

# Counting Inversions- Divide and Conquer

Srinidhi H  
Dept of CSE, MSRIT

# Counting Inversions

- Music site tries to match your song preferences with others.
  - You rank  $n$  songs.
  - Music site consults database to find people with **similar** tastes.
- Similarity metric: number of inversions between two rankings.
  - My rank:  $1, 2, \dots, n$ .
  - Your rank:  $a_1, a_2, \dots, a_n$ .
  - Songs  $i$  and  $j$  **inverted** if  $i < j$ , but  $a_i > a_j$ .

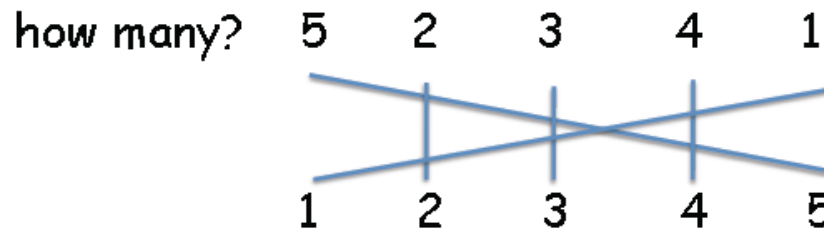
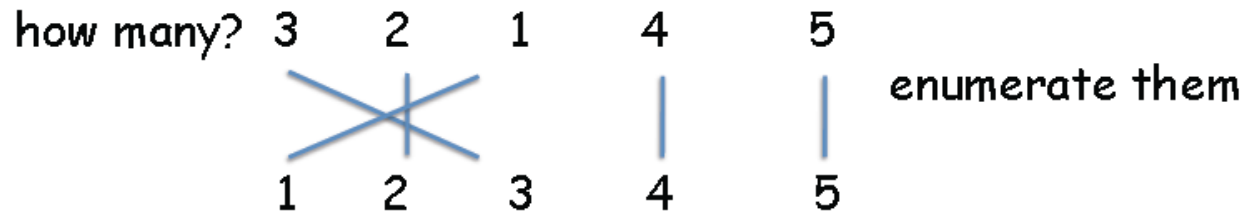
*Songs*

	A	B	C	D	E
Me	1	2	3	4	5
You	1	3	4	2	5

Inversions

3-2, 4-2

# Visualizing inversions



Naive algorithm

```
# test all pairs
c=0
for i in 0 to n-1
  for j in i+1 to n-1
    compare  $r_i$  and  $r_j$ 
    if inversion c++
```

# Counting Inversions: Divide-and-Conquer

- Divide-and-conquer.

1	5	4	8	10	2	6	9	12	11	3	7
---	---	---	---	----	---	---	---	----	----	---	---

# Counting Inversions: Divide-and-Conquer

- Divide-and-conquer.
  - **Divide**: separate list into two pieces.

1	5	4	8	10	2	6	9	12	11	3	7
---	---	---	---	----	---	---	---	----	----	---	---

Divide:  $O(1)$ .

1	5	4	8	10	2	6	9	12	11	3	7
---	---	---	---	----	---	---	---	----	----	---	---

# Counting Inversions: Divide-and-Conquer

- Divide-and-conquer.
  - Divide: separate list into two pieces.
  - **Conquer**: recursively count inversions in each half.



Divide:  $O(1)$ .



Conquer:  $2T(n/2)$

5 blue-blue inversions

8 green-green inversions

5-4, 5-2, 4-2, 8-2, 10-2

6-3, 9-3, 9-7, 12-3, 12-7, 12-11, 11-3, 11-7

# Counting Inversions: Divide-and-Conquer

- Divide-and-conquer.

- Divide: separate list into two pieces.
- Conquer: recursively count inversions in each half.
- **Combine**: count inversions where  $a_i$  and  $a_j$  are in different halves, and return sum of three quantities.



Divide:  $O(1)$ .



Conquer:  $2T(n/2)$

5 blue-blue inversions

8 green-green inversions

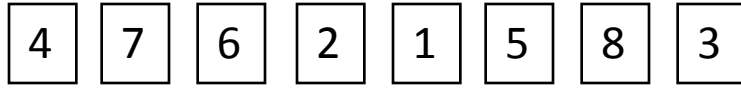
9 blue-green inversions

5-3, 4-3, 8-6, 8-3, 8-7, 10-6, 10-9, 10-3, 10-7

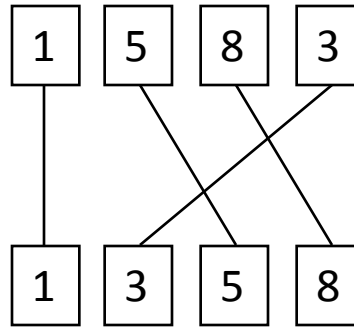
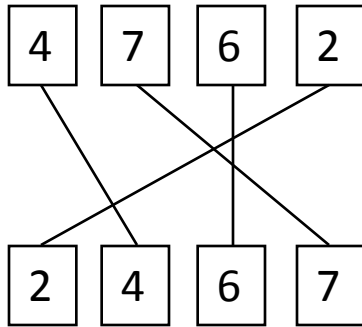
**Combine**: ???

Total =  $5 + 8 + 9 = 22$ .

# Counting Inversions



Divide into front and back halves



Count inversions within each half, and sort each half

Then, count inversions between front and back half



# Counting Inversions: Implementation

---

Sort-and-Count( $L$ )

    If the list has one element then  
        there are no inversions

Else

    Divide the list into two halves:

$A$  contains the first  $\lceil n/2 \rceil$  elements

$B$  contains the remaining  $\lfloor n/2 \rfloor$  elements

$(r_A, A) = \text{Sort-and-Count}(A)$

$(r_B, B) = \text{Sort-and-Count}(B)$

$(r, L) = \text{Merge-and-Count}(A, B)$

Endif

Return  $r = r_A + r_B + r$ , and the sorted list  $L$

---

# Counting Inversions: Implementation

---

Merge-and-Count( $A, B$ )

Maintain a *Current* pointer into each list, initialized to point to the front elements

Maintain a variable *Count* for the number of inversions, initialized to 0

While both lists are nonempty:

Let  $a_i$  and  $b_j$  be the elements pointed to by the *Current* pointer

Append the smaller of these two to the output list

If  $b_j$  is the smaller element then

Increment *Count* by the number of elements remaining in  $A$

Endif

Advance the *Current* pointer in the list from which the smaller element was selected.

EndWhile

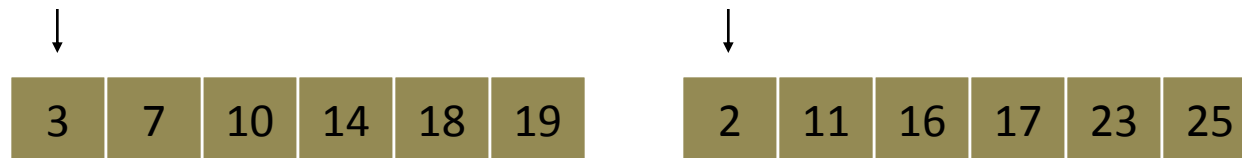
Once one list is empty, append the remainder of the other list to the output

Return *Count* and the merged list

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

$i = 6$



two sorted halves



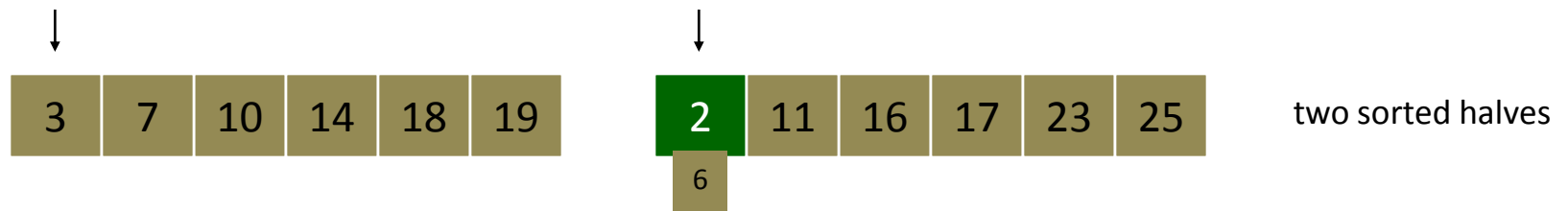
auxiliary array

Total:

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

$i = 6$

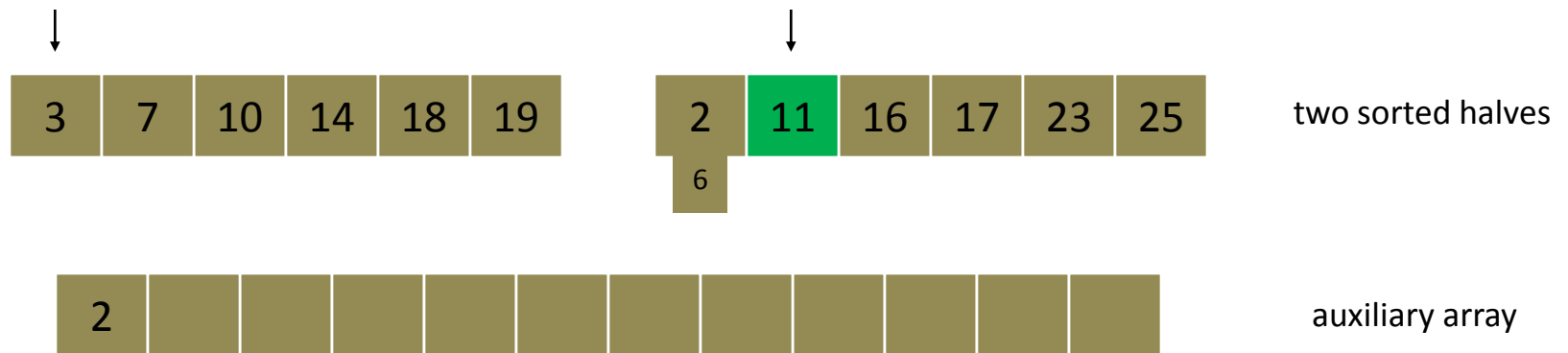


Total: 6

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

$i = 6$

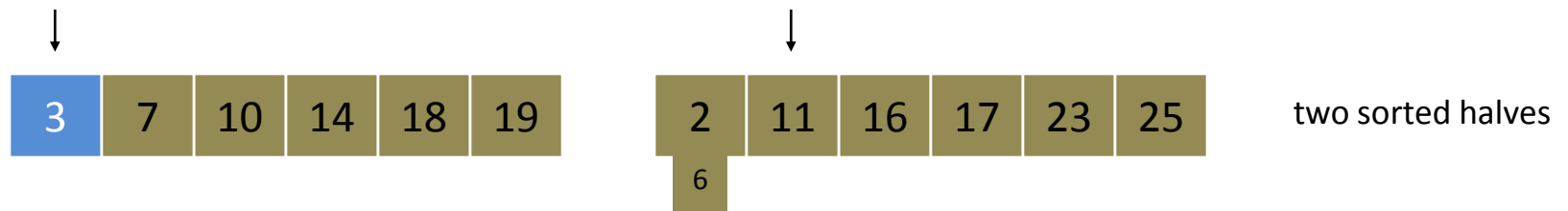


Total: 6

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

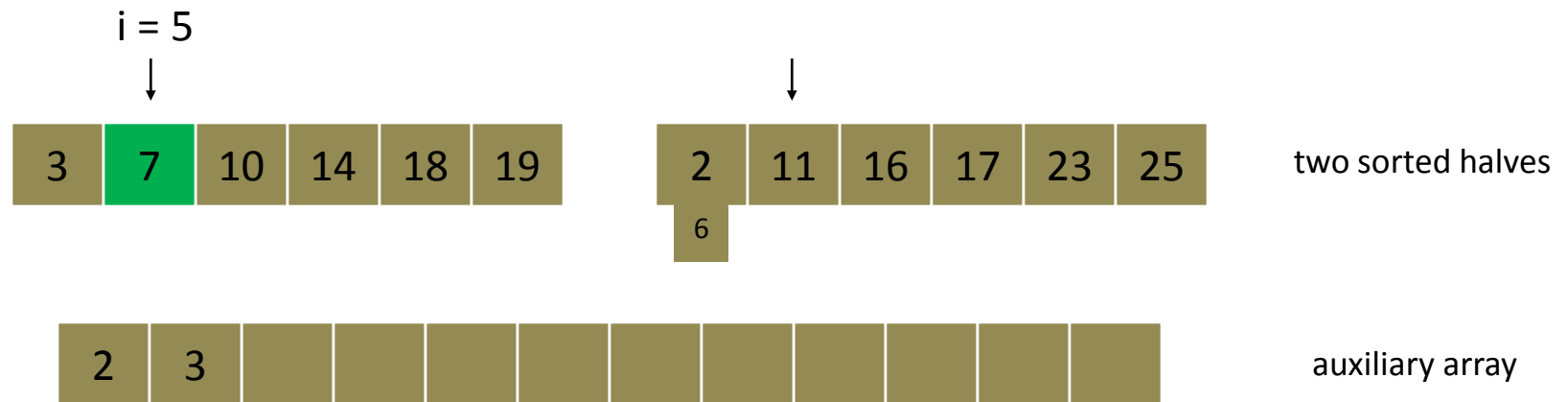
$i = 6$



Total: 6

# Counting Inversions

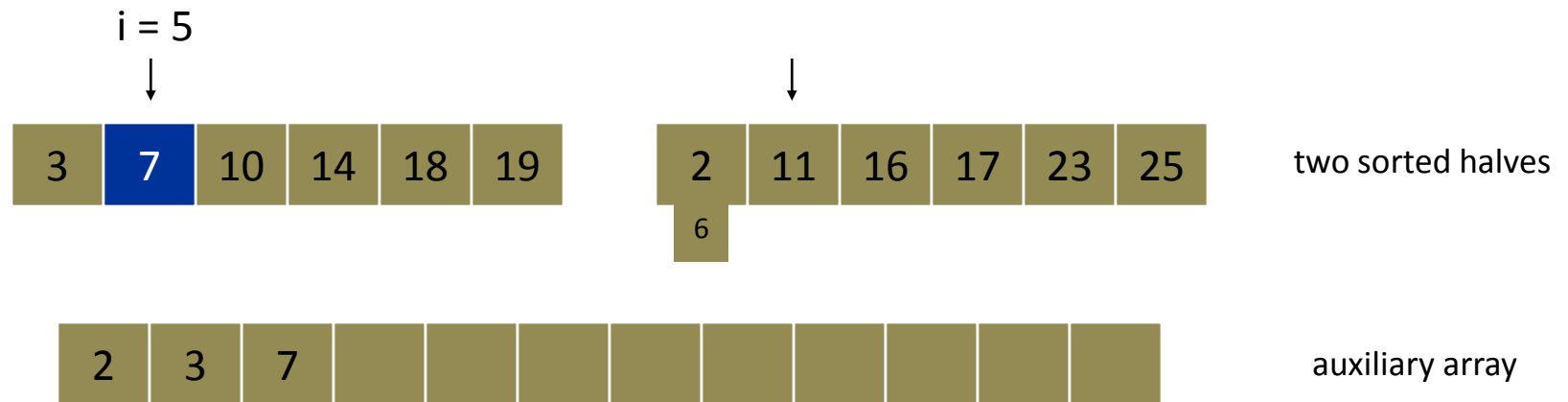
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

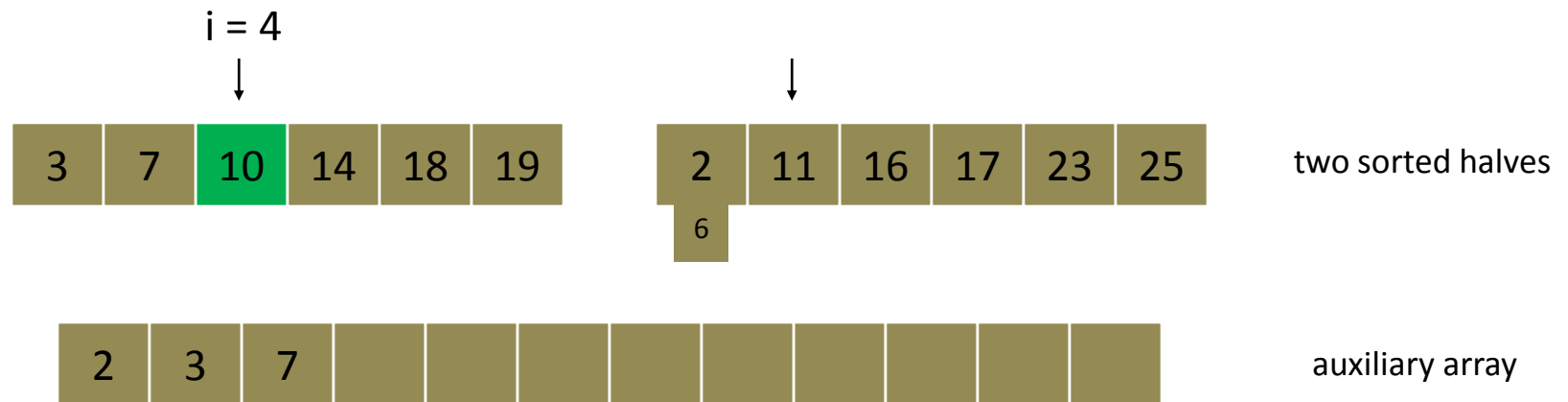


Total: 6



# Counting Inversions

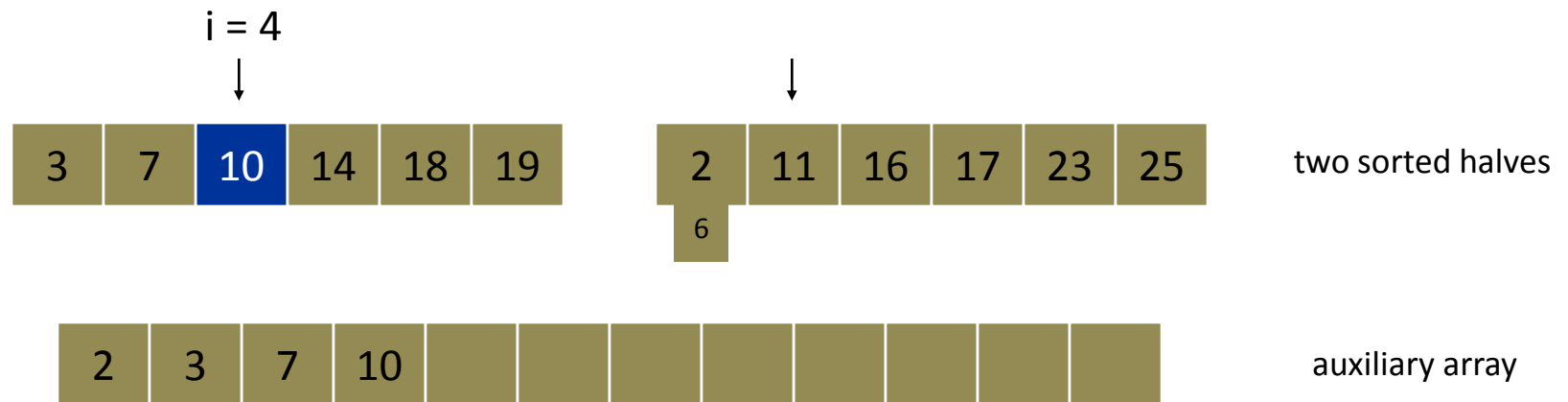
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6

# Counting Inversions

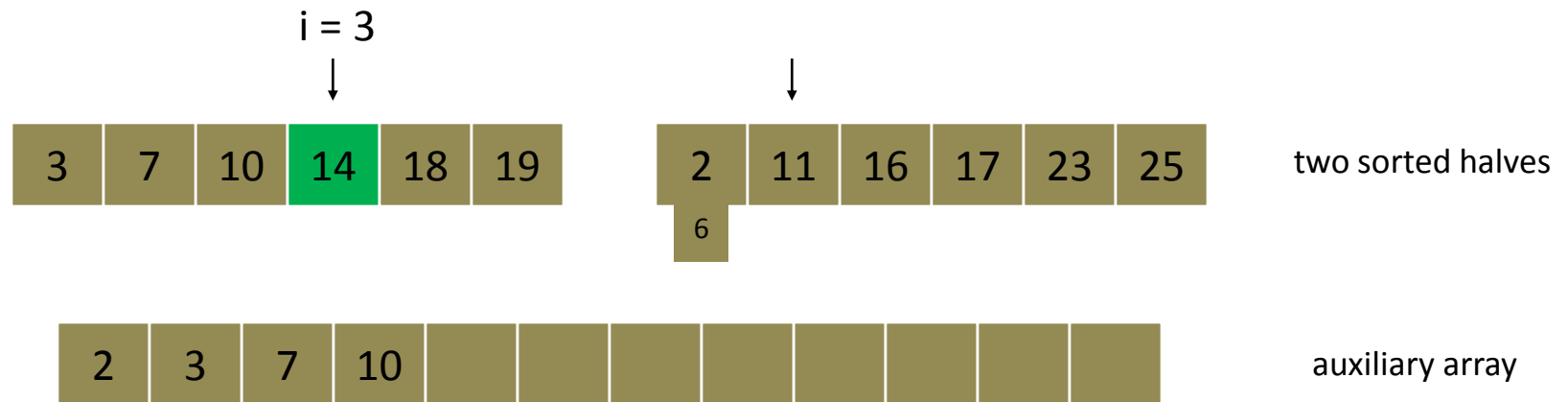
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6

# Counting Inversions

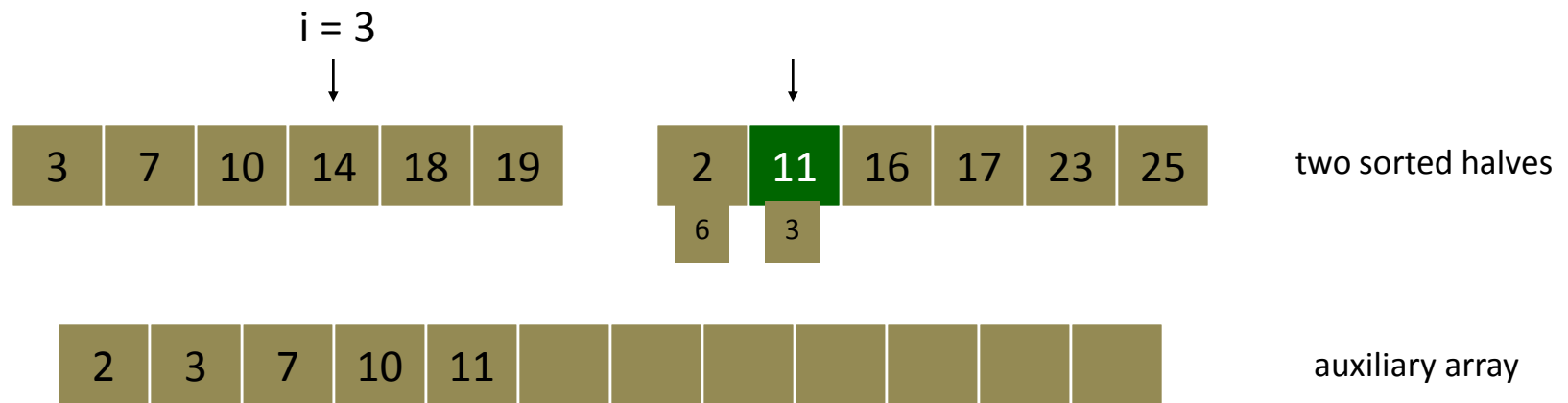
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6

# Counting Inversions

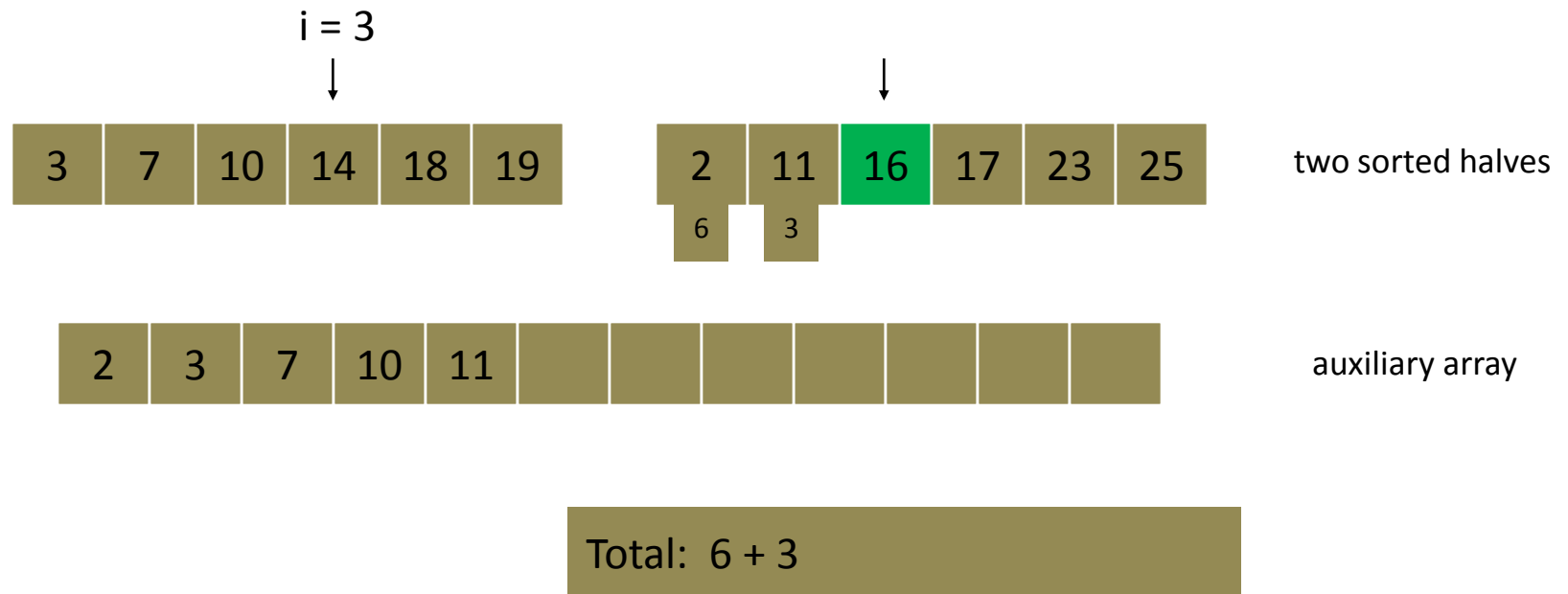
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6 + 3

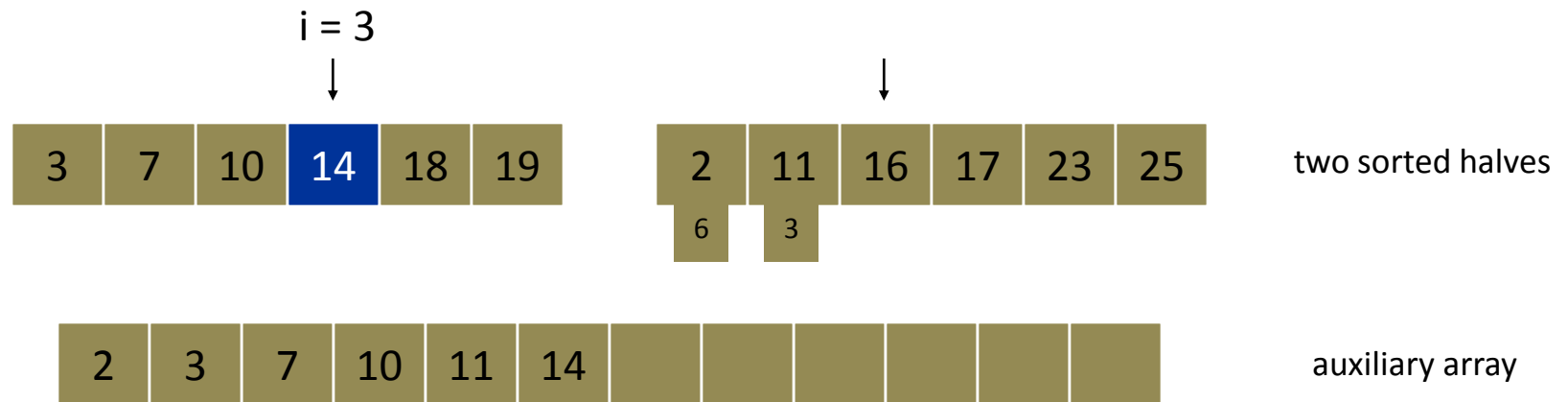
# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



# Counting Inversions

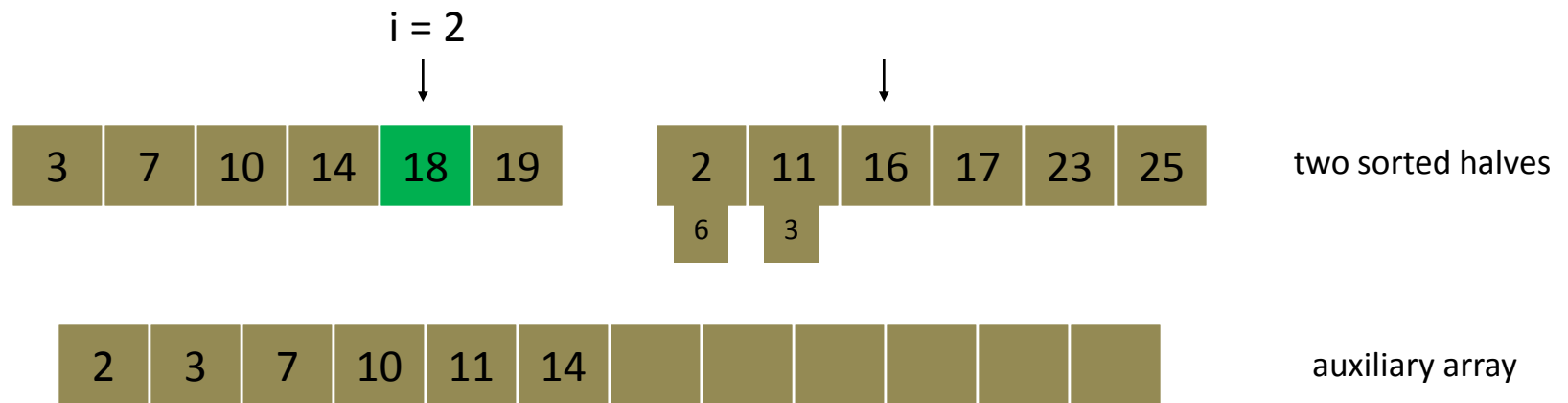
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6 + 3

# Counting Inversions

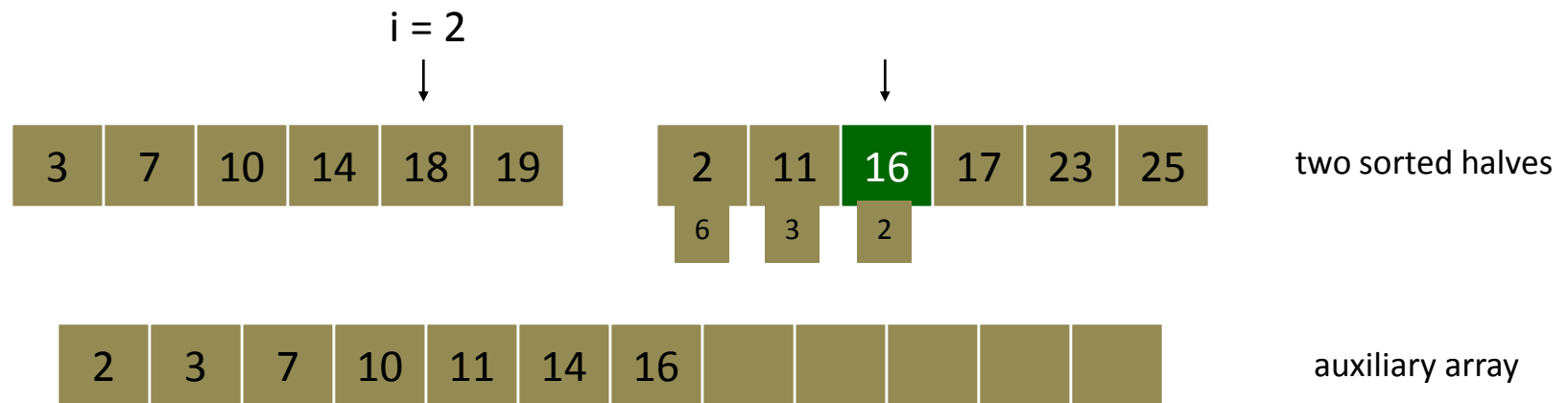
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total: 6 + 3

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

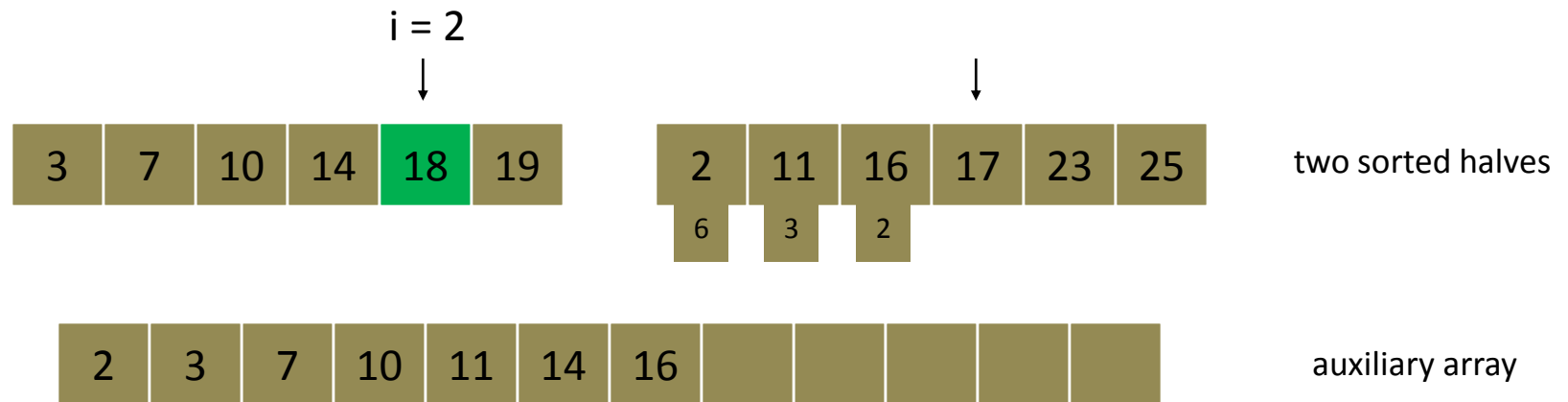


Total:  $6 + 3 + 2$



# Counting Inversions

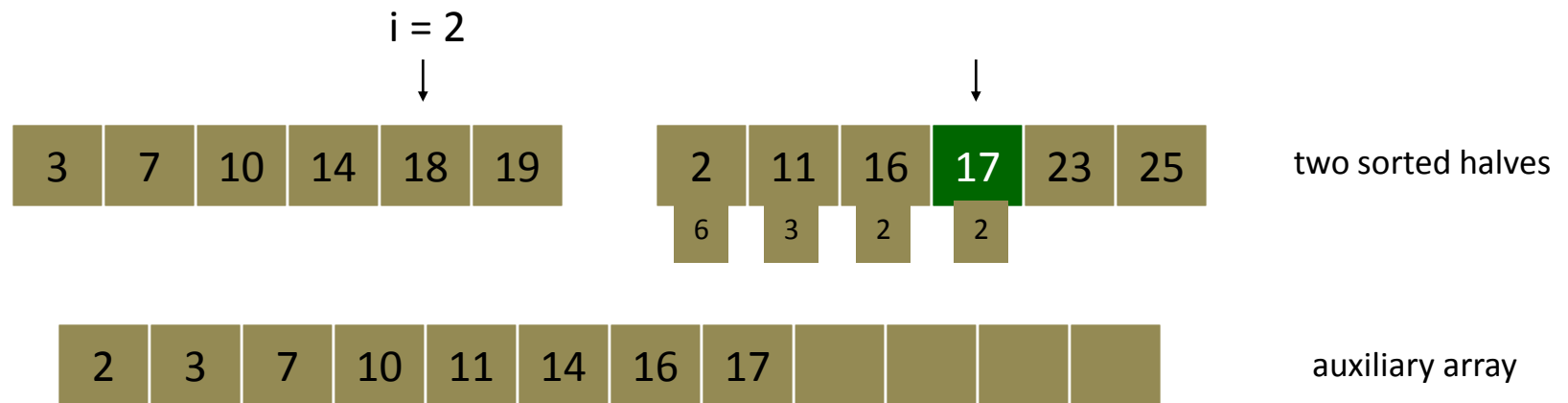
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2$

# Counting Inversions

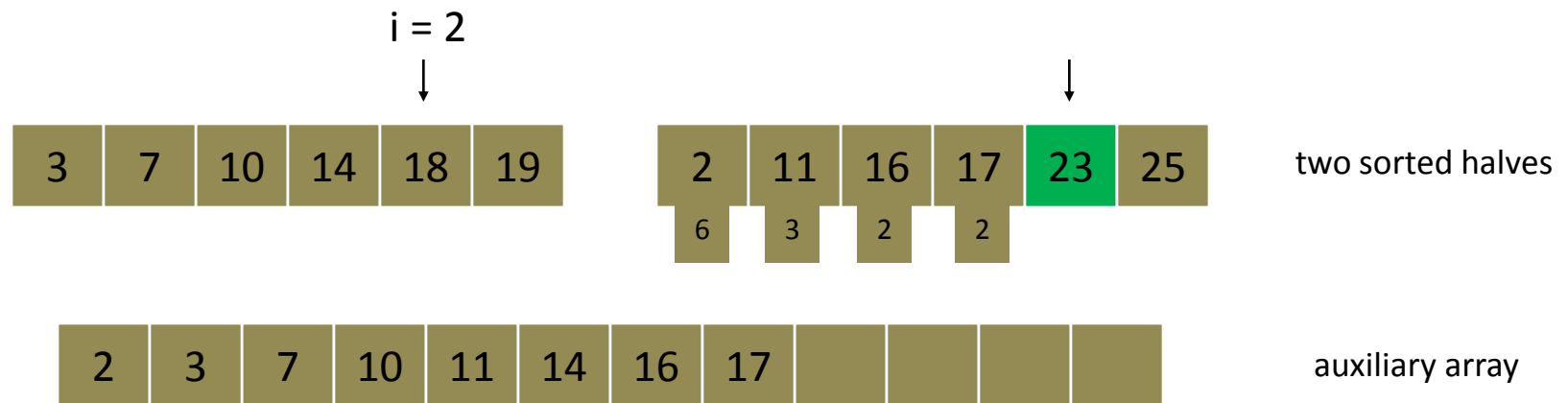
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2$

# Counting Inversions

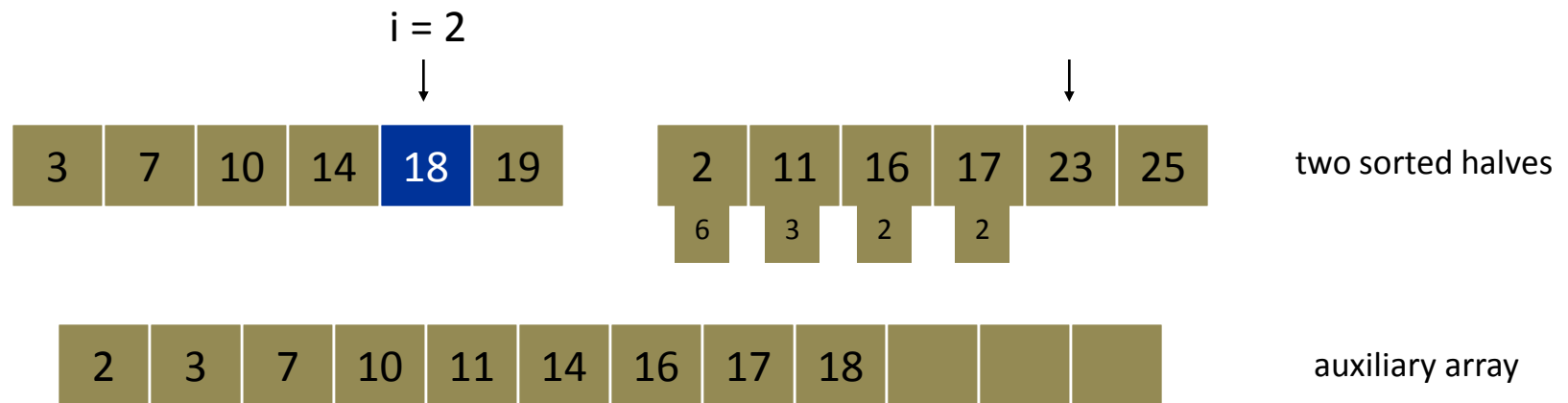
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2$

# Counting Inversions

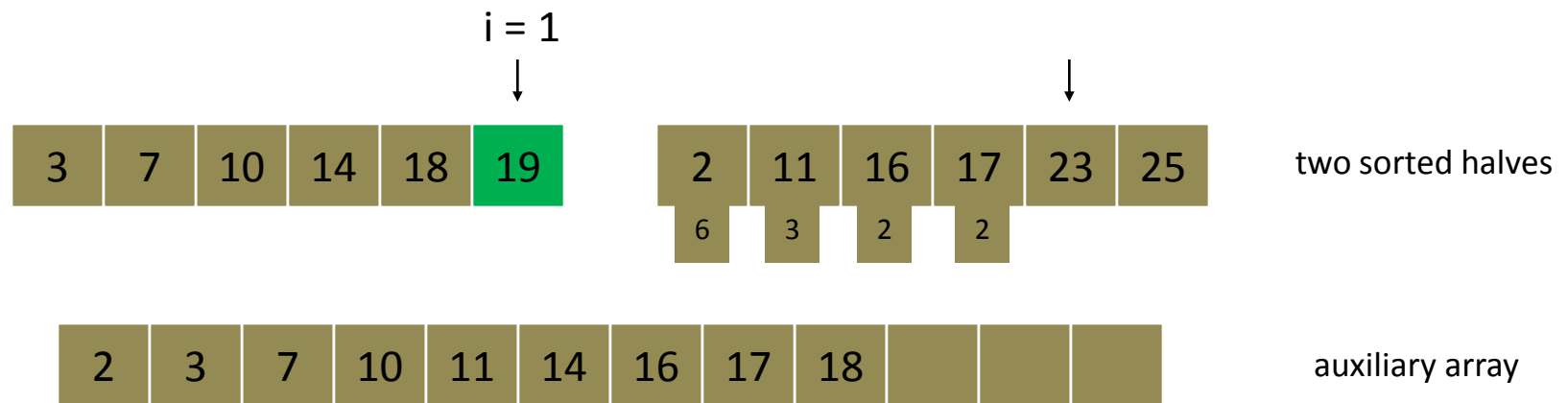
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2$

# Counting Inversions

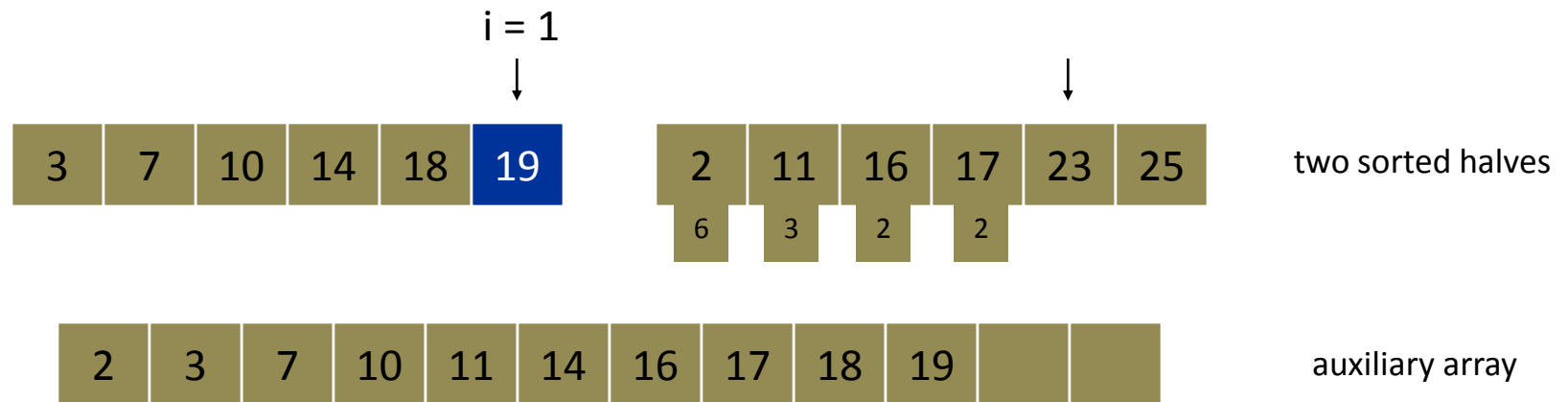
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2$

# Counting Inversions

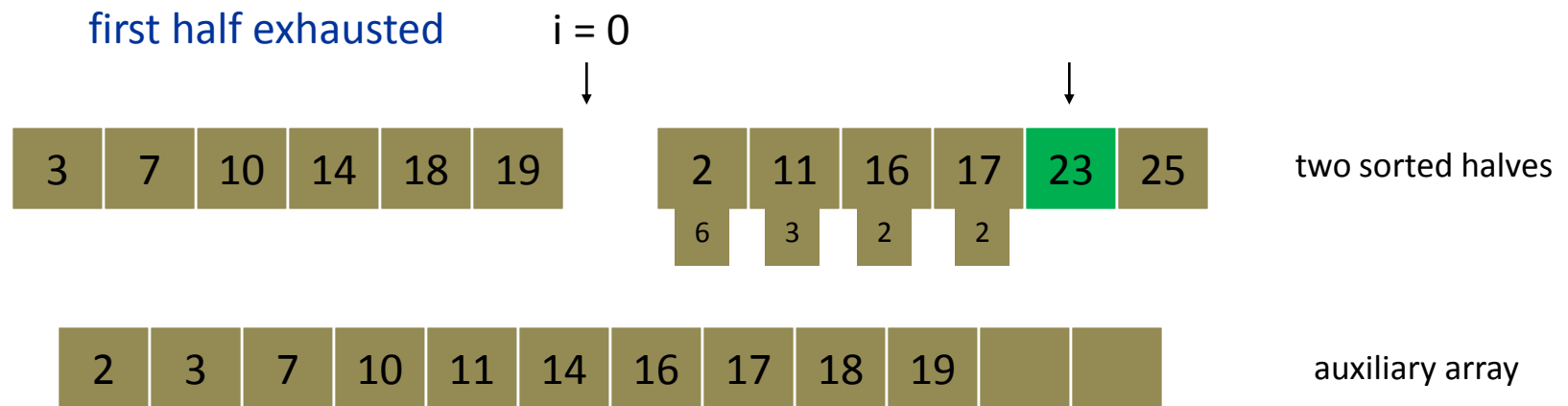
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2$

# Counting Inversions

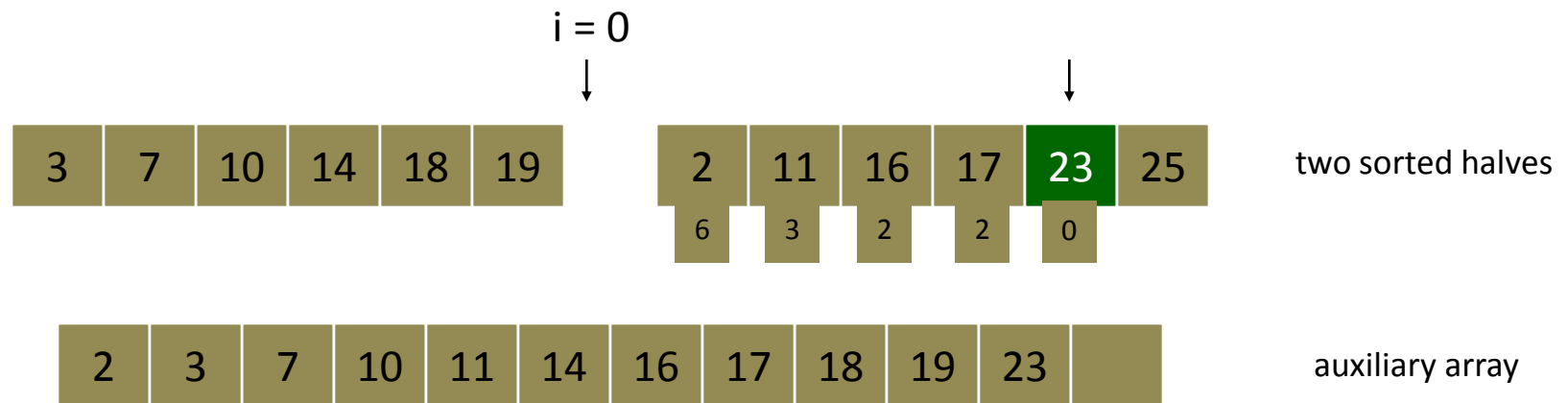
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2$

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.

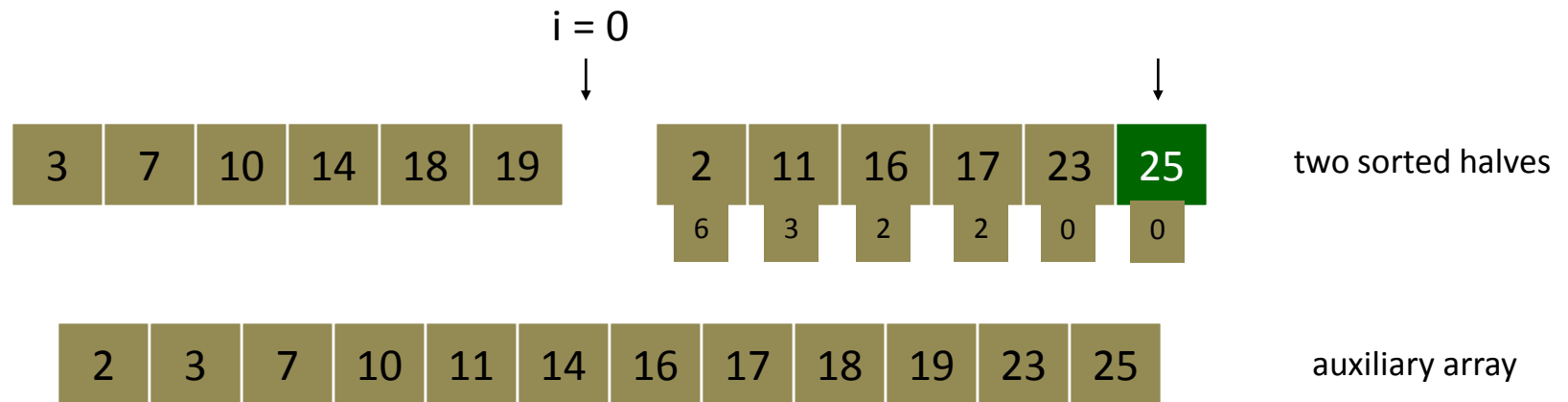


Total:  $6 + 3 + 2 + 2 + 0$



# Counting Inversions

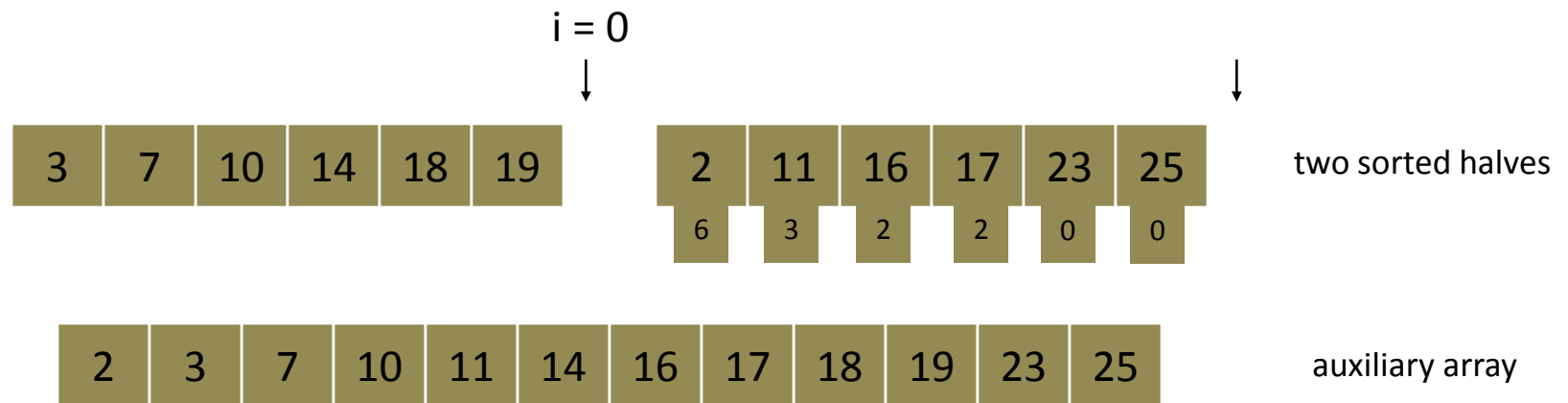
- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2 + 0 + 0$

# Counting Inversions

- Merge and count step.
  - Given two sorted halves, count number of inversions where  $a_i$  and  $a_j$  are in different halves.
  - Combine two sorted halves into sorted whole.



Total:  $6 + 3 + 2 + 2 + 0 + 0 = 13$

- **The e-commerce application**

Have you ever noticed on any e-commerce website, they have this section of "You might like", they have maintained an array for all the user accounts and then whichever has the least number of inversion with your array of choices, they start recommending what they have bought or they like