

Winter 2016  
COMP-250: Introduction  
to Computer Science

Lecture 21, March 29, 2016

INTRODUCTION TO  
**ALGORITHMS**

SECOND EDITION

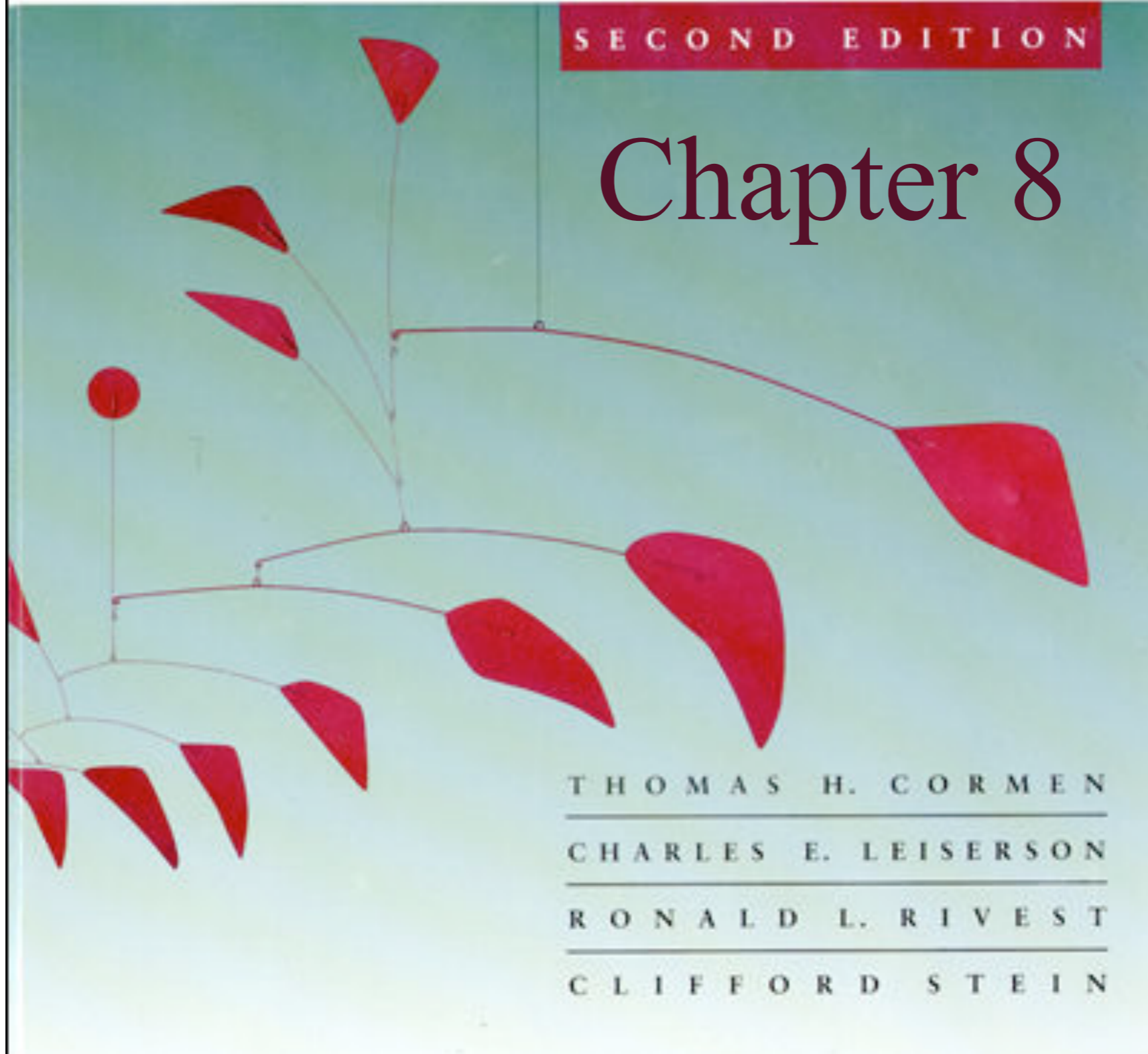
Chapter 8

THOMAS H. CORMEN

CHARLES E. LEISERSON

RONALD L. RIVEST

CLIFFORD STEIN



COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```



COUNTING-SORT( $A, B, k$ )

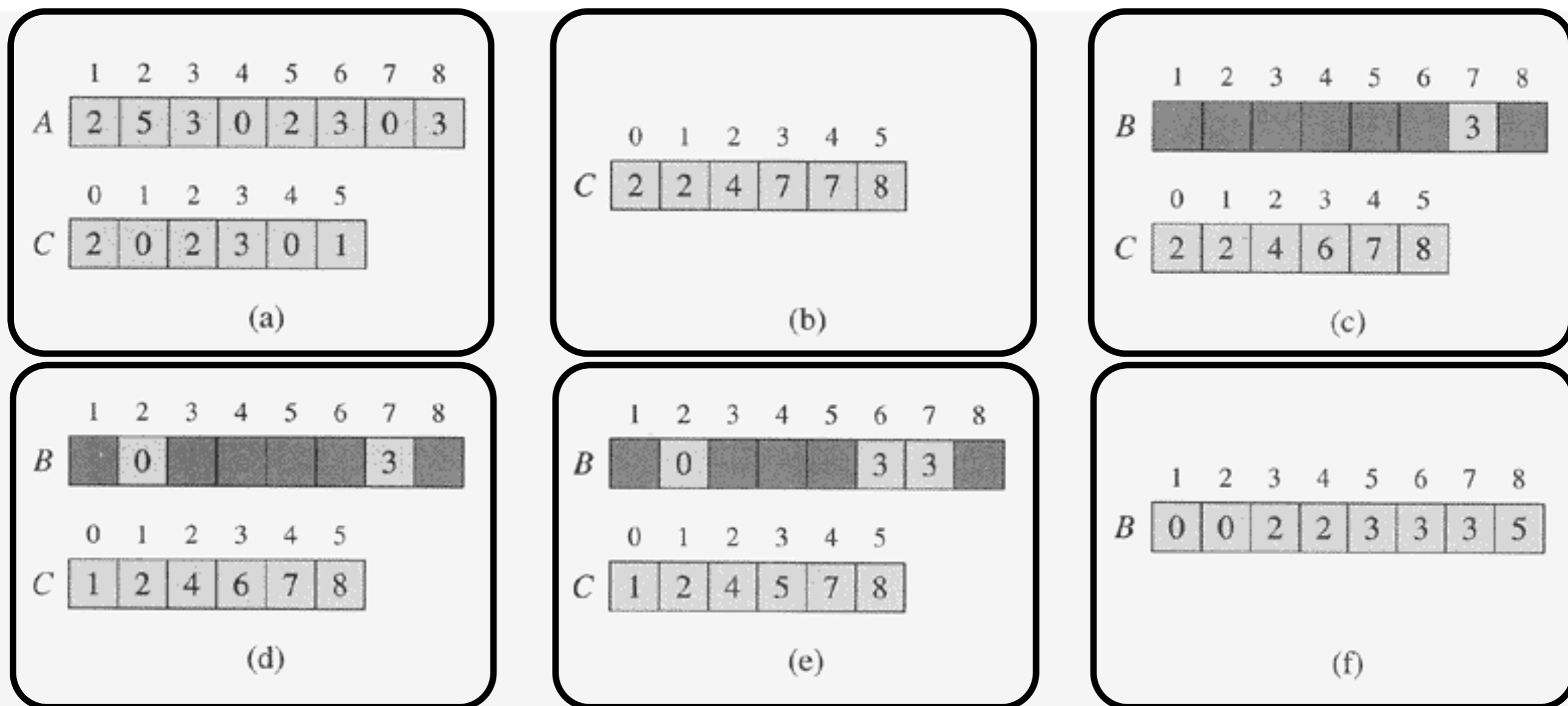
```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

COUNTING-SORT( $A, B, k$ )

```
1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $length[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5   $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8   $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow length[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

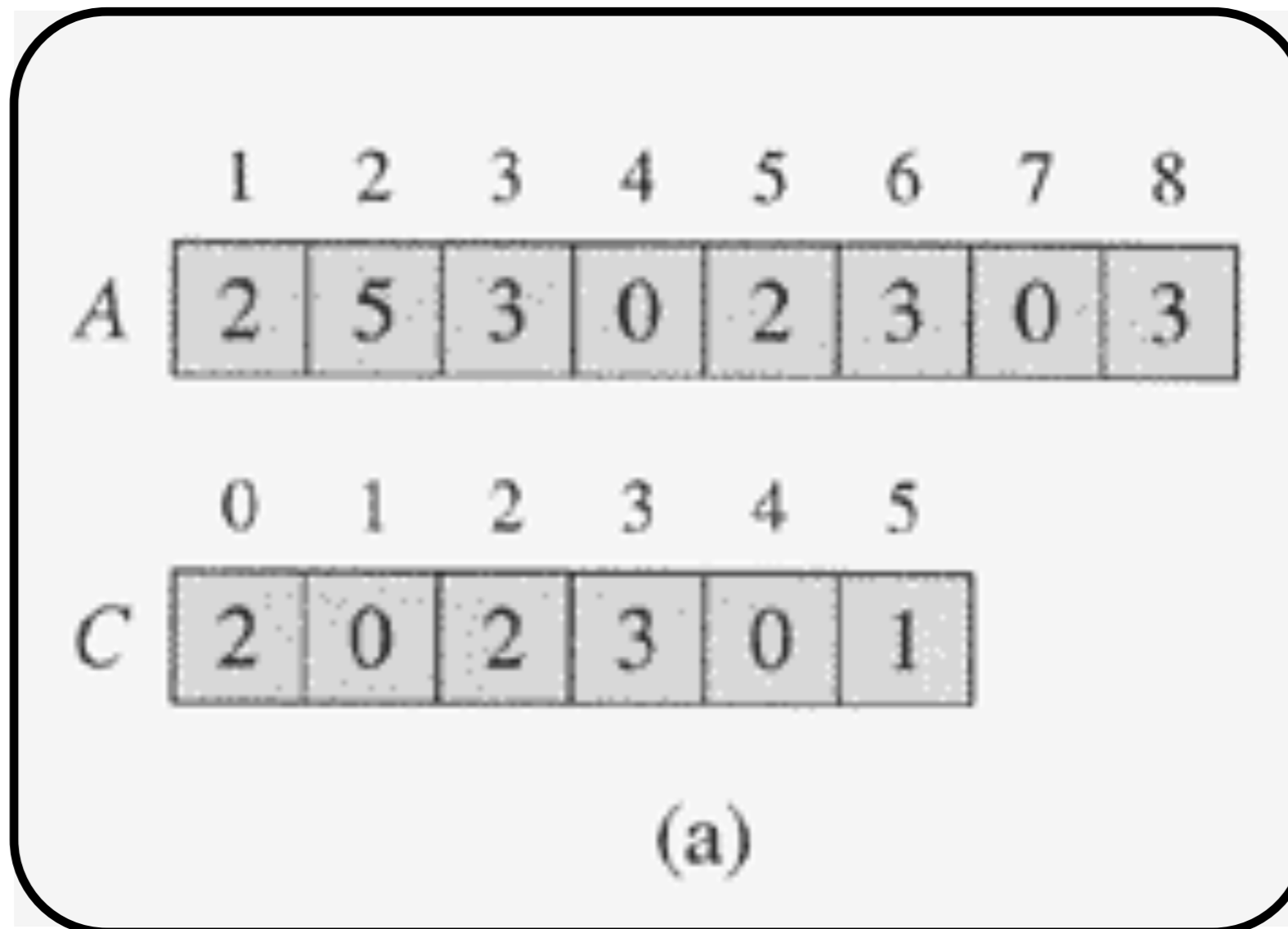


**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

```

1  for  $i \leftarrow 0$  to  $k$ 
2      do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $\text{length}[A]$ 
4      do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5  ▷  $C[i]$  now contains the number of elements equal to  $i$ 
6  for  $i \leftarrow 1$  to  $k$ 
7      do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8  ▷  $C[i]$  now contains the number of elements less than or equal to  $i$ .
9  for  $j \leftarrow \text{length}[A]$  downto 1
10     do  $B[C[A[j]]] \leftarrow A[j]$ 
11     do  $C[A[j]] \leftarrow C[A[j]] - 1$ 

```



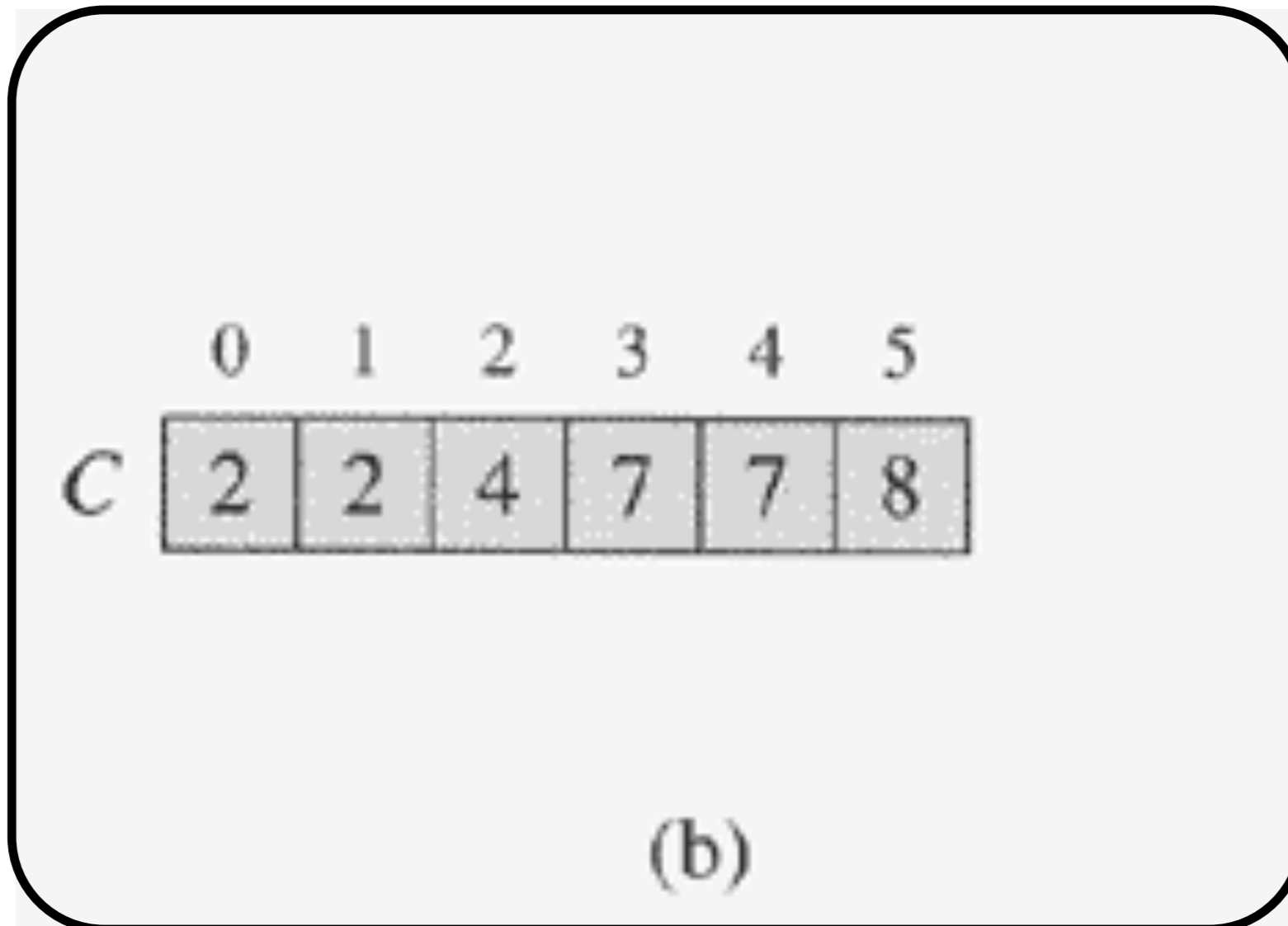
**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

COUNTING-SORT( $A, B, k$ )

```

1 for  $i \leftarrow 0$  to  $k$ 
2   do  $C[i] \leftarrow 0$ 
3 for  $j \leftarrow 1$  to  $\text{length}[A]$ 
4   do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5  $\triangleright C[i]$  now contains the number of elements equal to  $i$ .
6 for  $i \leftarrow 1$  to  $k$ 
7   do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8  $\triangleright C[i]$  now contains the number of elements less than or equal to  $i$ .
9 for  $j \leftarrow \text{length}[A]$  downto 1
10  do  $B[C[A[j]]] \leftarrow A[j]$ 
11     $C[A[j]] \leftarrow C[A[j]] - 1$ 

```

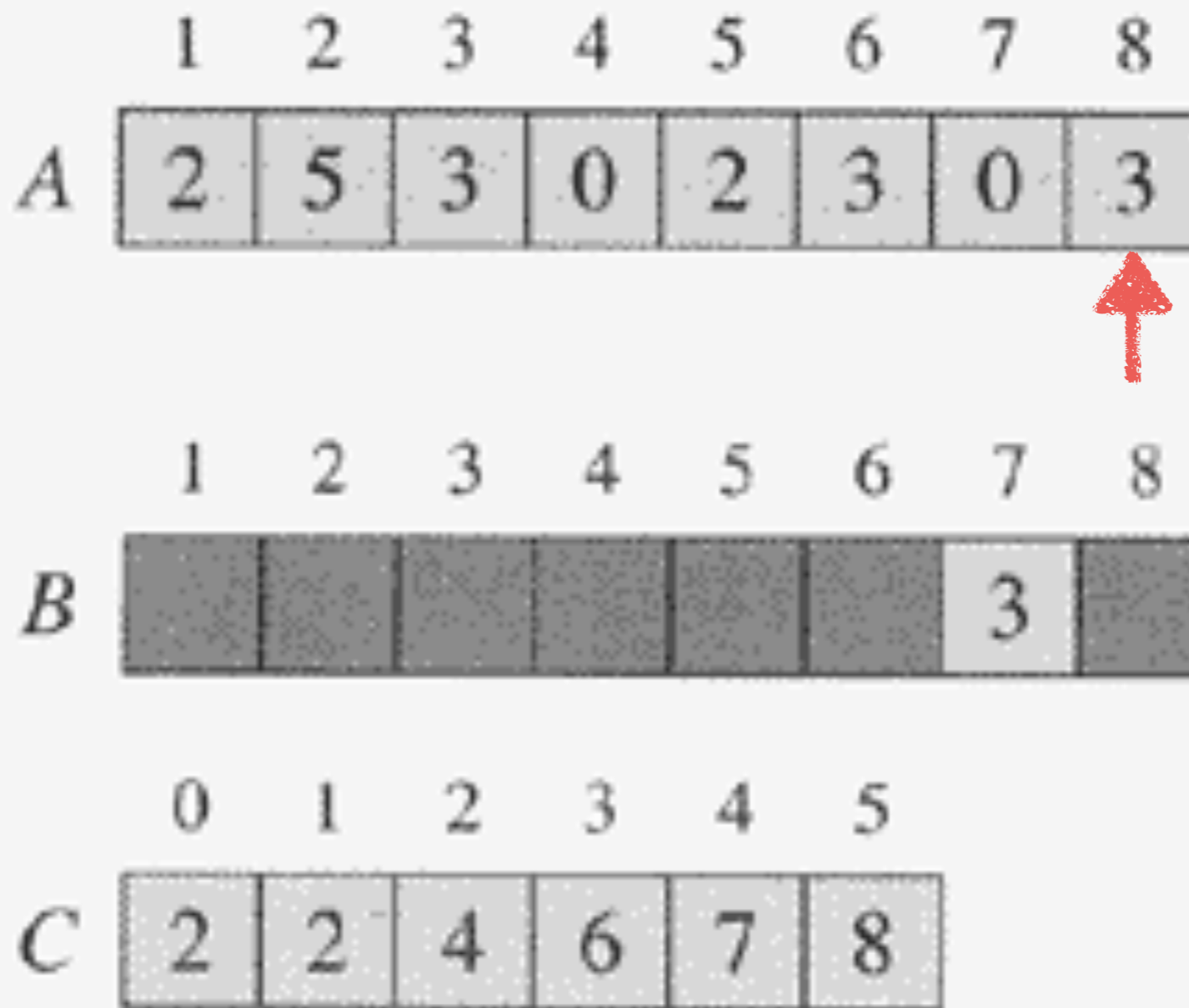


**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

```

1 for  $i \leftarrow 0$  to  $k$ 
2   do  $C[i] \leftarrow 0$ 
3 for  $j \leftarrow 1$  to  $\text{length}[A]$ 
4   do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5  $\triangleright C[i]$  now contains the number of elements
6 for  $i \leftarrow 1$  to  $k$ 
7   do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8  $\triangleright C[i]$  now contains the number of elements
9 for  $j \leftarrow \text{length}[A]$  downto 1
10  do  $B[C[A[j]]] \leftarrow A[j]$ 
11  do  $C[A[j]] \leftarrow C[A[j]] - 1$ 

```

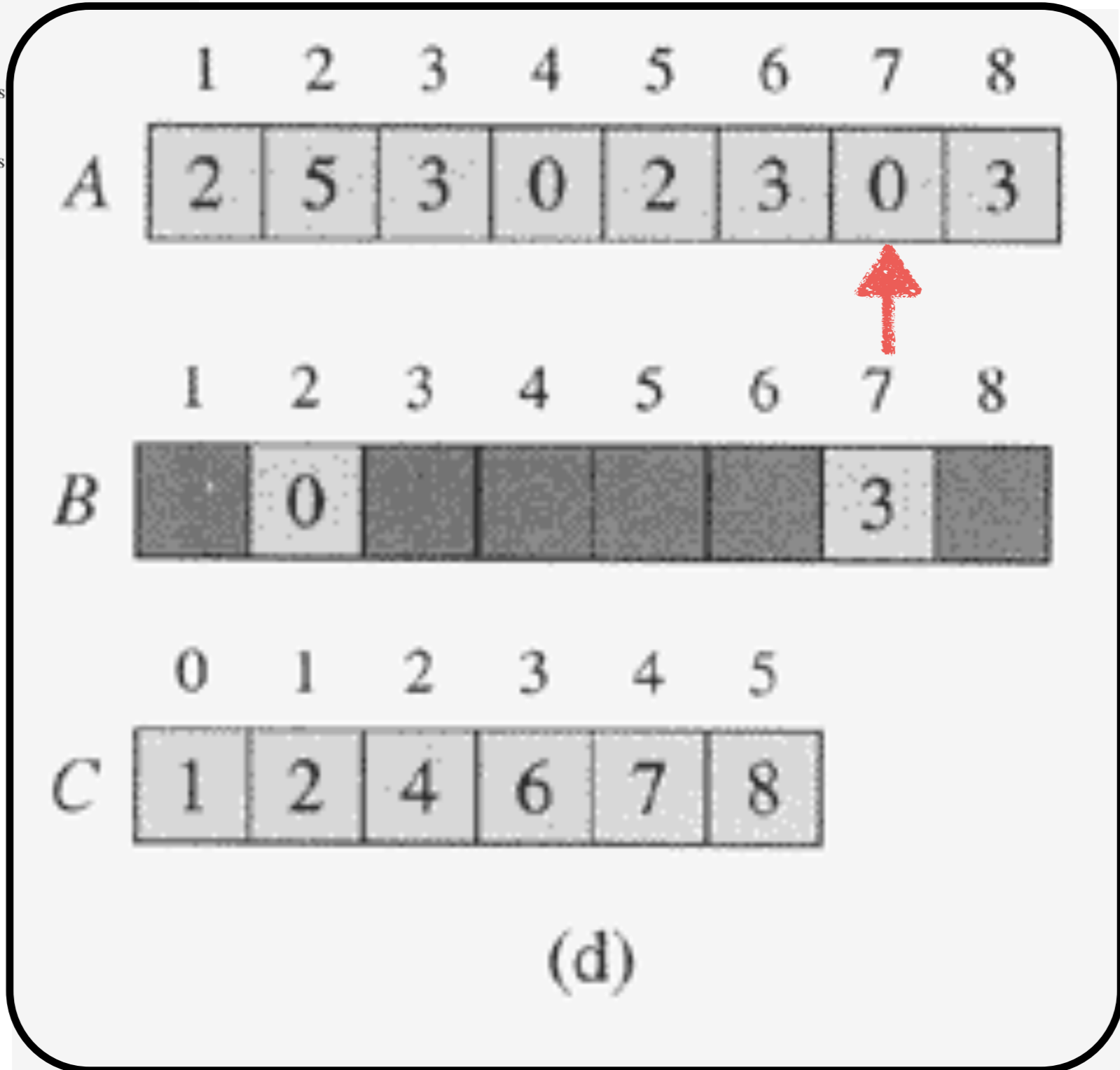


(c)

**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

```

COUNTING-SORT( $A, B, k$ )
1  for  $i \leftarrow 0$  to  $k$ 
2    do  $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1$  to  $\text{length}[A]$ 
4    do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5  ▷  $C[i]$  now contains the number of elements
6  for  $i \leftarrow 1$  to  $k$ 
7    do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8  ▷  $C[i]$  now contains the number of elements
9  for  $j \leftarrow \text{length}[A]$  downto 1
10   do  $B[C[A[j]]] \leftarrow A[j]$ 
11   do  $C[A[j]] \leftarrow C[A[j]] - 1$ 
    
```



**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

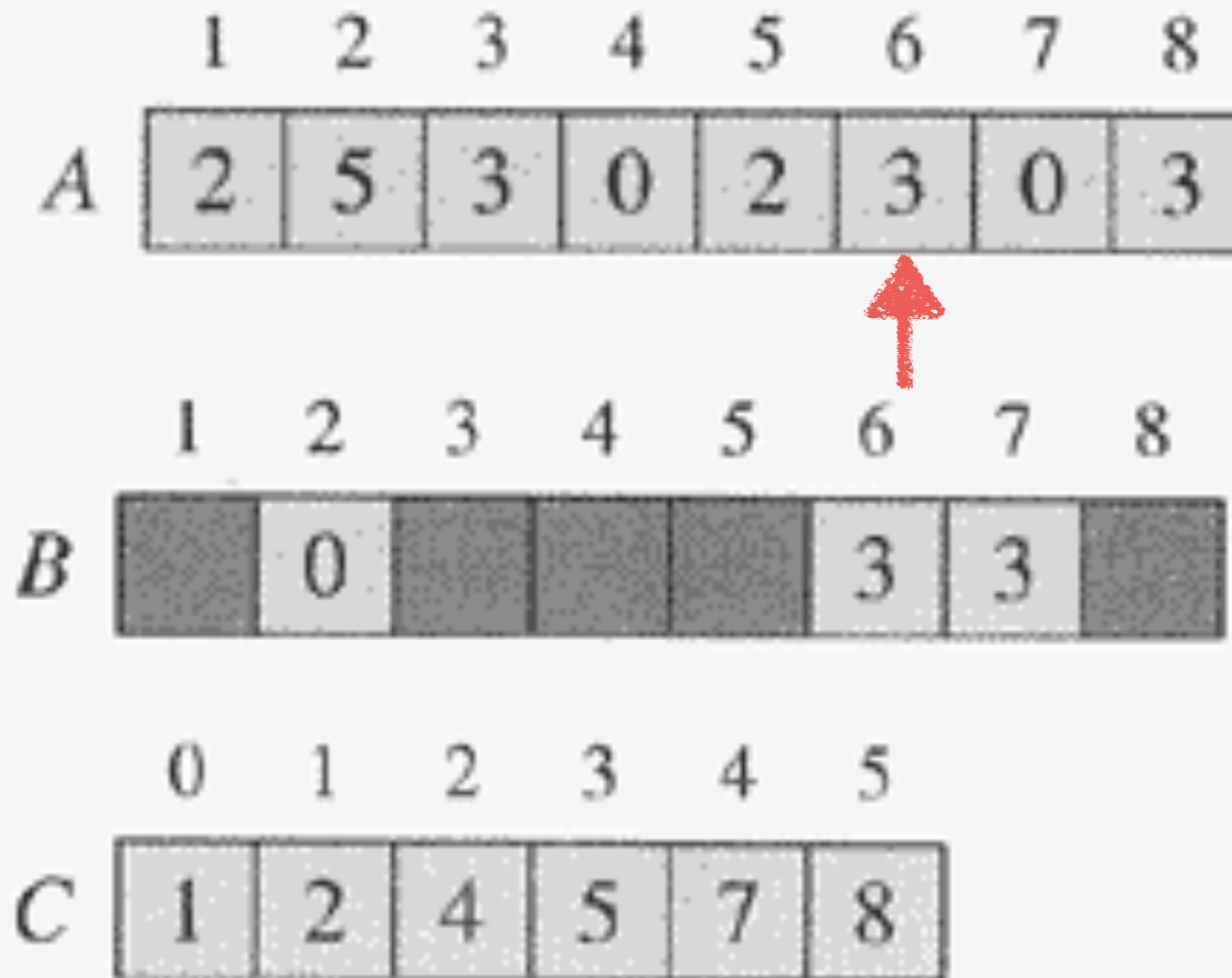


COUNTING-SORT( $A, B, k$ )

```

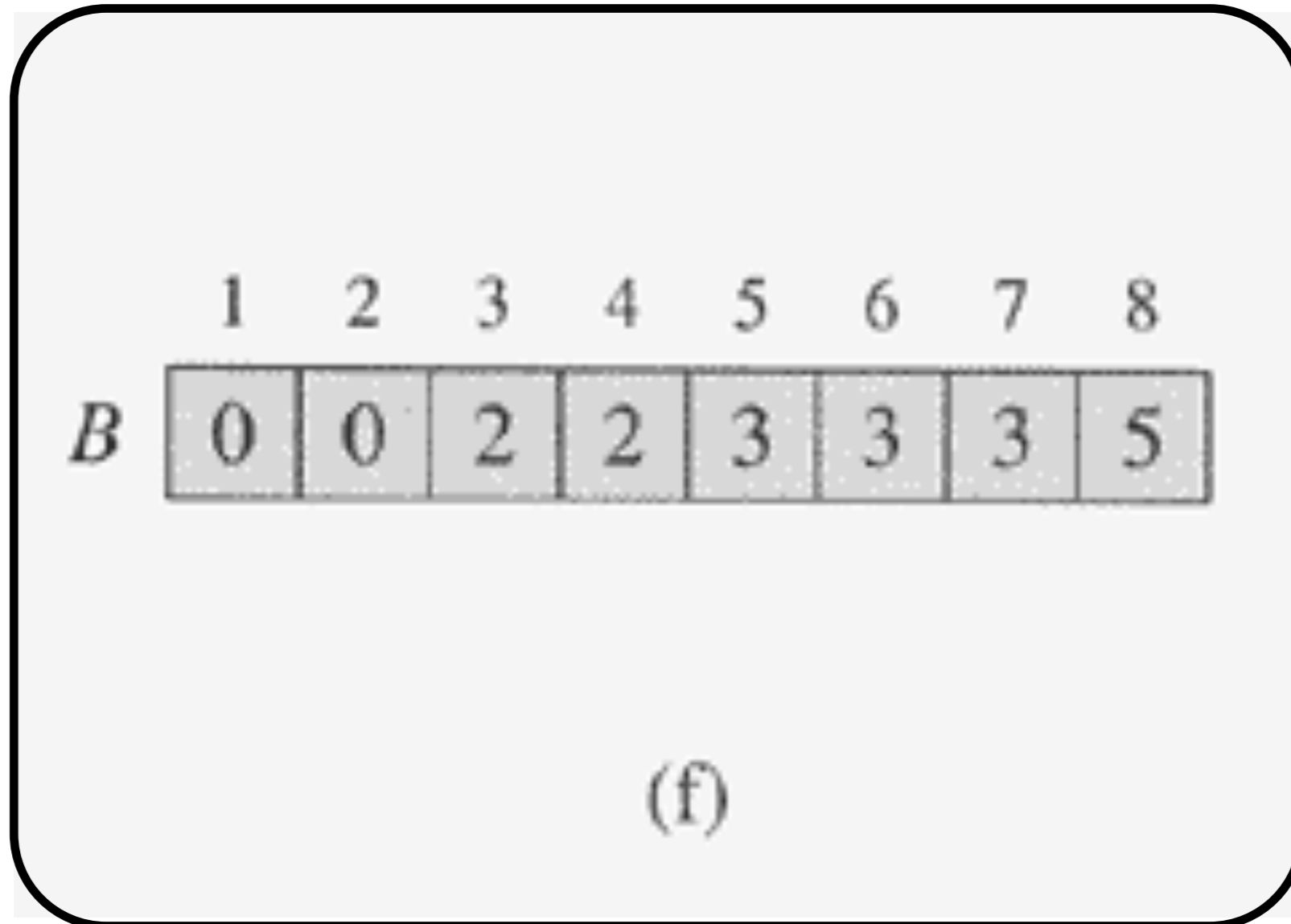
1 for  $i \leftarrow 0$  to  $k$ 
2   do  $C[i] \leftarrow 0$ 
3 for  $j \leftarrow 1$  to  $\text{length}[A]$ 
4   do  $C[A[j]] \leftarrow C[A[j]] + 1$ 
5  $\triangleright C[i]$  now contains the number of elements
6 for  $i \leftarrow 1$  to  $k$ 
7   do  $C[i] \leftarrow C[i] + C[i - 1]$ 
8  $\triangleright C[i]$  now contains the number of elements
9 for  $j \leftarrow \text{length}[A]$  downto 1
10  do  $B[C[A[j]]] \leftarrow A[j]$ 
11  do  $C[A[j]] \leftarrow C[A[j]] - 1$ 

```



(e)

**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .



**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 4. (b) The array  $C$  after line 7. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 9–11, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

## Radix sort

How IBM made its money. Punch card readers for census tabulation in early 1900's. Card sorters, worked on one column at a time. It's the algorithm for using the machine that extends the technique to multi-column sorting. The human operator was part of the algorithm!

*Key idea:* Sort *least* significant digits first.



**Figure 8.3** The operation of radix sort on a list of seven 3-digit numbers. The leftmost column is the input. The remaining columns show the list after successive sorts on increasingly significant digit positions. Shading indicates the digit position sorted on to produce each list from the previous one.

**RADIX-SORT**( $A, d$ )

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

2         **do** use a stable sort to sort array  $A$  on digit  $i$

### *Correctness:*

- Induction on number of passes ( $i$  in pseudocode).
- Assume digits  $1, 2, \dots, i - 1$  are sorted.
- Show that a stable sort on digit  $i$  leaves digits  $1, \dots, i$  sorted:
  - If 2 digits in position  $i$  are different, ordering by position  $i$  is correct, and positions  $1, \dots, i - 1$  are irrelevant.
  - If 2 digits in position  $i$  are equal, numbers are already in the right order (by inductive hypothesis). The stable sort on digit  $i$  leaves them in the right order.

This argument shows why it's so important to use a stable sort for intermediate sort.

**RADIX-SORT**( $A, d$ )

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

2         **do** use a stable sort to sort array  $A$  on digit  $i$

**Analysis:** Assume that we use counting sort as the intermediate sort.

- $\Theta(n + k)$  per pass (digits in range  $0, \dots, k$ )
- $d$  passes
- $\Theta(d(n + k))$  total
- If  $k = O(n)$ , time =  $\Theta(dn)$ .

## How to break each key into digits?

- $n$  words.
- $b$  bits/word.
- Break into  $r$ -bit digits. Have  $d = \lceil b/r \rceil$ .
- Use counting sort,  $k = 2^r - 1$ .

Example: 32-bit words, 8-bit digits.  $b = 32$ ,  $r = 8$ ,  $d = \lceil 32/8 \rceil = 4$ ,  $k = 2^8 - 1 = 255$ .

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

## How to break each key into digits?

- $n$  words.
- $b$  bits/word.
- Break into  $r$ -bit digits. Have  $d = \lceil b/r \rceil$ .
- Use counting sort,  $k = 2^r - 1$ .

Example: 32-bit words, 8-bit digits.  $b = 32$ ,  $r = 8$ ,  $d = \lceil 32/8 \rceil = 4$ ,  $k = 2^8 - 1 = 255$ .

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

How to choose  $r$ ? Balance  $b/r$  and  $n + 2^r$ . Choosing  $r \approx \lg n$  gives us  $\Theta\left(\frac{b}{\lg n} (n + n)\right) = \Theta(bn / \lg n)$ .

- If we choose  $r < \lg n$ , then  $b/r > b/\lg n$ , and  $n + 2^r$  term doesn't improve.
- If we choose  $r > \lg n$ , then  $n + 2^r$  term gets big. Example:  $r = 2 \lg n \Rightarrow 2^r = 2^{2 \lg n} = (2^{\lg n})^2 = n^2$ .

So, to sort  $2^{16}$  32-bit numbers, use  $r = \lg 2^{16} = 16$  bits.  $\lceil b/r \rceil = 2$  passes.



Compare radix sort to merge sort and quicksort:

- 1 million ( $2^{20}$ ) 32-bit integers.
- Radix sort:  $\lceil 32/20 \rceil = 2$  passes.
- Merge sort/quicksort:  $\lg n = 20$  passes.
- Remember, though, that each radix sort “pass” is really 2 passes—one to take census, and one to move data.

Compare radix sort to merge sort and quicksort:

- 1 million ( $2^{20}$ ) 32-bit integers.
- Radix sort:  $\lceil 32/20 \rceil = 2$  passes.
- Merge sort/quicksort:  $\lg n = 20$  passes.
- Remember, though, that each radix sort “pass” is really 2 passes—one to take census, and one to move data.

Uses 65536 memory cells, however...

Compare radix sort to merge sort and quicksort:

- 1 million ( $2^{20}$ ) 32-bit integers.
- Radix sort:  $\lceil 32/20 \rceil = 2$  passes.
- Merge sort/quicksort:  $\lg n = 20$  passes.
- Remember, though, that each radix sort “pass” is really 2 passes—one to take census, and one to move data.

Uses 65536 memory cells, however...

How does radix sort violate the ground rules for a comparison sort?

- Using counting sort allows us to gain information about keys by means other than directly comparing 2 keys.
- Used keys as array indices.

Winter 2016  
COMP-250: Introduction  
to Computer Science

Lecture 21, March 29, 2016