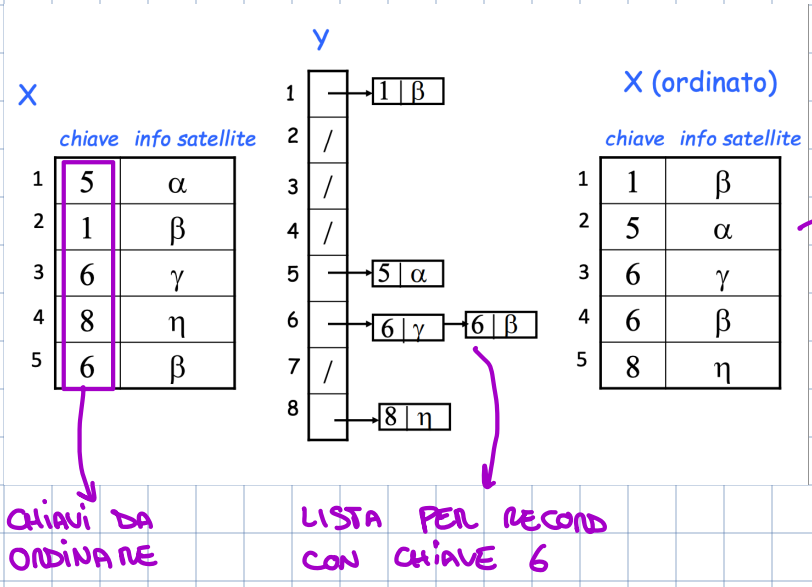
BUCKET SORT

SIANO m RECORD DA ORDINARE CON CHIAVI DA $[1, k]$, ESEMPIO DI RECORD:
NOME, COGNOME, ..., MATRICOLA
↳ CHIAVE DA ORDINARE

ABBIAMO QUINDI IN INPUT:
- m RECORD MANTENUTI IN UN ARRAY
- OGNI ELEMENTO DELL'ARRAY HA:
  - CAMPO CHIAVE
  - CAMPI ASSOCIATI

UTILIZZIAMO LA STESSA TECNICA DI INTEGER SORT PERÓ AL POSTO DEI CONTATORI ABBIAMO DELLE LISTE.
LA LISTA $Y[i]$ CONTERRÀ GLI ELEMENTI CON CHIAVE i, FATTO CIÓ CONCATENIAMO LE LISTE PER ORDINARE.



↳ ARRAY ORDINATO OTTENUTO CONCATENANDO LE LISTE

TEMPO $O(m+k)$ COME INTEGER SORT

```
BucketSort (X, k)
1.  Sia Y un array di dimensione k
2.  for i=1 to k do Y[i]=lista vuota
3.  for i=1 to n do
4.      appendi il record X[i] alla lista Y[chiave(X[i])]
5.  for i=1 to k do
6.      copia ordinatamente in X gli elemeti della lista Y[i]
```

SE MANTENIAMO LO STESSO ORDINE DELLE CHIAVI, BUCKET SORT É STABILE

STABILITÁ

UN ALGORITMO SI DICE STABILE SE PRESERVA L'ORDINE INIZIALE TRA ELEMENTI CON LA STESSA CHIAVE.

RADIX SORT

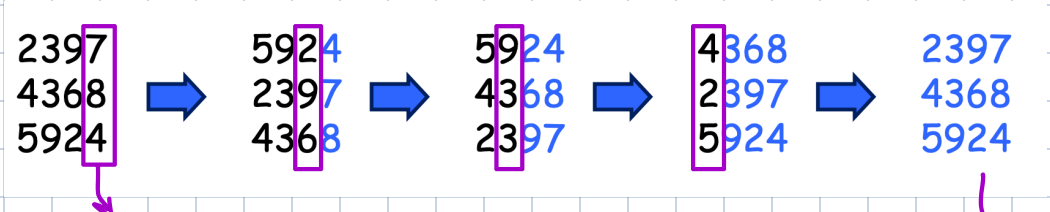
ORDINA m INTERI CON VALORI $[1, K]$, GLI ELEMENTI VENGONO RAPPRESENTATI IN UNA BASE b.

PARTIAMO DALLA CIFRA MENO SIGNIFICATIVA (QUELLA PIÚ A DX) E ARRIVIAMO A QUELLA PIÚ SIGNIFICATIVA (QUELLA PIÚ A SX).
AD OGNI PASSATA ORDINIAMO PER L'i-ESIMA CIFRA UTILIZZANDO IL BUCKET SORT.

↳ L'i-ESIMA CIFRA È LA CHIAVE E LE RESTANTI CIFRE SONO LE INFO SATELLITE.
↳ L'i-ESIMA CIFRA È UN INTERO TRA $[0, b-1]$

↳ IN BASE 10 OGNI CIFRA È UN INTERO TRA $[0, b-1]$

$b = 10$ (BASE 10)



ORDINIAMO PER QUESTE CIFRE

$\log_{10} 5924$

↳ NUMERI ORDINATI

QUESTO ALGORITMO È STABILE, SE 2 NUMERI HANNO L'i-ESIMA CIFRA UGUALE IL BUCKET SORT LI MANTIENE IN ORDINE.

UTILIZZANDO LA FORMULA CHIUSA $O(\log_b k)$ (b: BASE (es. 10); k: NUMERO MASSIMO (5924)) OTTENGO IL NUMERO DI CIFRE CHE MI SERVONO PER RAPPRESENTARE IL NUMERO k IN BASE b.

SE CI PENSIAMO IL RADIX SORT EFFETTUA TANTI BUCKET SORT QUANTE SONO LE CIFRE DEL NUMERO, QUINDI CON QUELLA FORMULA AVREMO IL NUMERO DI BUCKET SORT.
CIASCUNA PASSATA DI BUCKET SORT RICHIEDE: $O(m + b)$

↳ XCHÉ b E NON K COME NEL BUCKET?

↓

PERCHÉ EFFETTUANDO IL BUCKET SORT SU OGNI i-ESIMA CIFRA, ESSENDO IL NUMERO COMPLETO IN BASE b, OGNI i-ESIMA CIFRA PUÓ AVERE VALORE DA $[0, b-1]$, QUINDI LA LISTA Y DEL BUCKET È LUNGA $b-1$.

QUINDI $O((m+b)\cdot \log_b k)$ (SAREBBE k)

COSTO DEL BUCKET

MOLTIPLICATO IL NUMERO DI BUCKET (NUMERO DI CIFRE)

SE AVESSIMO $b = \Theta(m)$, QUINDI IL NUMERO DI NUMERI DA ORDINARE PARI AL VALORE MASSIMO, AVREMMO:

$$O((m+m)\cdot \log_m k) = O(m \cdot \log_m k) = O\left(m \cdot \frac{\log k}{\log m}\right)$$

→ BASE 10 SE NON SPECIFICATO

PROPRIETÀ DEI LOGARITMI PER CAMBIARE LA BASE

$$\log_a b = \frac{\log_c(a)}{\log_c(b)}$$

CON c NUOVA BASE

ESERCIZIO

Dato un vettore X di *n* interi in [1,*k*], costruire in tempo O(n+k) una struttura dati (oracolo) che sappia rispondere a domande (query) in tempo O(1) del tipo: "quanti interi in X cadono nell'intervallo [*a*,*b*]?", per ogni *a* e *b*.

COSTRUIRE IN TEMPO $O(m+k)$ UN ARRAY Y DI DIMENSIONE K, DOVE Y[i] È IL NUMERO DI ELEMENTI DI X CHE SONO $\leq i$.

CostruisciOracolo (X, k)

1. Sia Y un array di dimensione k
2. **for** i=1 **to** k **do** Y[i]=0
3. **for** i=1 **to** n **do** incrementa Y[X[i]]
4. **for** i=2 **to** k **do** Y[i]=Y[i]+Y[i-1]
5. **return** Y

InterrogaOracolo (Y, k, a, b)

1. **if** b > k **then** b=k
2. **if** a ≤ 1 **then return** Y[b]
   else return (Y[b]-Y[a-1])

8	2	1	4	2	1	1	2	3
1	2	3	4	5	6	7	8	9

3)

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

4)

3	6	7	8	8	8	8	9
1	2	3	4	5	6	7	8

1,1,1 — 2,5 — # ≤ 5 → 2, 1, 4, 2, 1, 1, 2, 3

IN OGNI POSIZIONE i ABBIAMO IL NUMERO DI NUMERI CHE SONO $\leq i$

ARRAY INIZIALE

8	2	1	4	2	1	1	2	3
1	2	3	4	5	6	7	8	9

4)

3	6	7	8	8	8	8	9
1	2	3	4	5	6	7	8

i

NELL' ARRAY INIZIALE CI SONO 7 ELEMENTI CHE SONO $\leq i$

ESEMPIO

[2,5]
a b

{ ~~1~~, ~~1~~, ~~1~~ }

{ 2, ~~1~~ 4, 2, ~~1~~, ~~1~~, 2, 3 } = 5

3	6	7	8	8	8	8	9
1	2	3	4	5	6	7	8

{ 2, 1, 2, 1, 1, 2 }

NON VA BENE

PRENDO Y[a-1]

BUCKET SORT

- STESSO PRINCIPIO DI INTEGERSORT MA NON UTILIZZA CONTATORI.
  VIENE DATA UNA LISTA X IN INPUT, OGNI ELEMENTO DELLA LISTA HA UN CAMPO CHIAVE E DELLE INFORMAZIONI SATELLITE, LE CHIAVI VANNO DA 1 A K.
  VIENE CREATO UN ARRAY Y LUNGO K DOVE OGNI CELLA È UNA LISTA VUOTA.
  SCORRE L'ARRAY X E PER OGNI ELEMENTO GUARDA LA CHIAVE E APPENDE L'ELEMENTO ALLA LISTA NELLA POSIZIONE PARI ALLA CHIAVE NELL'ARRAY Y. $O(m+k)$

SUPPONIAMO DI AVERE I SEGUENTI NUMERI
329, 457, 657, 839, 436, 720, 355.

CASO b = 10 → VIENE UTILIZZATA LA CIFRA PIÚ A DESTRA [0-9] E QUINDI IL BUCKET SORT HA 10 LISTE NEL SUO SECONDO ARRAY

```
Bucket 0: [720]
Bucket 5: [355]
Bucket 6: [436]
Bucket 7: [457, 657]
Bucket 9: [329, 839]
```

Ora riassembliamo i numeri in ordine:

`[720, 355, 436, 457, 657, 329, 839]`

```
Bucket 2: [720]
Bucket 3: [329]
Bucket 3: [355]
Bucket 3: [436]
Bucket 5: [457]
Bucket 5: [657]
Bucket 3: [839]
```

Riassembliamo:

`[720, 329, 355, 436, 839, 457, 657]`

```
Bucket 3: [329, 355]
Bucket 4: [436]
Bucket 4: [457]
Bucket 6: [657]
Bucket 7: [720]
Bucket 8: [839]
```

Riassembliamo:

`[329, 355, 436, 457, 657, 720, 839]` ✅ **Ordinato!**

- PASSAGGI RICHIESTI 3, BUCKET AVEVA MASSIMO 9 LISTE

CASO b = 100 → VENGONO UTILIZZATE LE 2 CIFRE PIÚ A DESTRA. IL BUCKET HA FINO A 100 LISTE

```
Bucket 29: [329]
Bucket 55: [355]
Bucket 36: [436]
Bucket 57: [457]
Bucket 57: [657]
Bucket 39: [839]
Bucket 20: [720]
```

Riassembliamo:

`[720, 329, 355, 436, 457, 657, 839]`

```
Bucket 3: [329, 355]
Bucket 4: [436, 457]
Bucket 6: [657]
Bucket 7: [720]
Bucket 8: [839]
```

Riassembliamo:

`[329, 355, 436, 457, 657, 720, 839]` ✅ **Ordinato!**

- PASSAGGI RICHIESTI 2, BUCKET AVEVA MASSIMO 100 LISTE

CASO b = 1000 → VENGONO UTILIZZATE LE 3 CIFRE PIÚ A DESTRA. IL BUCKET HA FINO A 1000 LISTE

```
Bucket 329: [329]
Bucket 355: [355]
Bucket 436: [436]
Bucket 457: [457]
Bucket 657: [657]
Bucket 720: [720]
Bucket 839: [839]
```

Riassembliamo direttamente:

`[329, 355, 436, 457, 657, 720, 839]` ✅ **Ordinato in 1 passaggio!**

- PASSAGGI RICHIESTI 1, BUCKET AVEVA MASSIMO 1000 LISTE

Base b	Numero di passaggi d	Costo per passaggio	Memoria usata
10	3	Basso (solo 10 bucket)	Bassa
100	2	Medio (100 bucket)	Moderata
1000	1	Alto (fino a 1000 bucket)	Elevata 🚀