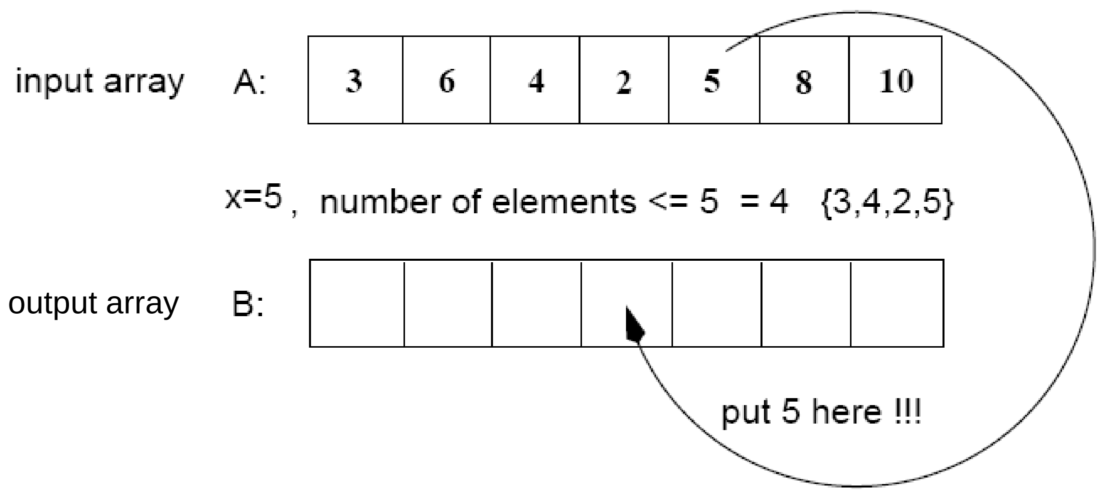
Counting Sort

Counting Sort

- Assumptions:
  - Sort n integers which are in the range $[0 \ldots r]$
  - r is in the order of n, that is, $r=O(n)$
- Idea:
  - For each element x, find the number of elements $\leq x$
  - Place x into its correct position in the output array



Step 1

Find the number of times $A[i]$ appears in A

input array A:

3	6	4	1	3	4	1	4

allocate C

1	2	3	4	5	6
0	0	0	0	0	0

Allocate $C[1..r]$ (histogram)

i=1, A[1]=3

1	2	3	4	5	6
0	0	1	0	0	0

C[A[1]]=C[3]=1

For $1 \leq i \leq n, ++C[A[i]];$

i=2, A[2]=6

1	2	3	4	5	6
0	0	1	0	0	1

C[A[2]]=C[6]=1

i=3, A[3]=4

1	2	3	4	5	6
0	0	1	1	0	1

C[A[3]]=C[4]=1

⋮

i=8, A[8]=4

1	2	3	4	5	6
2	0	2	3	0	1

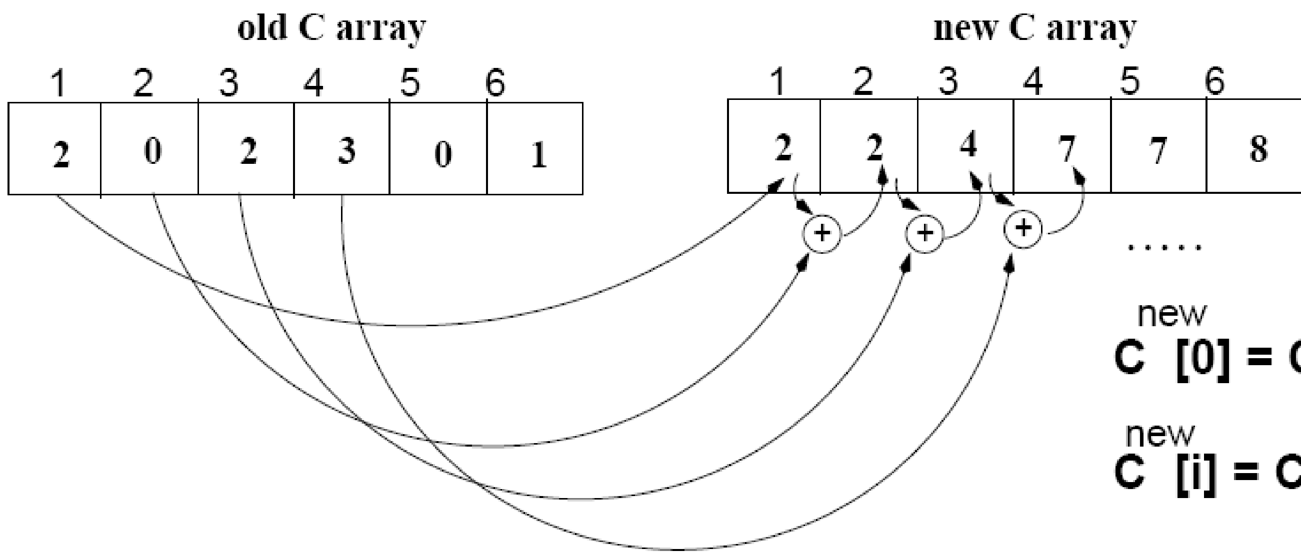
C[A[8]]=C[4]=3

C[i] = number of times element i appears in A (i.e., frequencies)

Step 2

Find the number of elements $\leq A[i]$,

(i.e., cumulative sums)



$$C^{new}[0] = C^{old}[0]$$

$$C^{new}[i] = C^{new}[i-1] + C^{old}[i]$$

C[i] = # elements <= i

Algorithm

- Start from the last element of A
- Place A[i] at its correct place in the output array
- Decrease C[A[i]] by one

Example

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

	0	1	2	3	4	5
C	2	0	2	3	0	1

(frequencies)

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

(cumulative sums)

	1	2	3	4	5	6	7	8
B							3	

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

	1	2	3	4	5	6	7	8
B		0					3	

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

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

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

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

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

Example (cont.)

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

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

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

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

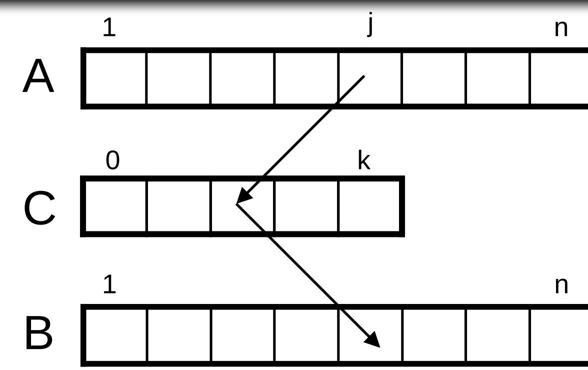
	0	1	2	3	4	5
C	0	2	3	4	7	7

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

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

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

COUNTING-SORT



Alg.: COUNTING-SORT(A, B, n, k)

for $i \leftarrow 0$ **to r**
    do $C[\,i\,] \leftarrow 0$
for $j \leftarrow 1$ **to** n
    do $C[A[\,j\,]] \leftarrow C[A[\,j\,]] + 1$
$\triangleright C[i]$ contains the number of elements equal to i
for $i \leftarrow 1$ **to r**
    do $C[\,i\,] \leftarrow C[\,i\,] + C[i\,-1]$
$\triangleright C[i]$ contains the number of elements $\leq i$
for $j \leftarrow n$ **downto** 1
    do $B[C[A[\,j\,]]] \leftarrow A[\,j\,]$
        $C[A[\,j\,]] \leftarrow C[A[\,j\,]] - 1$

Analysis of Counting Sort

Alg.: COUNTING-SORT(A, B, n, k)

```
for i ← 0 to r                                  Θ(r)
    do C[ i ] ← 0
for j ← 1 to n                                  Θ(n)
     do C[A[ j ]] ← C[A[ j ]] + 1
▷C[i] contains the number of elements equal to i
for i ← 1 to r                                  Θ(r)
    do C[ i ] ← C[ i ] + C[i -1]
▷C[i] contains the number of elements ≤ i
for j ← n downto 1                              Θ(n)
    do B[C[A[ j ]]] ← A[ j ]
         C[A[ j ]] ← C[A[ j ]] - 1
```

Overall time: $\Theta(n + r)$

Analysis of Counting Sort

- Overall time: $\Theta(n + r)$
- In practice we use COUNTING sort when $r = O(n)$

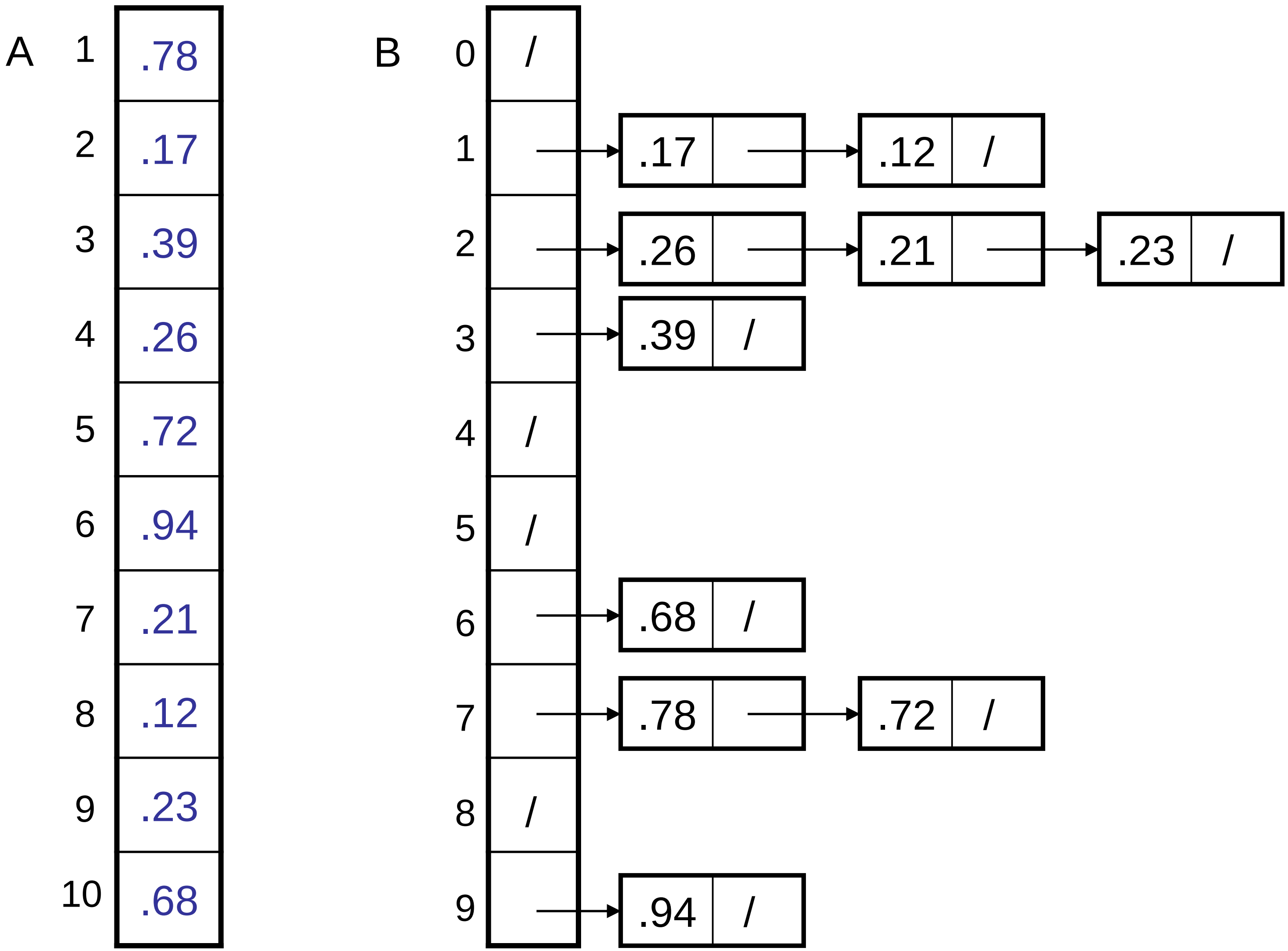
  $\Rightarrow$ running time is $\Theta(n)$
- Counting sort is **stable**
- Counting sort is not **in place** sort

Bucket Sort

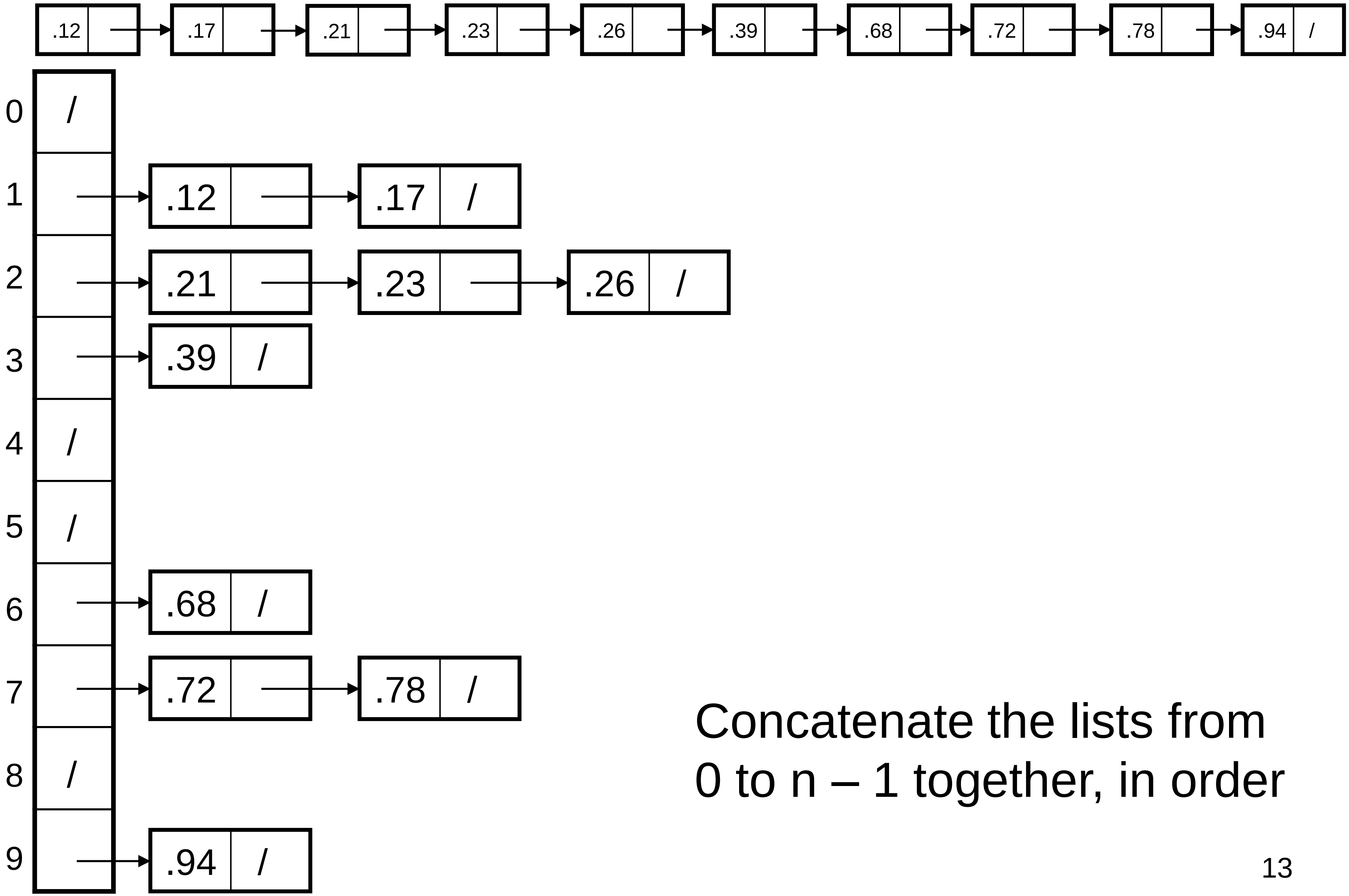
- Assumption:
  - the input is generated by a random process that distributes elements uniformly over [0, 1)
- Idea:
  - Divide [0, 1) into n equal-sized buckets
  - Distribute the n input values into the buckets
  - Sort each bucket (e.g., using quicksort)
  - Go through the buckets in order, listing elements in each one

- **Input:** $A[1 \, . \, . \, n]$, where $0 \le A[i] < 1$ for all i
- **Output:** elements $A[i]$ sorted
- **Auxiliary array:** $B[0 \, . \, . \, n - 1]$ of linked lists, each list initially empty

Example - Bucket Sort



Example - Bucket Sort



Concatenate the lists from 0 to n – 1 together, in order

Correctness of Bucket Sort

- Consider two elements $A[i]$, $A[j]$
- Assume without loss of generality that $A[i] \leq A[j]$
- Then $\lfloor nA[i] \rfloor \leq \lfloor nA[j] \rfloor$
  - $A[i]$ belongs to the same bucket as $A[j]$ or to a bucket with a lower index than that of $A[j]$
- If $A[i]$, $A[j]$ belong to the same bucket:
  - sorting puts them in the proper order
- If $A[i]$, $A[j]$ are put in different buckets:
  - concatenation of the lists puts them in the proper order

Analysis of Bucket Sort

Alg.: BUCKET-SORT(A, n)

for $i \leftarrow 1$ **to** n

do insert A[i] into list $B[\lfloor nA[i] \rfloor]$

O(n)

for $i \leftarrow 0$ **to** $n - 1$

do sort list $B[i]$ with quicksort sort

Θ(n)

concatenate lists $B[0], B[1], \dots, B[n-1]$

together in order

O(n)

return the concatenated lists

Θ(n)