O(n)}) = O(dn) = O\left(\frac{\log K}{\log n} n\right) = O\left(n \frac{\log K}{\log n}\right)$$

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

Titolo nota 17/03/2016

Merge Sort $\Theta(n \log n)$ Radix Sort $O\left(n \frac{\log K}{\log n}\right)$.

1) $K = \Theta(2^n)$

$MS = \Theta(n \log n)$ Radix Sort $= O\left(\frac{n^2}{\log n}\right)$

2) $K = \Theta\left(2^{(\log n)^2}\right)$ $\log K = (\log n)^2$

$MS = \Theta(n \log n)$ RADIX Sort $O(n \log n)$

3) $K = \Theta(n^t)$ con t costante

$MS = \Theta(n \log n)$ R_S $O\left(nt \frac{\log n}{\log n}\right) = O(nt)$