

Algoritmi e Strutture Dati

Lezione 18

9 novembre 2022

Ordinare senza confrontare

Problema: Ordinare n interi in $[0..b-1]$

A

1 0	0 0	4 1	2 2	0 2	2 4
-------------------	-------------------	-------------------	-------------------	-------------------	-------------------

Y

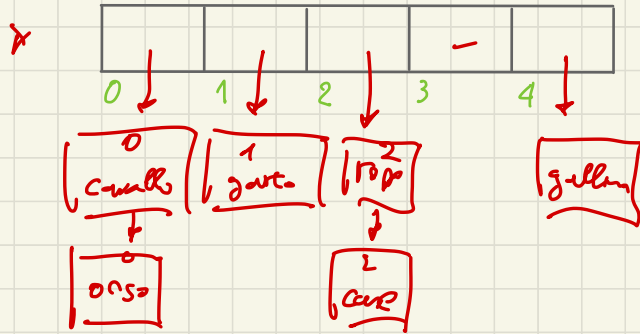
2	1	2	0	1
0	1	2	3	4

integerSort

Problema: Ordinare n record con chiavi intere in $[0..b-1]$

A

0	0	1	2	2	4
Cantale	Canale	galle	topo	comp	galler



bucket Sort

Problema: Ordinare n record con chiavi intere in $[0..b-1]$

ALGORITMO bucketSort (Array $A[0..n-1]$, intero b)

Sia $Y[0..b-1]$ un array

FOR $i \leftarrow 0$ TO $b-1$ DO

$Y[i] \leftarrow$ coda vuota

predisposizione
bucket

$\Theta(b)$
passi

FOR $i \leftarrow 0$ TO $n-1$ DO

$x \leftarrow A[i].\text{chiave}$

$Y[x].\text{enqueue}(A[i])$

riempimento
bucket

$\Theta(n)$
passi

$j \leftarrow 0$

FOR $i \leftarrow 0$ TO $b-1$ DO

 WHILE NOT $Y[i].\text{isEmpty}()$ DO

$A[j] \leftarrow Y[i].\text{dequeue}()$

$j \leftarrow j+1$

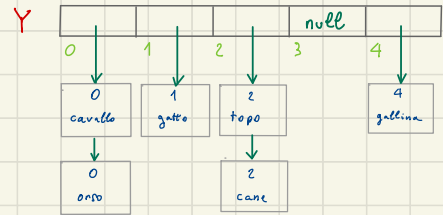
riempimento
array
ordinato

$\Theta(b+n)$
passi

$\Theta(b+n)$

A

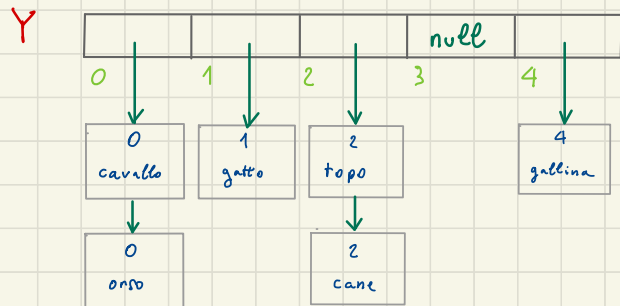
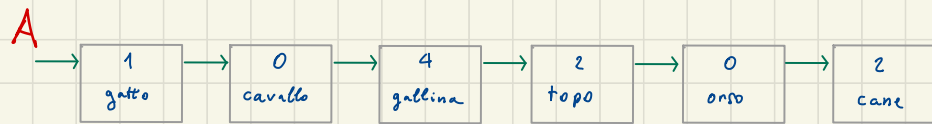
1	0	4	2	0	2
gatto	cavallo	gallina	topo	orso	cane



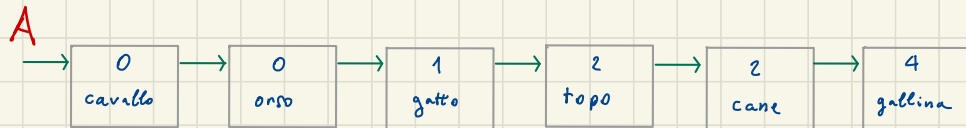
A

0	0	1	2	2	4
cavallo	orso	gatto	topo	cane	gallina

BucketSort può essere utilizzato anche per ordinare liste
manipolando direttamente i puntatori



colloca ciascun
nodo di A nella
coda corrispondente
alla chiave



concatena
le liste (code)
una dopo
l'altra

BucketSort

- Tempo $\Theta(n+b)$

→ se $b = \Theta(n)$

Tempo $\Theta(n)$

→ se $b = \Theta(n^2)$

Tempo $\Theta(n^2)$

↑
merge
sort
heapSort

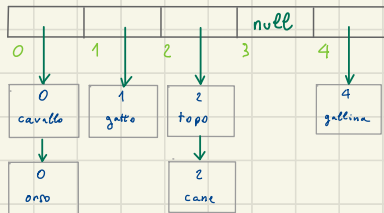
- NON IN LOCO

- Stabile

- Utilizzabile anche per ordinare

liste (stessa complessità in tempo)

1	0	4	2	0	2
gatto	cavallo	gallina	topo	orso	cane



0	0	1	2	2	4
cavallo	orso	gatto	topo	cane	gallina

Problema: ordinare un insieme di persone rispetto alla data del compleanno

Alberto 22 gennaio

Anna 9 giugno

Claudia 27 aprile

Franco 31 maggio

Lorenzo 15 gennaio

Lucia 14 dicembre

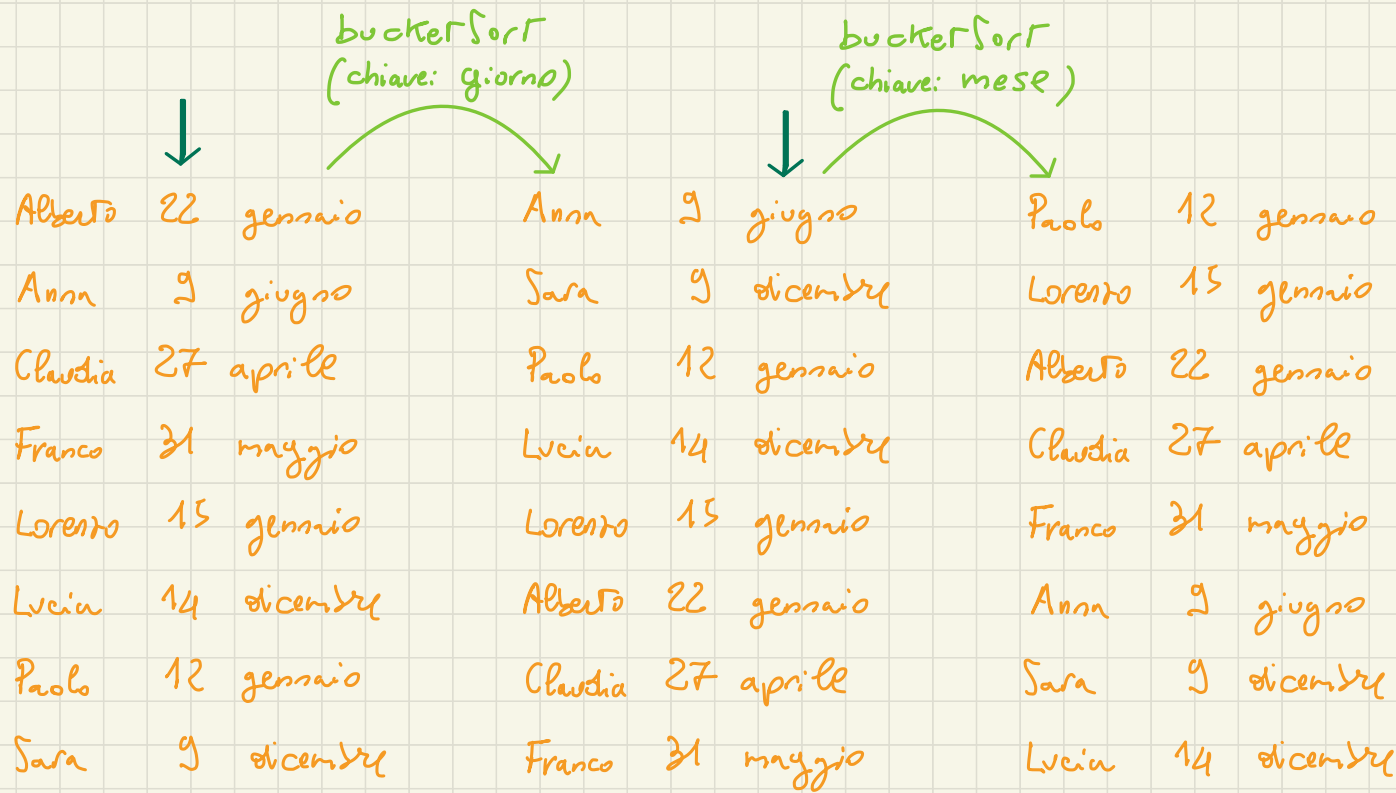
Paolo 12 gennaio

Sara 9 dicembre

① bucket Sort su giorni

② bucket Sort su mesi

Problema: ordinare un insieme di persone rispetto alla data del compleanno



Problema: ordinare un insieme di dati con chiavi intere

Radix Sort

↓
1 2 3 4
3 7
2 2 4
5 1 3 1
4 1
4 8
4 2 3 7

↓
5 1 3 1
4 1
1 2 3 4
2 2 4
3 7
4 2 3 7
4 8

↓
2 2 4
5 1 3 1
1 2 3 4
3 7
4 2 3 7
4 1
4 8

↓
3 7
4 1
4 8
5 1 3 1
2 2 4
1 2 3 4
4 2 3 7

3 7
4 1
4 8
2 2 4
1 2 3 4
4 2 3 7
5 1 3 1

ALGORITHM radixSort (Array $A[0..n-1]$)

$t \leftarrow 0$

WHILE (\exists chiave K in A t.c. $\lfloor K/b^t \rfloor \neq 0$) DO

 | bucketSort (A, b, t)

 | $t \leftarrow t+1$

↑
A contiene una chiave
composta da almeno
 $t+1$ cifre

PROCEDURA bucketSort (Array $A[0..n-1]$, intero b , intero t)

Sia $Y[0..b-1]$ un array

predisposizione

FOR $i \leftarrow 0$ TO $b-1$ DO

bucket

 | $Y[i] \leftarrow$ coda vuota

FOR $i \leftarrow 0$ TO $n-1$ DO

 | $x \leftarrow$ cifra di posto t nella
 rappresentazione in base b
 di $A[i]$. chiave

riempimento
bucket

 | $Y[x].enqueue(A[i])$

$j \leftarrow 0$

FOR $i \leftarrow 0$ TO $b-1$ DO

riempimento
array
ordinato

 | WHILE NOT $Y[i].isEmpty()$ DO

 | $A[j] \leftarrow Y[i].dequeue()$

 | $j \leftarrow j+1$

bucketSort(10, 0)

bucketSort(10, 1)

bucketSort(10, 2)

bucketSort(10, 3)

1234	5131	224	37	37
37	41	5131	41	41
224	1234	1234	48	48
5131	224	37	5131	224
41	37	4237	224	1234
48	4237	41	1234	4237
4237	48	48	4237	5131

RadixSort

- Se devo ordinare n chiavi tra 0 e 999'999'999:
con $B = 10^3$ bastano 3 passate di bucketSort

Meglio usare B potenza di 2 (divisioni più veloci!)

- Se devo ordinare n chiavi su 32 bit, cioè tra 0 e $2^{32}-1$
 - con $B = 2^8$: $\rightarrow 256$ bucket 4 passate di bucketSort
 - con $B = 2^{16}$: 2 passate di bucketSort

Tempo $\Theta(n)$

Come ottenere la cifra di posizione* t di x in base b ?

es base 10, $t=2$

$x = 1234$

$(x/10^2) \text{ MOD } 10$ (divisione intera)

$x = 12345$

$\rightsquigarrow 12345/10^2$
123

in generale: $(x/b^t) \text{ MOD } b$ (divisione intera)

* contando da 0 dalla meno significativa

Come ottenere la cifra di posizione t di x in base b ?

$$(x / b^t) \text{ MOD } b$$

es base 16, $t=2$

$$x = 9999$$

$$(9999 / 16^2) \text{ MOD } 16 = 7$$

in binario:

$$x \quad \underbrace{0010} \quad \underbrace{0111} \quad \underbrace{0000} \quad \underbrace{1111} \quad \gg 8 = \underbrace{0010} \quad \underbrace{0111} \quad x / 16^2$$

shift
a dx di
8 posizioni

$$16^2 = (2^4)^2 = 2^8$$

$$\underbrace{0010} \quad \underbrace{0111} \quad \& \quad \text{AND}$$

$$0000 \quad 1111 = 2^4 - 1$$

$$0000 \quad 0111$$

$$(x / 16^2) \text{ MOD } 16$$

Analisi Ammortizzata

Quanto costa incrementare un contatore a n bit?

costo \equiv n° di bit da ispezionare

0 0 1 0 1 1 1

0 0 1 1 0 0 0

costo 4

1 1 1 1 1 1 1

0 0 0 0 0 0 0

costo n

ALGORITMO incrementaContatore (Array $v[0..n-1]$ di bit)

$i \leftarrow 0$

WHILE $i < n$ AND $v[i] = 1$ DO

$v[i] \leftarrow 0$
 $i \leftarrow i + 1$

IF $i < n$ THEN $v[i] \leftarrow 1$

Caso peggiore costo $O(n)$

k incrementi $O(kn)$

Esempio

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

Il bit in posizione

i cambia ogni 2^i incrementi

$$\begin{aligned}
 T(n, k) &= k + \left\lfloor \frac{k}{2} \right\rfloor + \left\lfloor \frac{k}{2^2} \right\rfloor + \dots + \left\lfloor \frac{k}{2^{n-1}} \right\rfloor \\
 &= \sum_{i=0}^{n-1} \left\lfloor \frac{k}{2^i} \right\rfloor \leq \sum_{i=0}^{n-1} \frac{k}{2^i} = k \sum_{i=0}^{n-1} \frac{1}{2^i} \leq k \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \\
 &= k \frac{1}{1 - \frac{1}{2}} = k \frac{1}{\frac{2-1}{2}} = 2k
 \end{aligned}$$

in medi-

$$T_{a,n}(k) = \frac{T(n, k)}{k} = \frac{2k}{k} = 2$$

↑
costo ammortizzato

ESPOENZIALE ITERATO

$$F(k) = \begin{cases} 1 & \text{se } k=0 \\ 2^{F(k-1)} & \text{se } k \geq 1 \end{cases}$$

$$F(0) = 1$$

$$F(1) = 2^{F(0)} = 2$$

$$F(2) = 2^{F(1)} = 2^2 = 4$$

$$F(3) = 2^{2^2}$$

$$F(4) = 2^{2^{2^2}}$$

⋮

$$F(n) = 2^{2^{2^{\dots^2}}} \left. \vphantom{2^{2^{2^{\dots^2}}}} \right\} n \text{ volte}$$

LOGARITMO ITERATO

$$e_g^{(1)} n = e_{g_2} n$$

$$e_g^{(2)} n = e_{g_2} (e_g^{(1)} n) = e_{g_2} e_{g_2} n$$

!

$$e_g^{(k)} n = e_{g_2} (e_g^{(k-1)} n) = \underbrace{e_{g_2} e_{g_2} \dots e_{g_2}}_{k \text{ volte}} n$$

$$e_g^* n = \min k \mid e_g^{(k)} n \leq 1$$

$$e_g^* x = 3 \quad 4 < x \leq 2^4 = 16$$

$$e_g^* x = 4 \quad 16 < x \leq 2^{16}$$

$$e_g^* x = 5 \quad 2^{16} < x < 2^{2^{15}}$$

;

$$e_g^* 1 = 0$$

$$e_g^* 2 = 1$$

$$e_g^* 3 = 2$$

$$e_g^* 4 = 2$$