ORDINAMENTO IN TEMPO LINEARE

PRESENTEREMO TRE ALGORITMI PER L'ORDINAMENTO

- COUNTING SORT
- RADIX SORT
- BUCKET SORT

CHE UTILIZZANO OPERAZIONI DIVERSE DAI CONFRONTI PER EFFETTUARE L'ORDINAMENTO E PER I QUALI IL LIMITE INFERIORE $\Omega(n \lg n)$ NON VALE

COUNTING SORT

- DATO UN ARRAY $A[1..n]$, PRESUPPONE CHE CIASCUNO DEGLI ELEMENTI $A[i]$ DA ORDINARE SIA UN NUMERO INTERO COMPRESO NELL'INTERVALLO $[0..k]$, PER QUALCHE INTERO k

- COMPLESSITA': $O(k+n)$
  (DUNQUE SE $k = O(n)$ LA COMPLESSITA' SARÀ $O(n)$)

IDEA: PER OGNI $0 \leq i \leq k$, CONTARE IL NUMERO DI ELEMENTI IN A MINORI O UGUALI AD i, DETERMINANDO COSÌ IL RANGO DI CIASCUN ELEMENTO DI A

ESEMPIO

	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3
	A	B	C	D	E	F	G	H

range: [0 .. 5]

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

- PER $0 \le i \le 5$, CI SONO C[i] ELEMENTI IN A MINORI O UGUALI AD i

```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k
 3      C[i] = 0
 4  for j = 1 to A.length
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k
 8      C[i] = C[i] + C[i - 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] - 1
```

ANALISI DI COMPLESSITA'

COUNTING-SORT(A, B, k)

```
1   let C[0..k] be a new array
2   for i = 0 to k
3       C[i] = 0
4   for j = 1 to A.length
5       C[A[j]] = C[A[j]] + 1
6   // C[i] now contains the number of elements equal to i.
7   for i = 1 to k
8       C[i] = C[i] + C[i − 1]
9   // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

$\Theta(k)$

$\Theta(n)$

$\Theta(k)$

$\Theta(n)$

TOTALE $\Theta(n+k)$

- SE $k = O(n)$, ALLORA $\Theta(n+k) = \Theta(n)$

OSSERVAZIONI

- COUNTING SORT UTILIZZA MEMORIA ESTERNA DI DIMENSIONE $\Theta(n+k)$

- E' UN ALGORITMO STABILE, CIOE' LE CHIAVI CON IDENTICO VALORE SI PRESENTANO NELL'ARRAY DI OUTPUT NELLO STESSO ORDINE IN CUI SI TROVANO NELL'ARRAY DI INPUT (IMPORTANTE IN PRESENZA DI DATI SATELLITE)

ESERCIZI

8.2-1

Illustrate the operation of COUNTING-SORT on the array $A = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2 \rangle$.

8.2-3

Suppose that we were to rewrite the **for** loop header in line 10 of the COUNTING-SORT as

10 **for** $j = 1$ **to** $A.length$

Show that the algorithm still works properly. Is the modified algorithm stable?

8.2-4

Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a \,..\, b]$ in $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

RADIX SORT

- RADIX SORT ORDINA RISPETTO A SEQUENZE DI CHIAVI

ESEMPI:

- SCHEDE PERFORATE 80 COLONNE, 12 POSIZIONI
- NUMERI IN NOTAZIONE DECIMALE (O IN ALTRA BASE)
- ♣ < ♦ < ♥ < ♠

  $2 < 3 < \ldots < 9 < 10 < J < Q < K < A$

- ORDINAMENTO LESSICOGRAFICO

  $(x_1, x_2, \ldots, x_r) < (y_1, y_2, \ldots, y_r)$

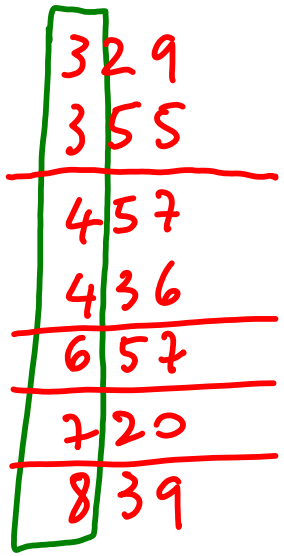
  SSE $x_i = y_i$, $1 \le i \le s$ E $x_{s+1} < y_{s+1}$,

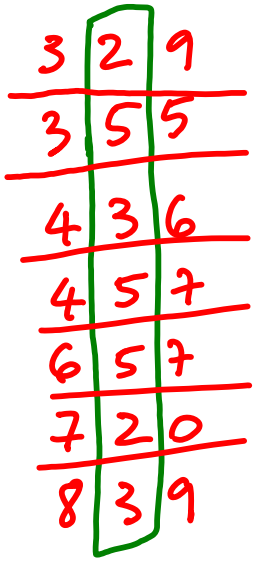
  PER QUALCHE $s \in [0 \ldots r-1]$

DUE APPROCCI ALL'ORDINAMENTO RISPETTO A PIÙ CHIAVI

- ORDINAMENTO DALLE CHIAVI PIÙ SIGNIFICATIVE A QUELLE MENO SIGNIFICATIVE (RICORSIVO)

329
457
657
839
436
720
355

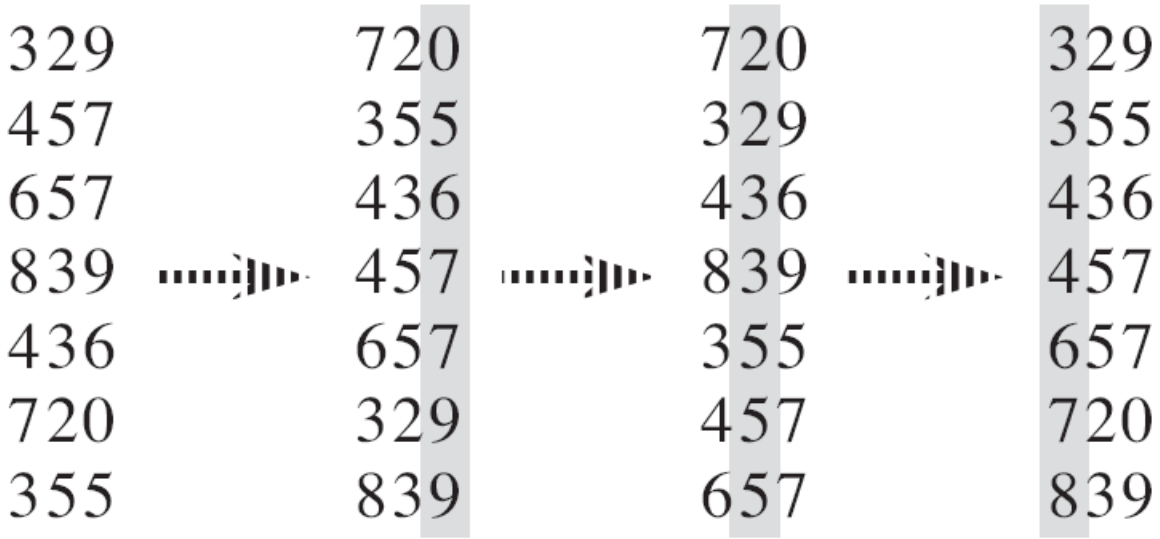




PROBLEMA: ALTO NUMERO DI PILE INTERMEDIE DA GESTIRE

- ORDINAMENTO DALLE CHIAVI MENO SIGNIFICATIVE A QUELLE PIÙ SIGNIFICATIVE (CON ALGORITMO DI ORDINAMENTO STABILE) ⇒ RADIX-SORT

perché funzina?



RADIX-SORT(A, d)

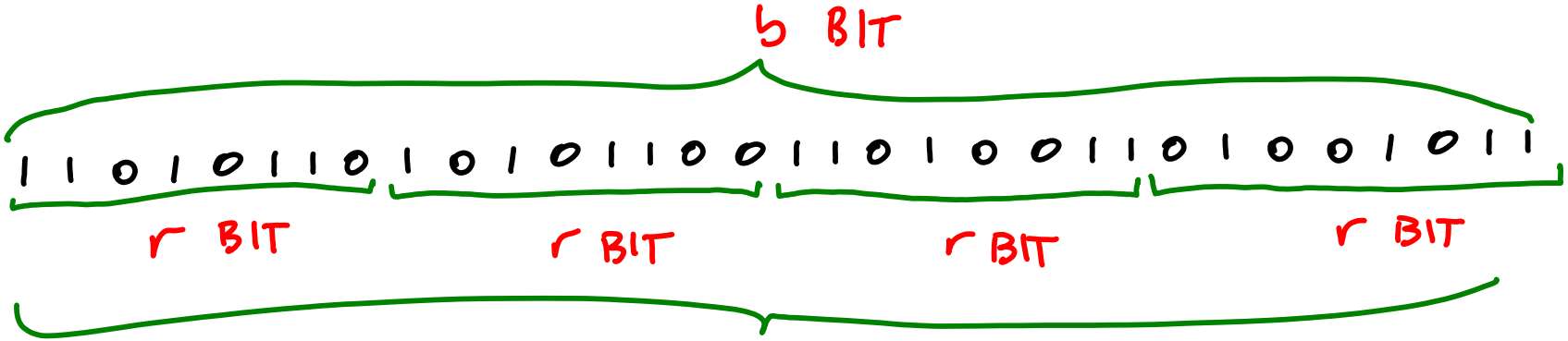
1 **for** $i = 1$ **to** d
2 use a stable sort to sort array A on digit i

- DATI n NUMERI DI d CIFRE, DOVE OGNI CIFRA PUÒ ASSUMERE SINO A k VALORI, LA PROCEDURA RADIX-SORT ORDINA CORRETTAMENTE I NUMERI IN TEMPO $\Theta(d(n+k))$ SE L'ORDINAMENTO STABILE UTILIZZATO DA RADIX-SORT HA COMPLESSITÀ $\Theta(n+k)$

- SE $d = \Theta(1)$ E $k = O(n)$, LA COMPLESSITÀ DI RADIX-SORT È $\Theta(n)$

LEMMA DATI n NUMERI DI b BIT ED $r \leq b$, RADIX-SORT ORDINA CORRETTAMENTE TALI NUMERI NEL TEMPO $\Theta\left(\frac{b}{r} \cdot (n+2^r)\right)$, PURCHE' UTILIZZI UN ALGORITMO STABILE DI ORDINAMENTO CON COMPLESSITA' $\Theta(n+k)$ PER INPUT NELL' INTERVALLO $[0..k]$ (ES. COUNTING SORT)

DIM.



$\left\lceil \frac{b}{r} \right\rceil$ CIFRE DA r BIT CIASCUNA NEL RANGE $[0..2^r-1]$

DUNQUE: $d = \left\lceil \frac{b}{r} \right\rceil$, $k = 2^r \Longrightarrow$

$$\Theta(d(n+k)) = \Theta\left(\frac{b}{r} \cdot (n+2^r)\right)$$

■

- DATI n E b, COME SCEGLIERE r PER MINIMIZZARE

$$\frac{b}{r} \cdot (n + 2^r) \quad ?$$

CASO $b < \lfloor \lg n \rfloor$

PER QUALUNQUE VALORE $r \leq b$ SI HA:

$$\frac{b}{r} \cdot n < \frac{b}{r} \cdot (n + 2^r) < \frac{b}{r} \cdot (n + 2^{\lfloor \lg n \rfloor}) \leq \frac{b}{r} \cdot (n + 2^{\lg n}) = 2\frac{b}{r} \cdot n$$

DA CUI $\frac{b}{r} \cdot (n + 2^r) = \Theta\left(\frac{bn}{r}\right)$

- IN PARTICOLARE, PER $r = \Theta(b)$ (ES. $r = b$) SI HA:

$$\Theta\left(\frac{b}{r} \cdot (n + 2^r)\right) = \Theta(n) .$$

CASO $b \geq \lfloor \lg m \rfloor$

LA SCELTA MIGLIORE (A MENO DI UN FATTORE COSTANTE) SI HA PER $r = \lfloor \lg n \rfloor$.

$$\Theta\left(\frac{b}{r}(n+2^r)\right) = \Theta\left(\frac{b}{\lfloor \lg m \rfloor}(n+2^{\lfloor \lg n \rfloor})\right) = \Theta\left(\frac{bn}{\lg n}\right)$$

- SE $r > \lfloor \lg n \rfloor$, $\frac{2^r}{r} \geq \frac{2^{\lfloor \lg n \rfloor}}{\lfloor \lg n \rfloor} = \Omega\left(\frac{n}{\lg n}\right)$

  E QUINDI $\frac{b}{r}(n+2^r) = \Omega\left(\frac{bn}{\lg n}\right)$

- SE $r < \lfloor \lg n \rfloor$, $\frac{b}{r} > \frac{b}{\lfloor \lg n \rfloor}$ E DUNQUE

$$\frac{b}{r}(n+2^r) = \Omega\left(\frac{bn}{\lg n}\right)$$

- SE $b = O(\lg n)$ E $r \approx \lg n$,

  IL TEMPO DI ESECUZIONE DI RADIX-SORT E' $\Theta(n)$.

- TUTTAVIA,

  - LA COSTANTE NASCOSTA IN $\Theta(n)$ E' PIÙ ALTA DI QUELLA NASCOSTA NELLA COMPLESSITA' $\Theta(n \lg n)$ DI QUICKSORT
  - RADIX-SORT (BASATO SU COUNTING SORT) UTILIZZA MEMORIA ESTERNA

ESERCIZIO

8.3-1

Illustrate the operation of RADIX-SORT on the following list of English words: COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX.

8.3-4

Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

BUCKET SORT

- BUCKET SORT ASSUME CHE I VALORI DA ORDINARE SIANO NUMERI REALI RANDOM NELL'INTERVALLO $[0, 1)$ CON DISTRIBUZIONE UNIFORME.

- IN TAL CASO BUCKET SORT HA UN TEMPO MEDIO DI ESECUZIONE $\Theta(n)$.

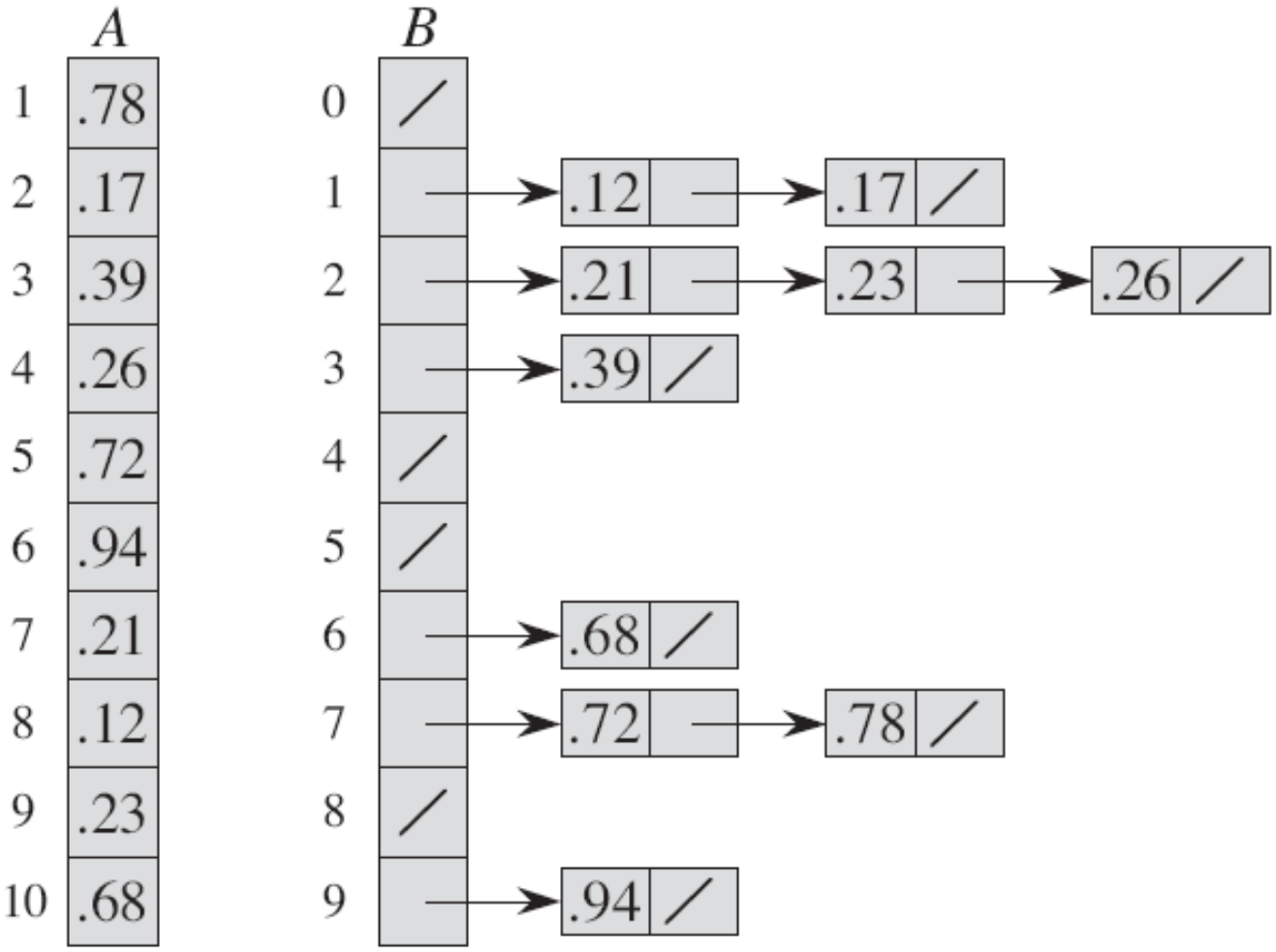
- PER ORDINARE UN ARRAY $A[1..n]$, BUCKET SORT DIVIDE L'INTERVALLO $[0,1)$ IN n SOTTOINTERVALLI UGUALI $[0, \frac{1}{n})$, $[\frac{1}{n}, \frac{2}{n})$, ..., $[\frac{n-1}{n}, 1)$ (BUCKET) E POI DISTRIBUISCE GLI n INPUT NEI BUCKET

- I BUCKET, MANTENUTI COME LISTE, SONO ORDINATI E CONCATENATI

- L'IPOTESI DI DISTRIBUZIONE UNIFORME PER GLI ELEMENTI DELL'ARRAY A IMPLICA CHE CIASCUN BUCKET CONTERRÀ, IN MEDIA, UN SOLO ELEMENTO

```
BUCKET-SORT(A)
1  n = A.length
2  let B[0..n − 1] be a new array
3  for i = 0 to n − 1
4       make B[i] an empty list
5  for i = 1 to n
6       insert A[i] into list B[⌊nA[i]⌋]
7  for i = 0 to n − 1
8       sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

ESEMPIO



OUTPUT

.12
.17
.21
.23
.26
.39
.68
.72
.78
.94

ANALISI NEL CASO PEGGIORE

(TUTTI GLI ELEMENTI CONFLUISCONO IN UN UNICO BUCKET)

BUCKET-SORT(A)

```
1  n = A.length
2  let B[0..n − 1] be a new array
3  for i = 0 to n − 1
4      make B[i] an empty list
5  for i = 1 to n
6      insert A[i] into list B[⌊nA[i]⌋]
7  for i = 0 to n − 1
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], ..., B[n − 1] together in order
```

$\Theta(n)$

$\Theta(n)$

$\Theta(n^2)$

$\Theta(n)$

$\Theta(n^2)$

ANALISI NEL CASO MIGLIORE

(CIASCUN BUCKET RICEVE ESATTAMENTE UN ELEMENTO)

BUCKET-SORT(A)

```
1  n = A.length                                                    Θ(n)
2  let B[0..n − 1] be a new array
3  for i = 0 to n − 1
4      make B[i] an empty list
5  for i = 1 to n
6      insert A[i] into list B[⌊nA[i]⌋]
7  for i = 0 to n − 1                                              O(n)
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], ..., B[n − 1] together in order   Θ(n)
```

$\Theta(n)$

ANALISI PROBABILISTICA NEL CASO MEDIO

```
BUCKET-SORT(A)
1  n = A.length
2  let B[0..n − 1] be a new array
3  for i = 0 to n − 1
4      make B[i] an empty list
5  for i = 1 to n
6      insert A[i] into list B[⌊nA[i]⌋]
7  for i = 0 to n − 1
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], ..., B[n − 1] together in order
```

O(n)

O(?) ← linea 8

O(n) ← linea 9

- OCCORRE CALCOLARE IL TEMPO TOTALE RICHIESTO DALLE n CHIAMATE DI INSERTION SORT (LINEA 8)

- n_i : VARIABILE CASUALE RAPPRESENTANTE IL NUMERO DI ELEMENTI INSERITI NEL BUCKET $B[i]$

- SI HA: $T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$

$$E[T(n)] = E\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[n_i^2])$$

$$= \Theta(n) + O\left(\sum_{i=0}^{n-1} E[n_i^2]\right)$$

- VERIFICHIAMO CHE $E[n_i^2] = 2 - \frac{1}{n}$

- PONIAMO $X_{ij} =_{df} \mathbb{I}\{ A[j] \text{ È INSERITO IN } B[i] \}$

$(1 \le j \le n, \quad 0 \le i \le n-1)$

DA CUI $n_i = \sum_{j=1}^{n} X_{ij}$

- $E[n_i^2] = E\left[\left(\sum_{j=1}^{n} X_{ij}\right)^2\right] = E\left[\sum_{j=1}^{n}\sum_{k=1}^{n} X_{ij} X_{ik}\right]$

$$= E\left[\sum_{j=1}^{n} X_{ij}^2 + \sum_{j=1}^{n}\sum_{\substack{k=1 \\ k\neq j}}^{n} X_{ij} X_{ik}\right]$$

$$= \sum_{j=1}^{n} E[X_{ij}^2] + \sum_{j=1}^{n}\sum_{\substack{k=1 \\ k\neq j}}^{n} E[X_{ij} X_{ik}]$$

- SI HA: $X_{ij}^2 = X_{ij}$

DUNQUE $E[X_{ij}^2] = Pr\{X_{ij}=1\} = \frac{1}{n}$

- PER $k \neq j$, X_{ij} E X_{ik} SONO INDIPENDENTI.

QUINDI

$$E[X_{ij} X_{ik}] = E[X_{ij}] \cdot E[X_{ik}] = \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

- PERTANTO

$$E[n_i^2] = \sum_{j=1}^{n} E[X_{ij}^2] + \sum_{j=1}^{n} \sum_{\substack{k=1 \\ k \neq j}}^{n} E[X_{ij} X_{ik}]$$

$$= \sum_{j=1}^{n} \frac{1}{n} + \sum_{j=1}^{n} \sum_{\substack{k=1 \\ k \neq j}}^{n} \frac{1}{n^2} \qquad \text{(SOSTITUENDO)}$$

$$= n \cdot \frac{1}{n} + n \cdot (n-1) \frac{1}{n^2} = 1 + \frac{n-1}{n} = 2 - \frac{1}{n}$$

- CONCLUDIAMO CHE

$$E[T(n)] = \Theta(n) + O\left(\sum_{i=0}^{n-1} E[n_i^2]\right)$$

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

ESERCIZIO

8.4-1

Illustrate the operation of BUCKET-SORT on the array $A = \langle .79, .13, .16, .64, .39, .20, .89, .53, .71, .42 \rangle$.