COMP 355
Advanced Algorithms

Divide and Conquer: Inversion Counting
KT: 5.1-5.3

Rhodes College
—1848—

Divide-and-Conquer

- Divide-and-conquer.
  - **Divide**: Break up problem into several parts.
  - **Conquer**: Solve each part recursively.
  - **Combine**: Merge solutions to sub-problems into overall solution.

- Most common usage.
  - Break up problem of size n into two equal parts of size ½n.
  - Solve two parts recursively.
  - Combine two solutions into overall solution in linear time.

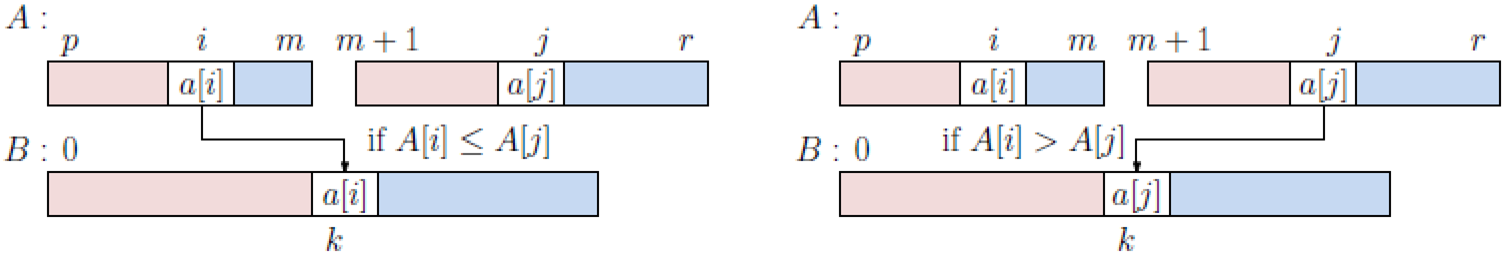
- Consequence.
  - Brute force: n^2.
  - Divide-and-conquer: n log n.

MergeSort

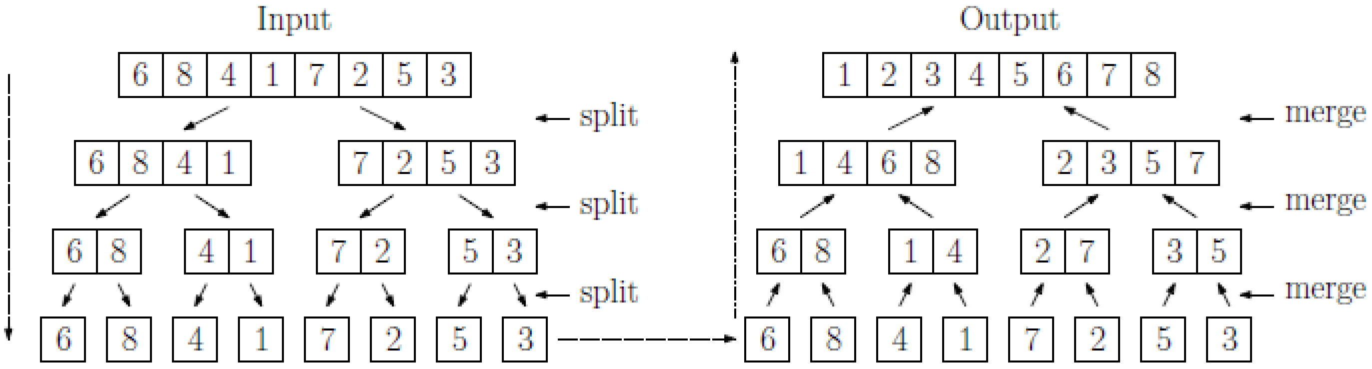
Basis case: If size(A) = 1, then the array is trivially sorted and we are done.

General case: Otherwise:

- **Divide:** Split A into two subsequences, each of size roughly $n/2$. (More precisely, one will be of size $\lfloor n/2 \rfloor$ and the other of size $\lceil n/2 \rceil$.)
- **Conquer:** Sort each subsequence (by calling MergeSort recursively on each).
- **Combine:** Merge the two sorted subsequences into a single sorted list.



Merging two sorted lists.



MergeSort example.

Inversion Counting

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, …, n.
- Your rank: $a_1, a_2, \ldots, a_n$.
- 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

Brute force: check all $\Theta(n^2)$ pairs i and j.

Applications

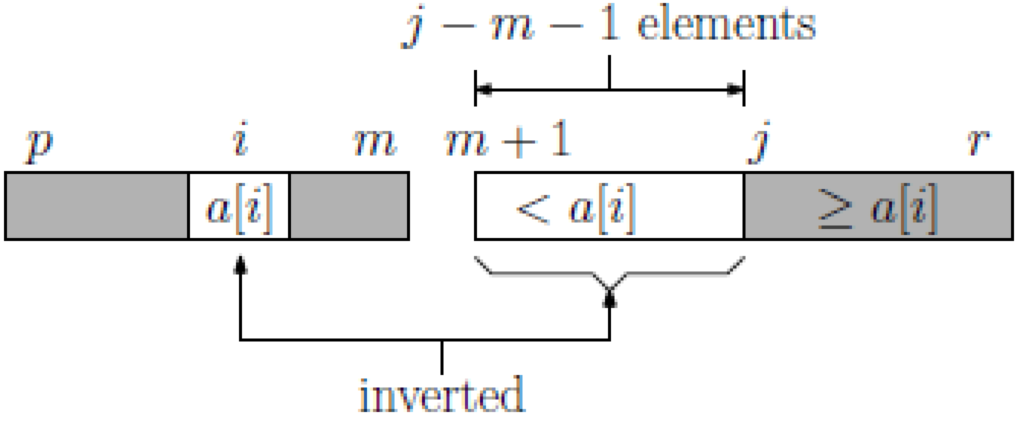
- Voting theory.
- Collaborative filtering.
- Measuring the "sortedness" of an array.
- Sensitivity analysis of Google's ranking function.
- Rank aggregation for meta-searching on the Web.

Divide-and-conquer solution

Basis case: If size$(A) = 1$, then there are no inversions.

General case: Otherwise:

- **Divide:** Split A into two subsequences, each of size roughly $n/2$.
- **Conquer:** Compute the number of inversions *within* each of the subsequences.
- **Combine:** Count the number of inversions occurring *between* the two sequences.



Counting inversions when $A[i] \leq A[j]$.

Inversion Counting Algorithm

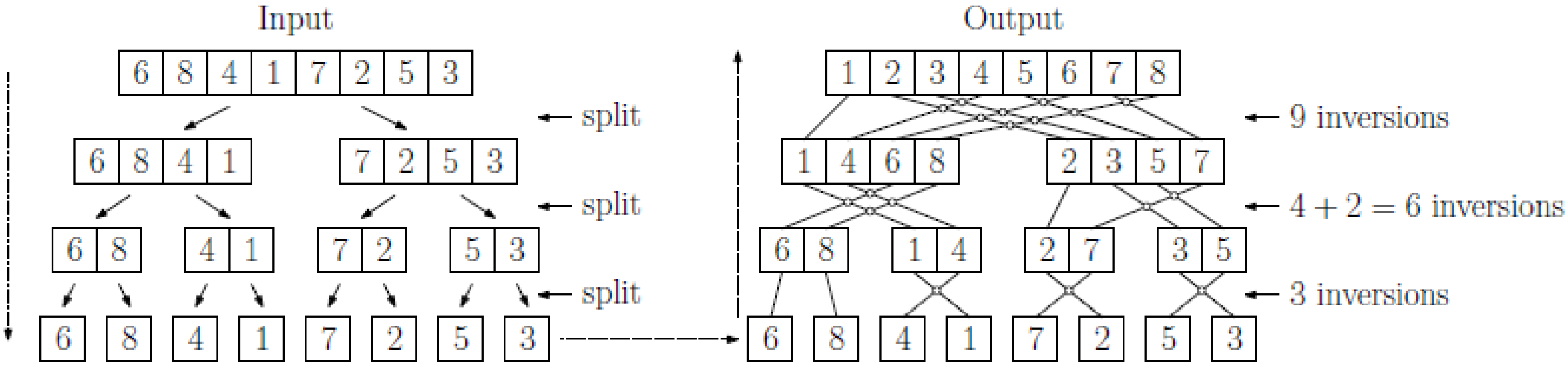
```
def InvCount(A):                     # sort A
    if len(A) <= 1:
        return A, 0                  # 1 element or fewer -> no inversions
    mid = int(len(A)/2)              # find midpoint
    left, x1 = InvCount(A[:mid])     # count inversions in the left half
    right, x2 = InvCount(A[mid:])    # count inversions in the right half
    A, x3 = invMerge(left, right)    # merge and count inversions
    return A, (x1 + x2 + x3)

def invMerge(A, B):                  # merges left and right lists
    m = []
    cnt = 0                          # inversion counter
    i = j = 0
    while i < len(A) and j < len(B): # while both subarrays are nonempty
        if A[i] < B[j]:
            m.append(A[i])           # take next item from left subarray
            i += 1                   # increment the left array counter
        else:
            m.append(B[j])           # take next item from right subarray
            cnt += len(A) - i        # increment the inversion counter
            j += 1

    m.extend(A[i:])                  # copy extras from left to m
    m.extend(B[j:])                  # copy extras from right to m

    return m, cnt
```

Inversion Counting Example



Inversion counting by divide and conquer.

Counting Inversions: Divide-and-Conquer

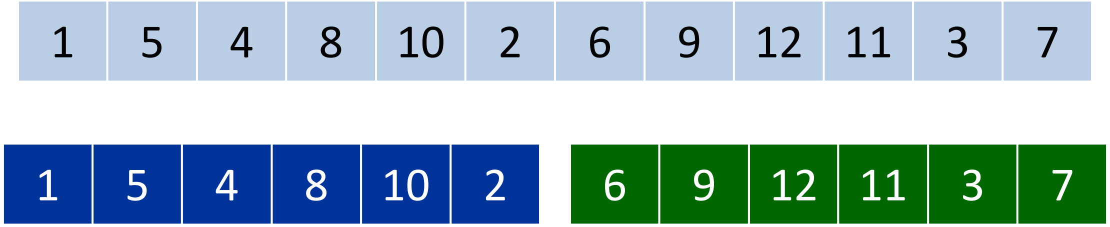
Divide-and-conquer.

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

1	5	4	8	10	2

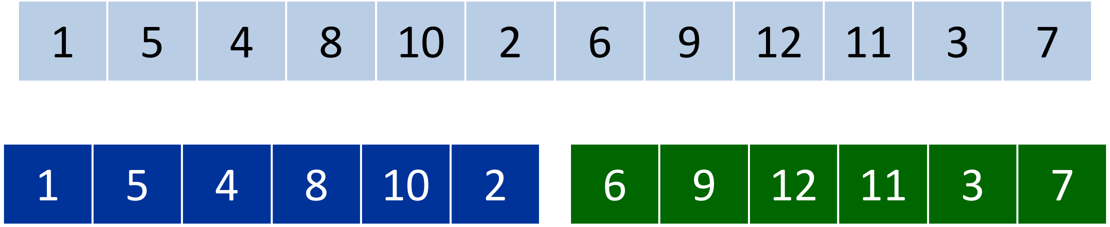
6	9	12	11	3	7

Divide: O(1).

Counting Inversions: Divide-and-Conquer

Divide-and-conquer.

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



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

Conquer: 2T(n / 2)

5 blue-blue inversions

8 green-green inversions

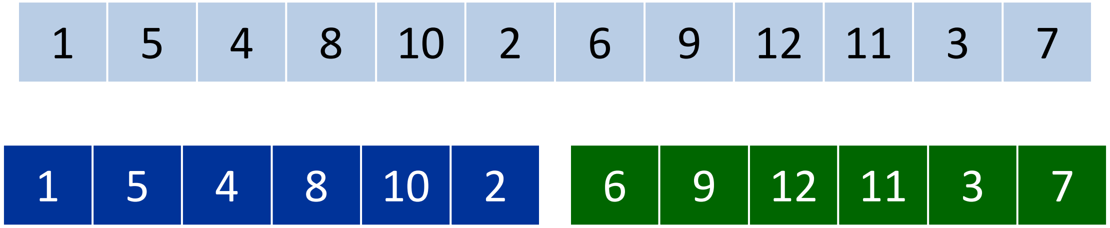
5-4, 5-2, 4-2, 8-2, 10-2

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.



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

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: Combine

Combine: count blue-green inversions

- Assume each half is sorted.
- Count inversions where a_i and a_j are in different halves.
- Merge two sorted halves into sorted whole.

to maintain sorted invariant

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2	2	0	0

13 blue-green inversions: 6 + 3 + 2 + 2 + 0 + 0

Count: O(n)

2	3	7	10	11	14	16	17	18	19	23	25

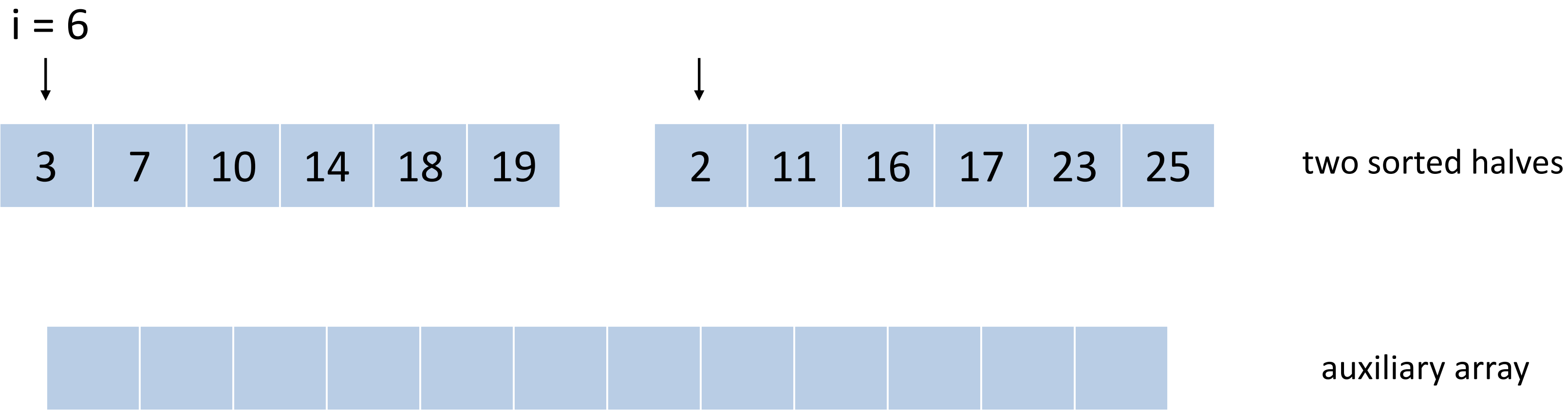
Merge: O(n)

$$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \quad \Rightarrow \quad \mathrm{T}(n) = O(n \log n)$$

Merge and Count

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

3	7	10	14	18	19

2	11	16	17	23	25

two sorted halves

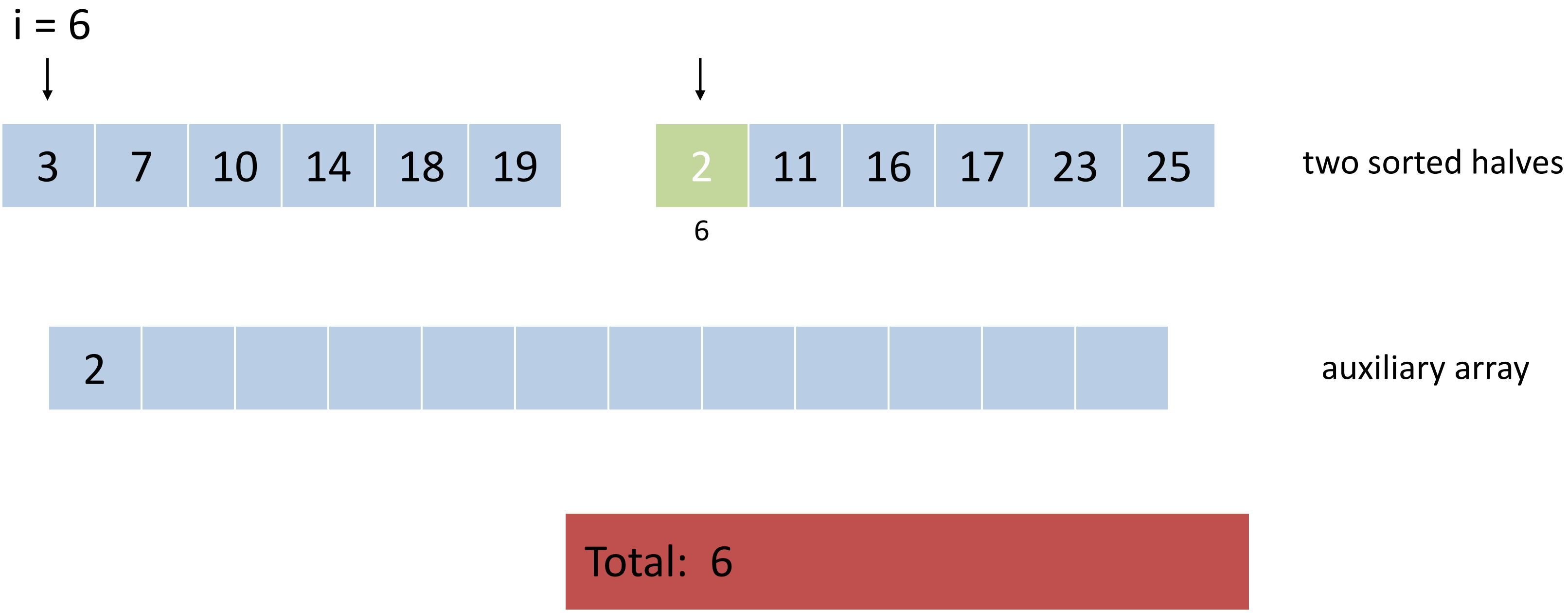
auxiliary array

Total:

Merge and Count

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

3	7	10	14	18	19		🟩 2	11	16	17	23	25
							6					

two sorted halves

2											

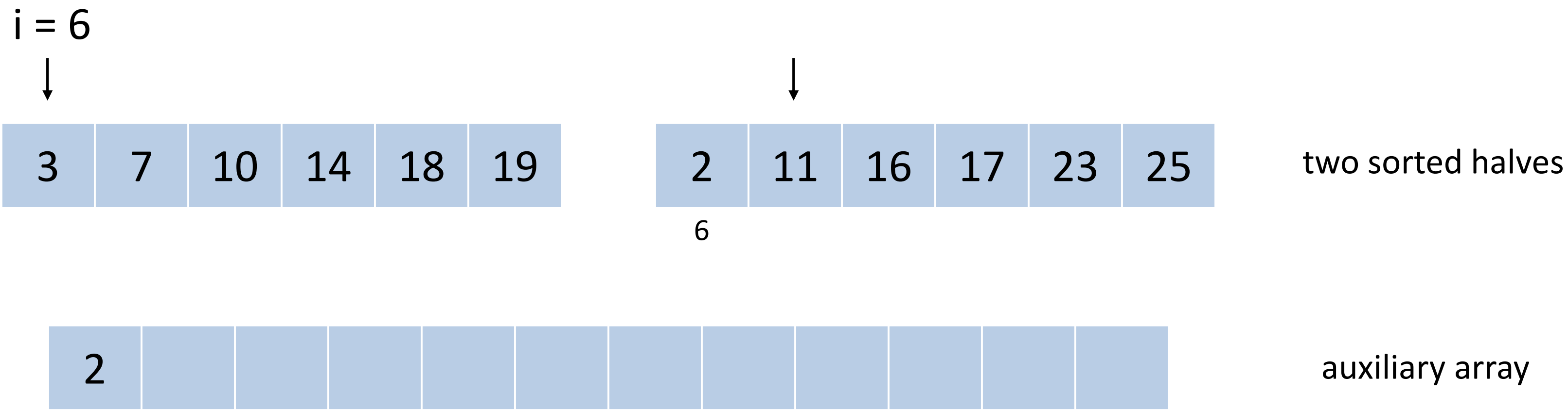
auxiliary array

Total: 6

Merge and Count

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

3	7	10	14	18	19	2	11	16	17	23	25
						6					

two sorted halves

2											

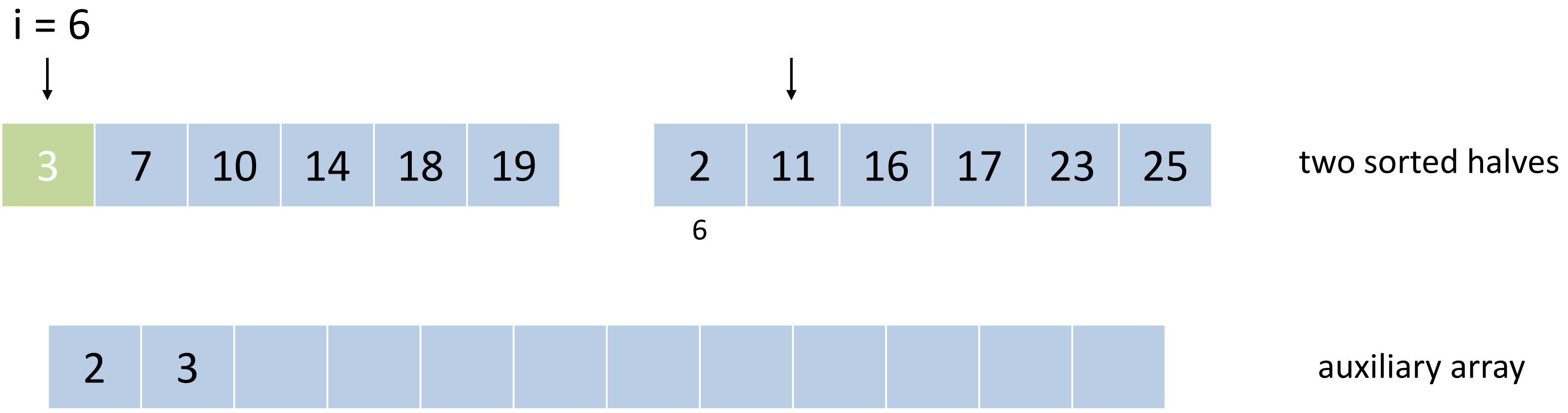
auxiliary array

Total: 6

Merge and Count

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.



3	7	10	14	18	19

2	11	16	17	23	25
6					

two sorted halves

2	3										

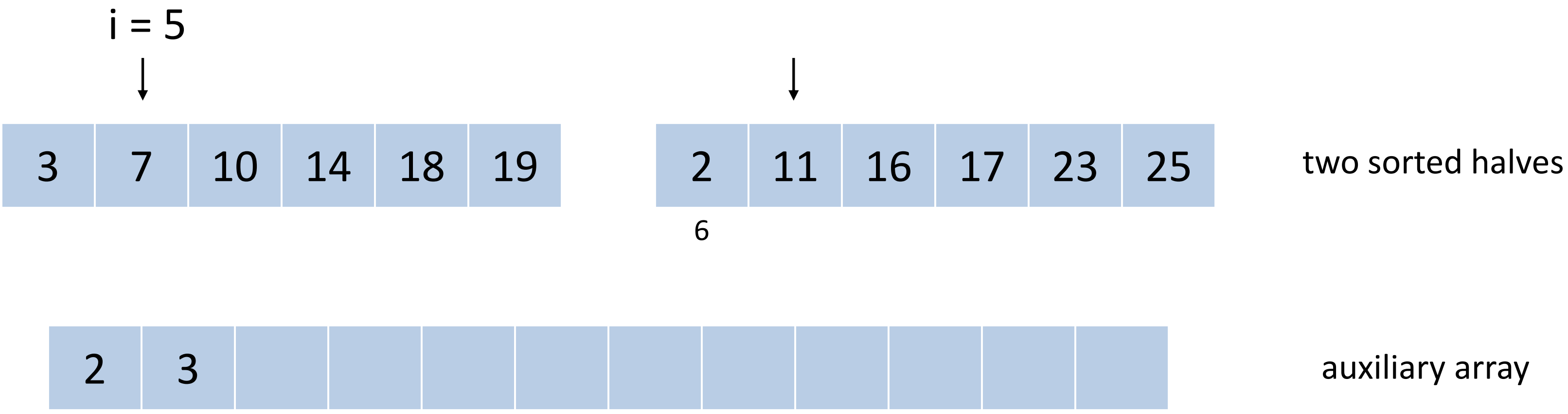
auxiliary array

Total: 6

Merge and Count

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 = 5

3	7	10	14	18	19	2	11	16	17	23	25
						6					

two sorted halves

2	3										

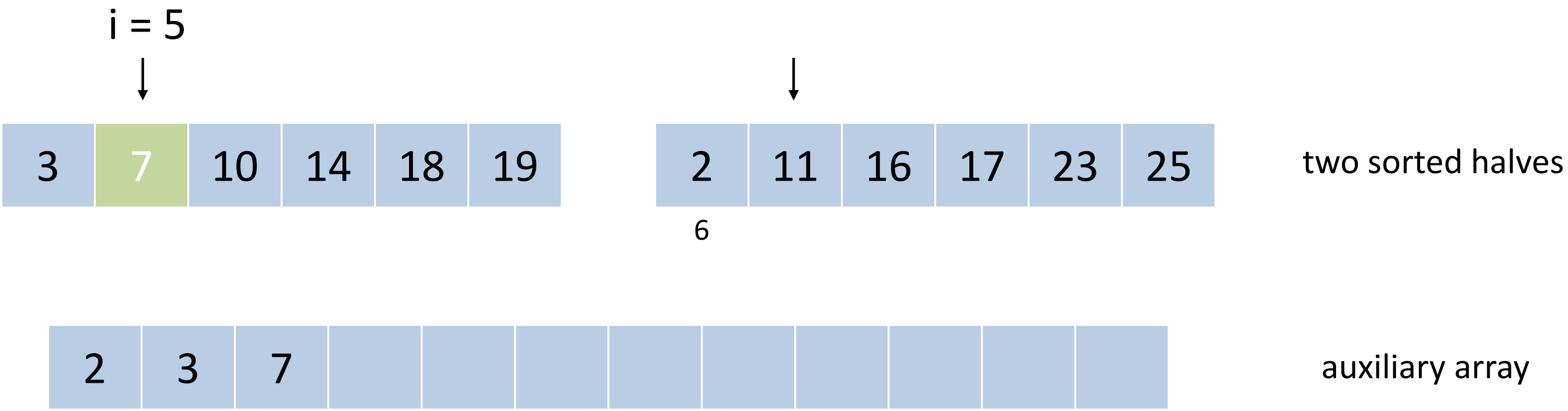
auxiliary array

Total: 6

Merge and Count

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 = 5

3	7	10	14	18	19

2	11	16	17	23	25

6

two sorted halves

2	3	7									

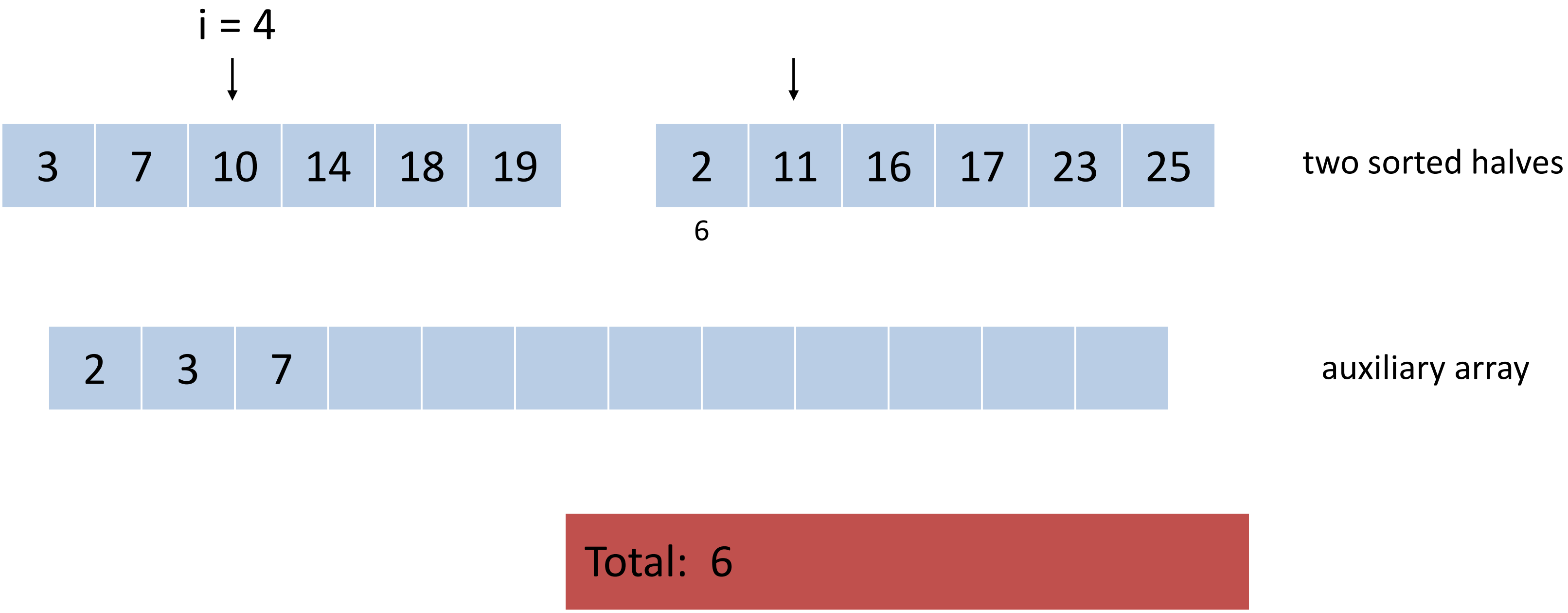
auxiliary array

Total: 6

Merge and Count

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 = 4

3	7	10	14	18	19

2	11	16	17	23	25

6

two sorted halves

2	3	7									

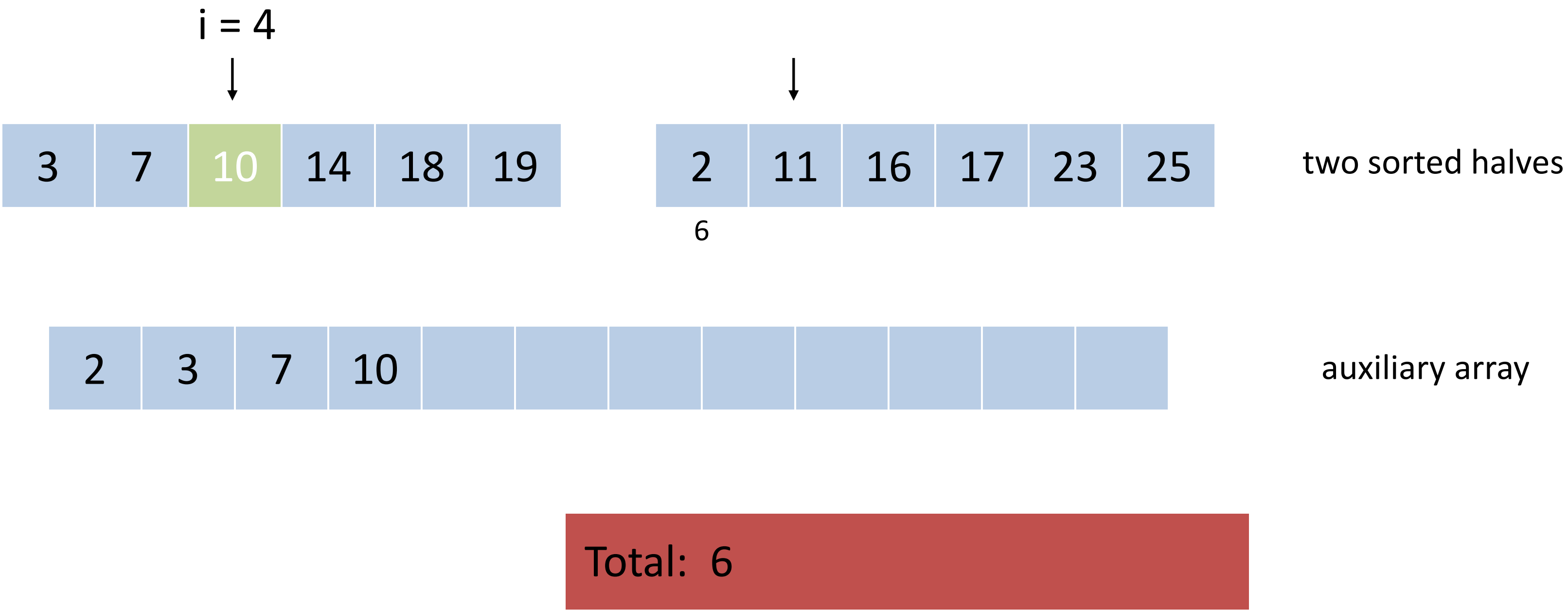
auxiliary array

Total: 6

Merge and Count

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 = 4

3	7	10	14	18	19		2	11	16	17	23	25
							6					

two sorted halves

2	3	7	10								

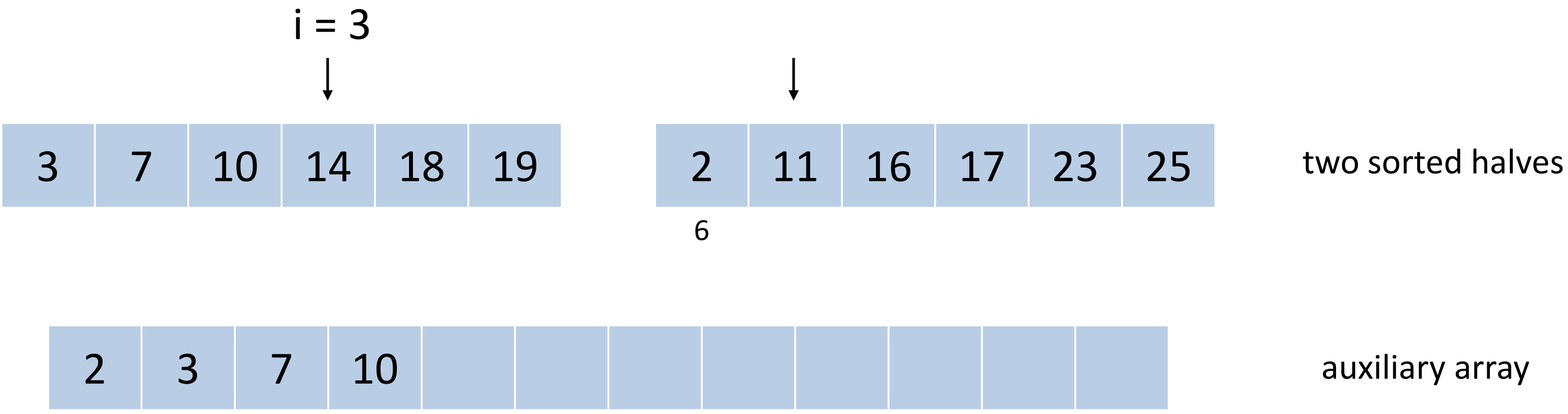
auxiliary array

Total: 6

Merge and Count

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.



3	7	10	14	18	19

2	11	16	17	23	25

two sorted halves

2	3	7	10								

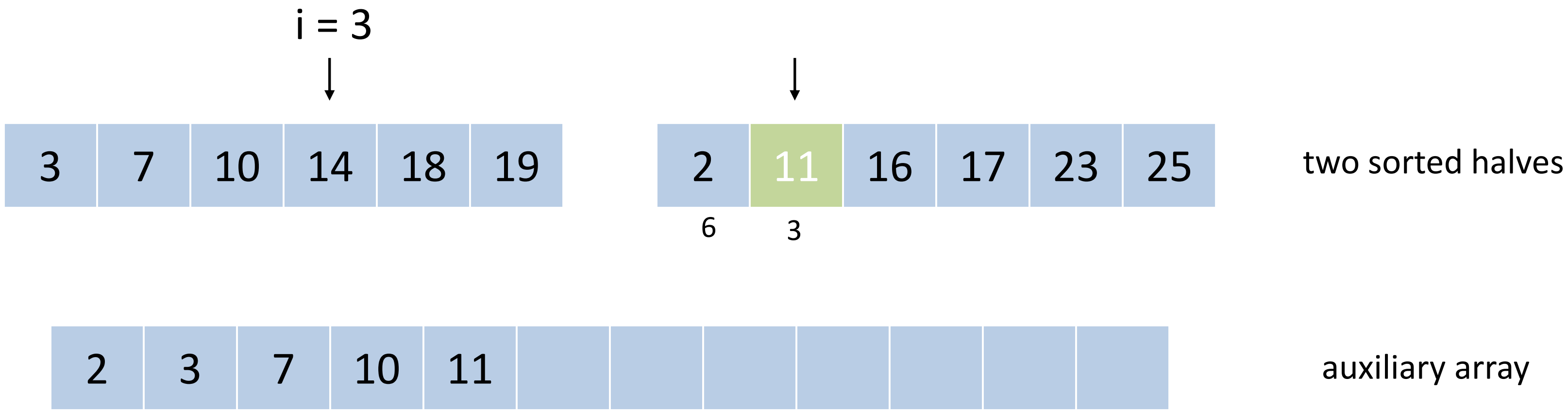
auxiliary array

Total: 6

Merge and Count

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 = 3

3	7	10	14	18	19	2	11	16	17	23	25	two sorted halves
						6	3					

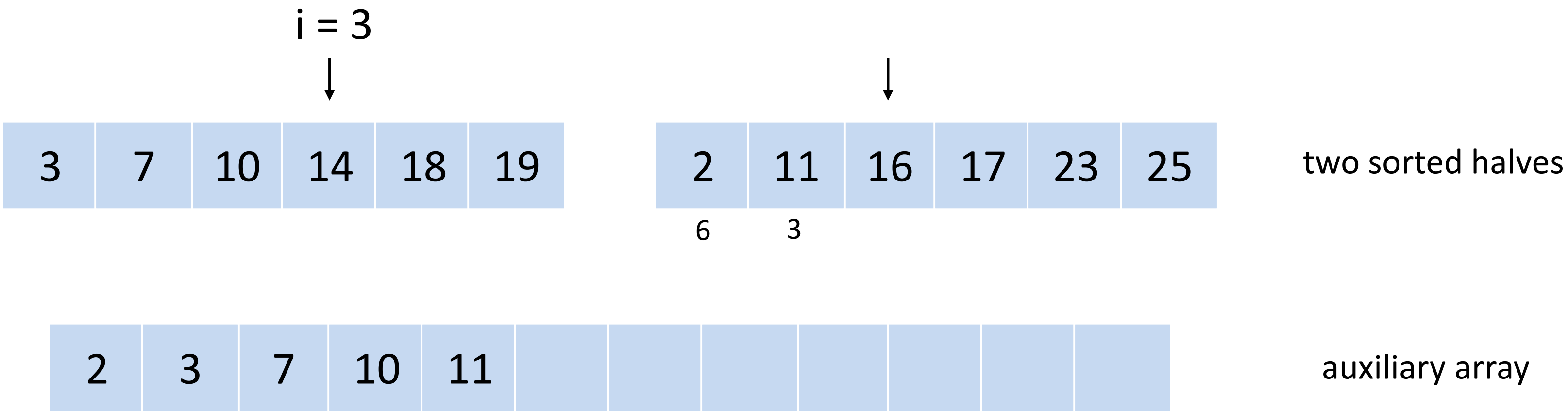
2	3	7	10	11								auxiliary array

Total: 6 + 3

Merge and Count

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 = 3

3	7	10	14	18	19	2	11	16	17	23	25	two sorted halves
						6	3					

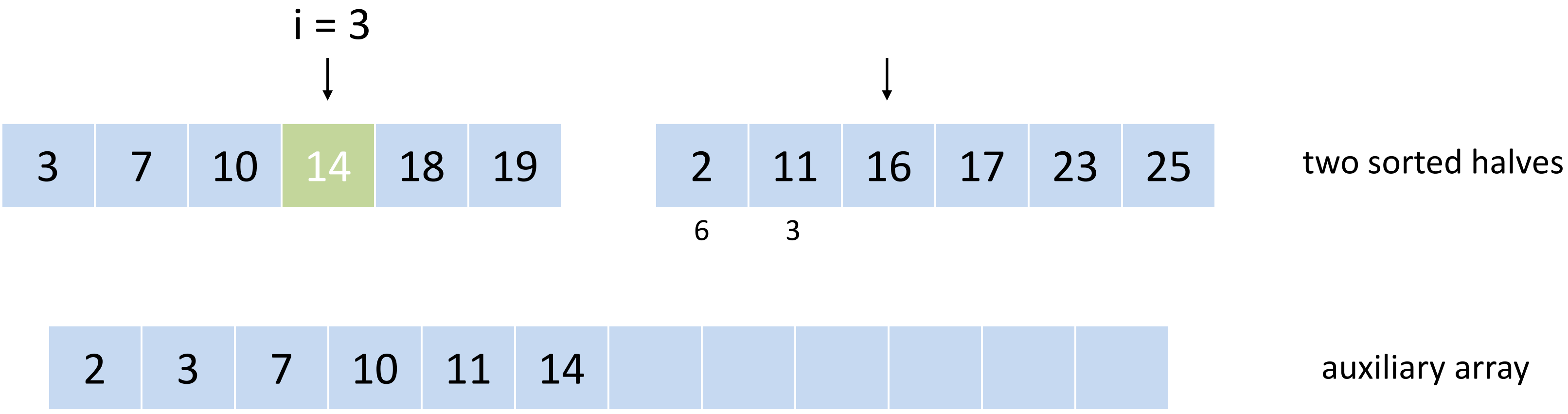
2	3	7	10	11								auxiliary array

Total: 6 + 3

Merge and Count

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.



3	7	10	14	18	19

2	11	16	17	23	25
6	3				

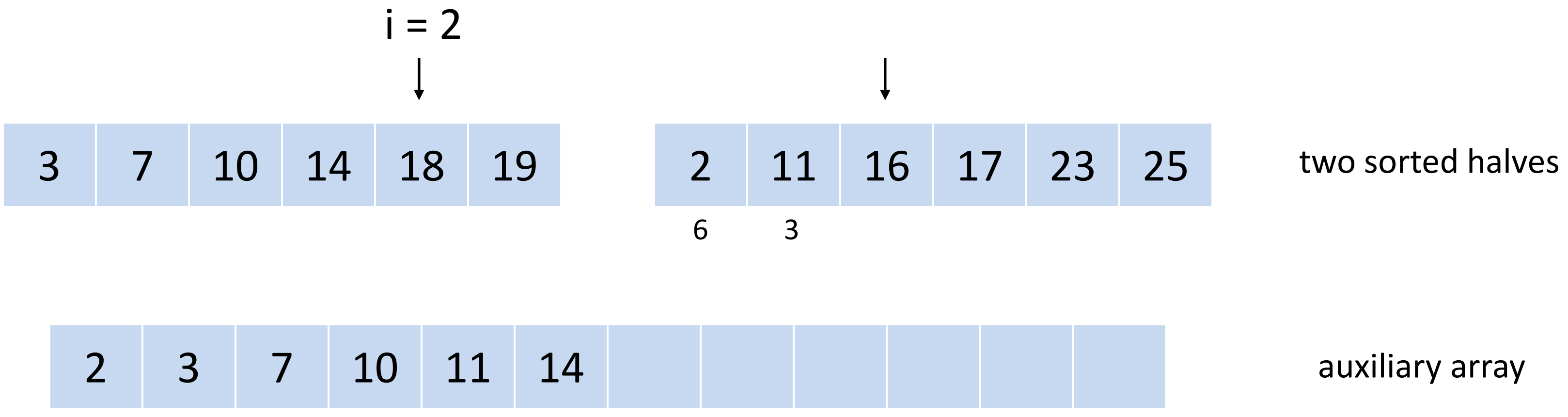
2	3	7	10	11	14						

Total: 6 + 3

Merge and Count

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.



3	7	10	14	18	19

2	11	16	17	23	25
6	3				

two sorted halves

2	3	7	10	11	14						

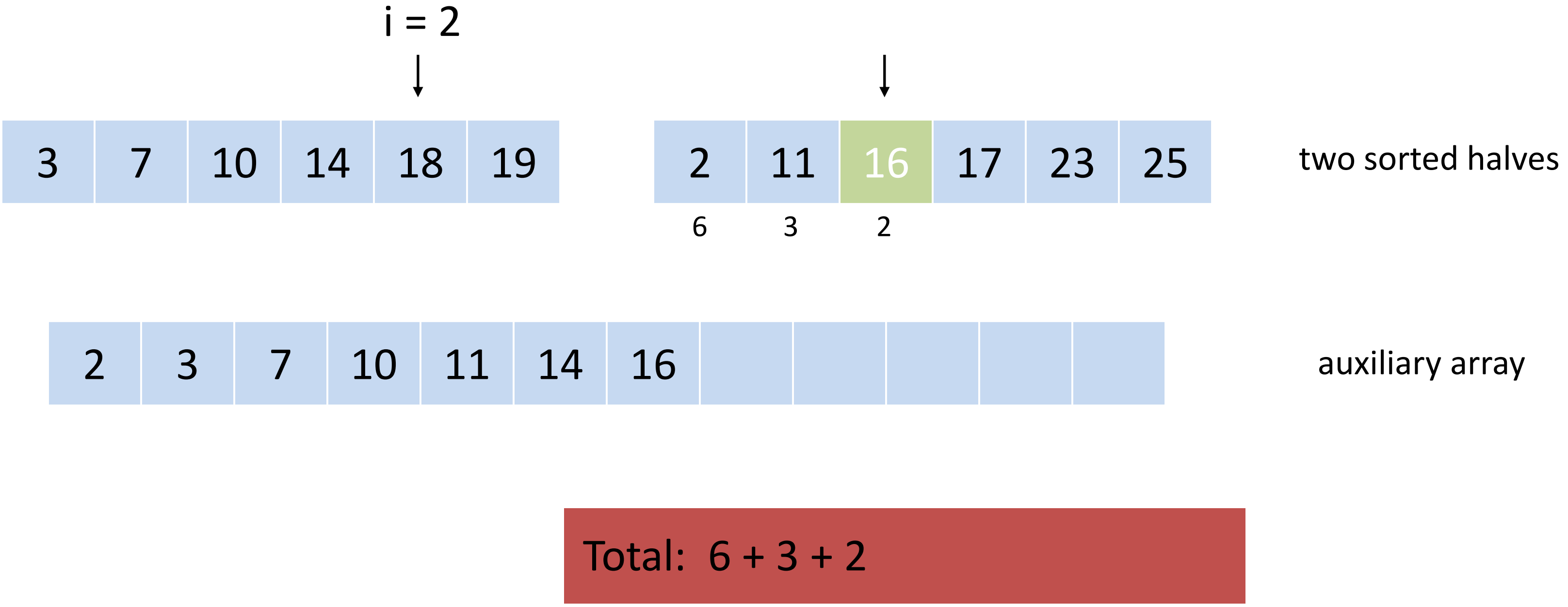
auxiliary array

Total: 6 + 3

Merge and Count

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 = 2

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2			

two sorted halves

2	3	7	10	11	14	16					

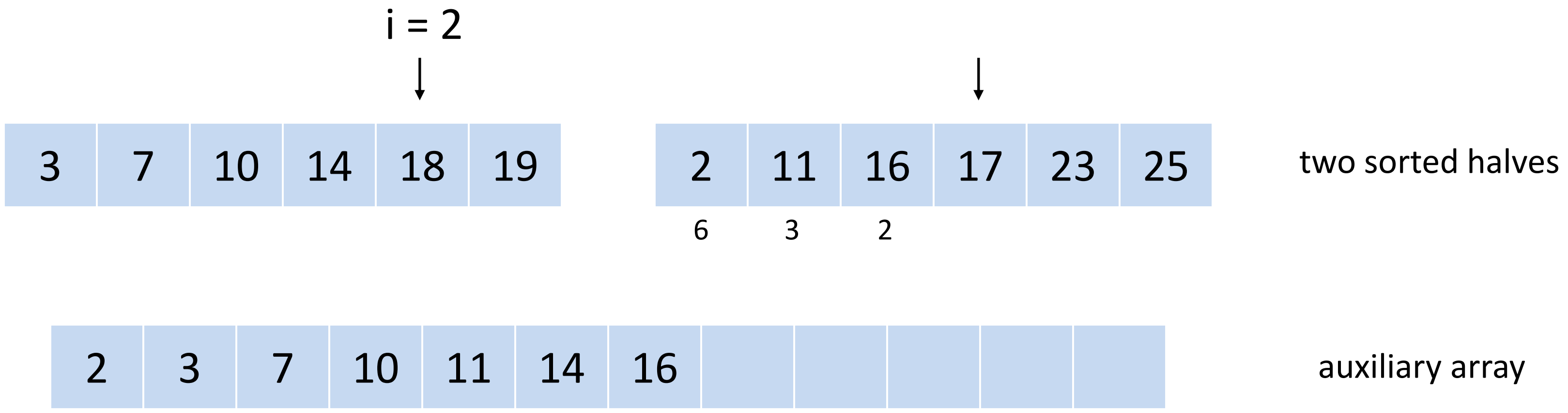
auxiliary array

Total: 6 + 3 + 2

Merge and Count

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 = 2

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2			

two sorted halves

2	3	7	10	11	14	16					

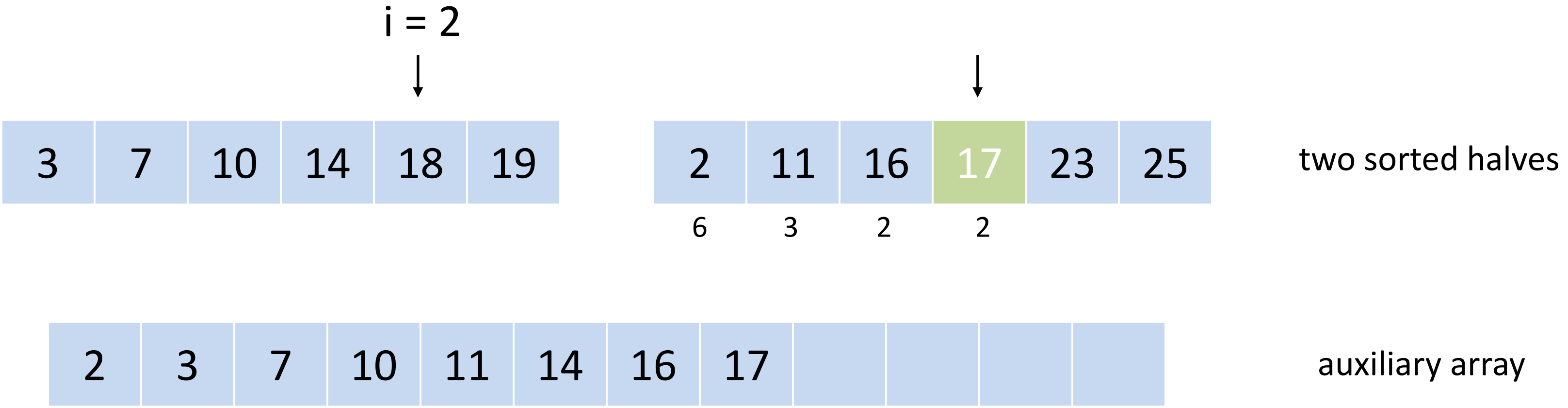
auxiliary array

Total: 6 + 3 + 2

Merge and Count

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 = 2

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2	2		

two sorted halves

2	3	7	10	11	14	16	17				

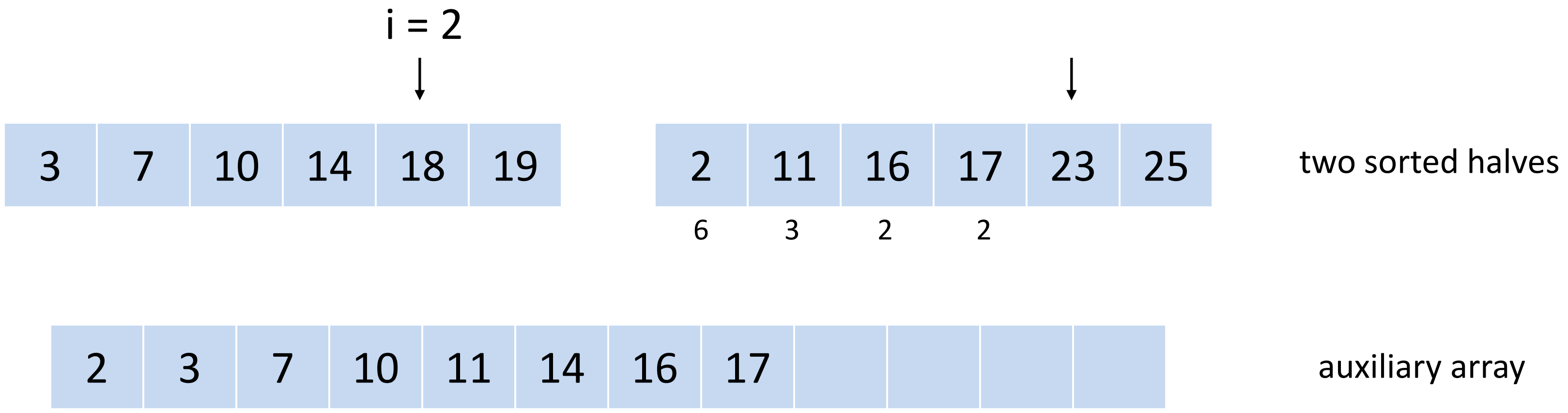
auxiliary array

Total: 6 + 3 + 2 + 2

Merge and Count

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 = 2

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2	2		

two sorted halves

2	3	7	10	11	14	16	17				

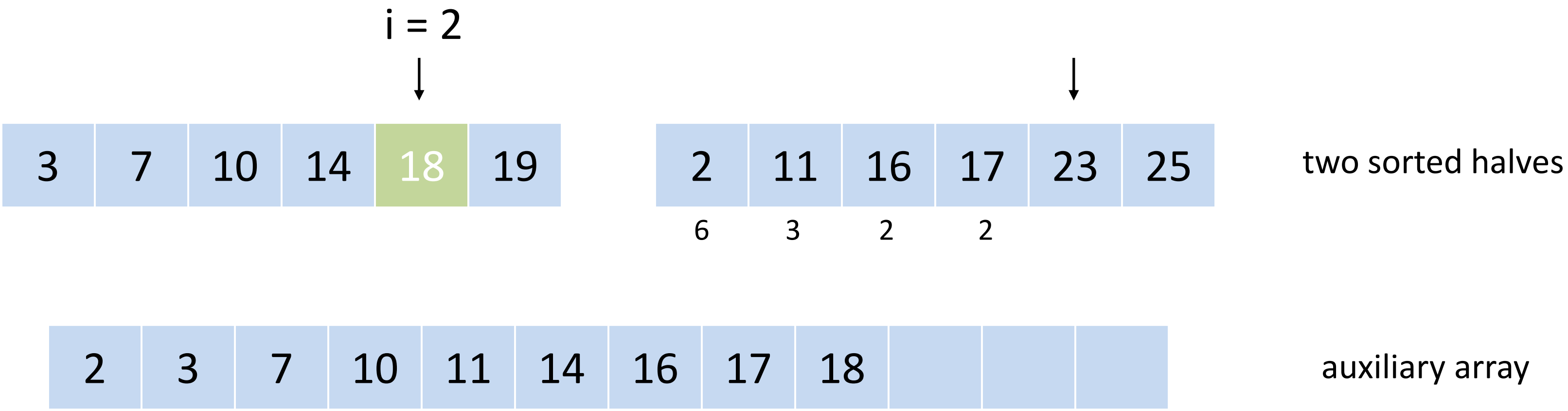
auxiliary array

Total: 6 + 3 + 2 + 2

Merge and Count

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 = 2

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2	2		

two sorted halves

2	3	7	10	11	14	16	17	18			

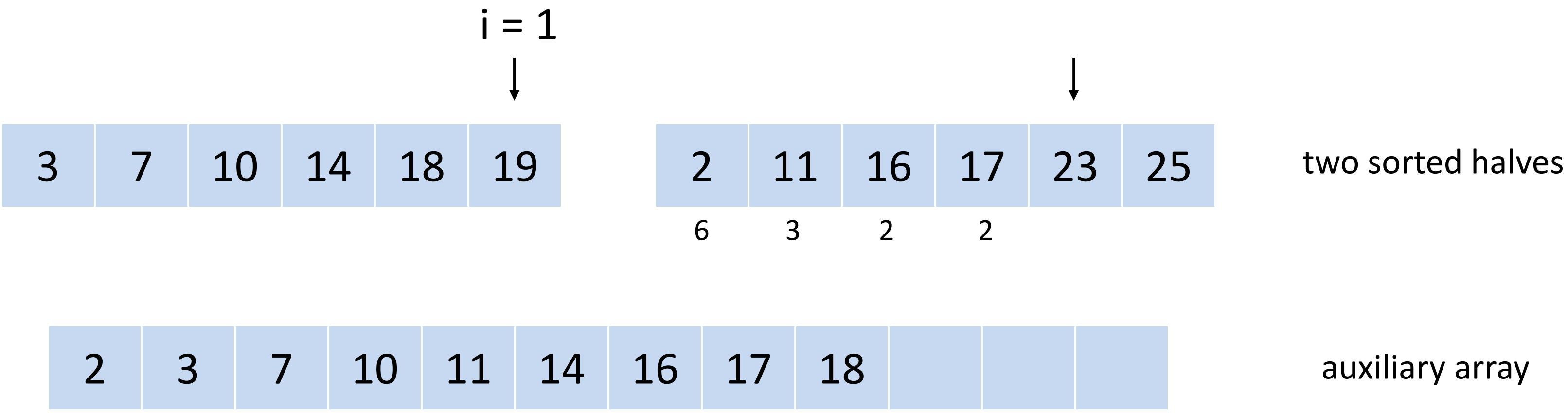
auxiliary array

Total: 6 + 3 + 2 + 2

Merge and Count

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.



3	7	10	14	18	19

2	11	16	17	23	25
6	3	2	2		

two sorted halves

2	3	7	10	11	14	16	17	18			

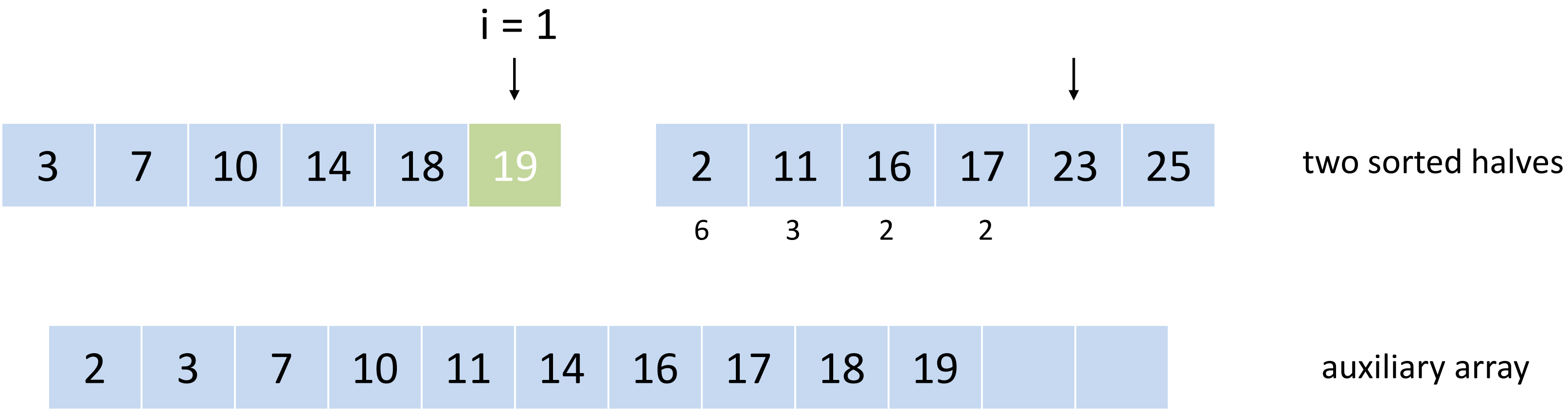
auxiliary array

Total: 6 + 3 + 2 + 2

Merge and Count

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.



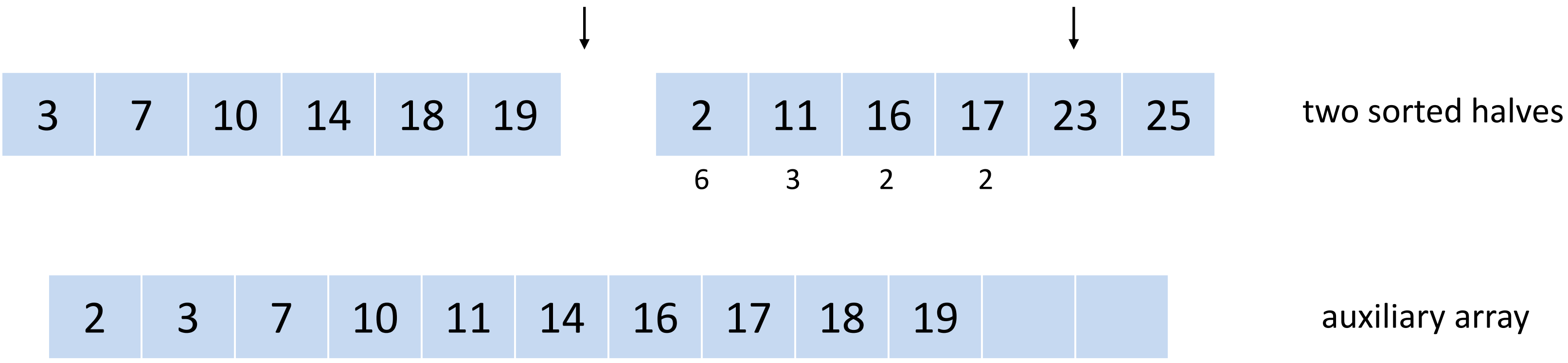
Total: 6 + 3 + 2 + 2

Merge and Count

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.

first half exhausted i = 0



3	7	10	14	18	19		2	11	16	17	23	25
							6	3	2	2		

two sorted halves

2	3	7	10	11	14	16	17	18	19		

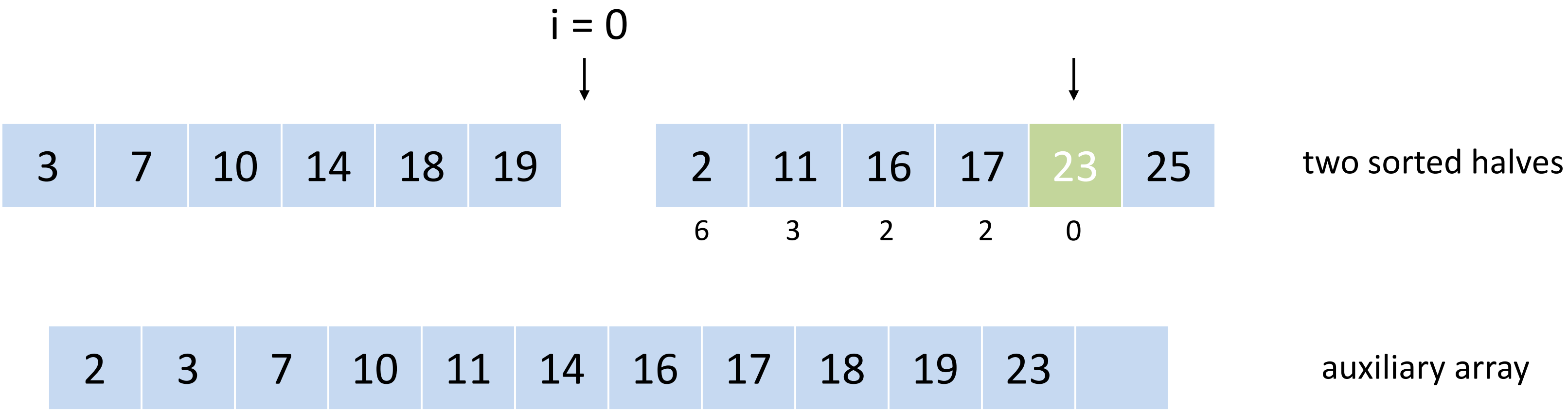
auxiliary array

Total: 6 + 3 + 2 + 2

Merge and Count

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 = 0

3	7	10	14	18	19		2	11	16	17	23	25
							6	3	2	2	0	

two sorted halves

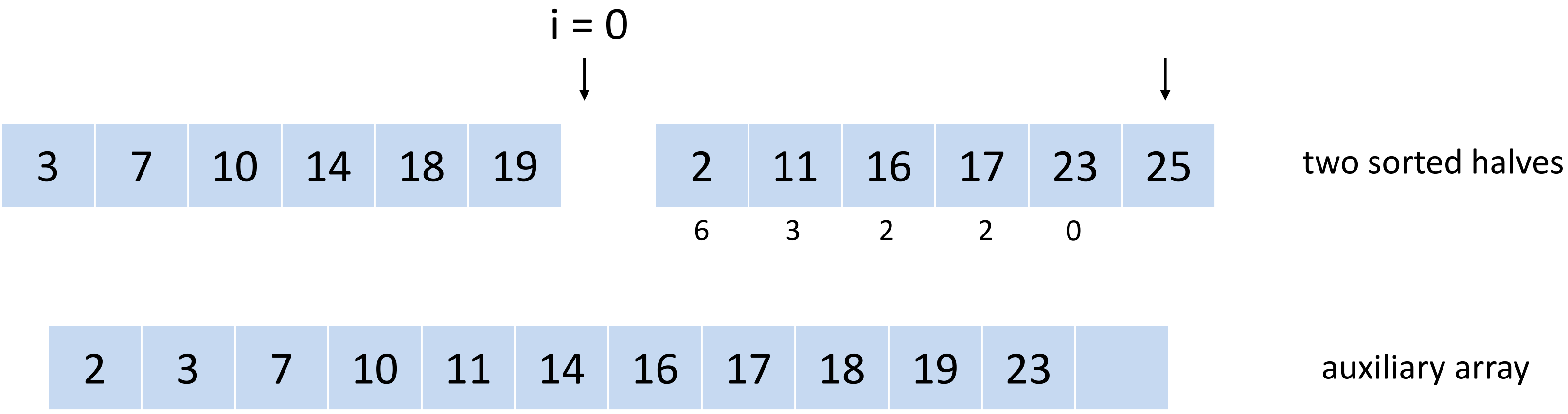
2	3	7	10	11	14	16	17	18	19	23	

auxiliary array

Total: 6 + 3 + 2 + 2 + 0

Merge and Count

- 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 = 0

3	7	10	14	18	19		2	11	16	17	23	25
							6	3	2	2	0	

two sorted halves

2	3	7	10	11	14	16	17	18	19	23	

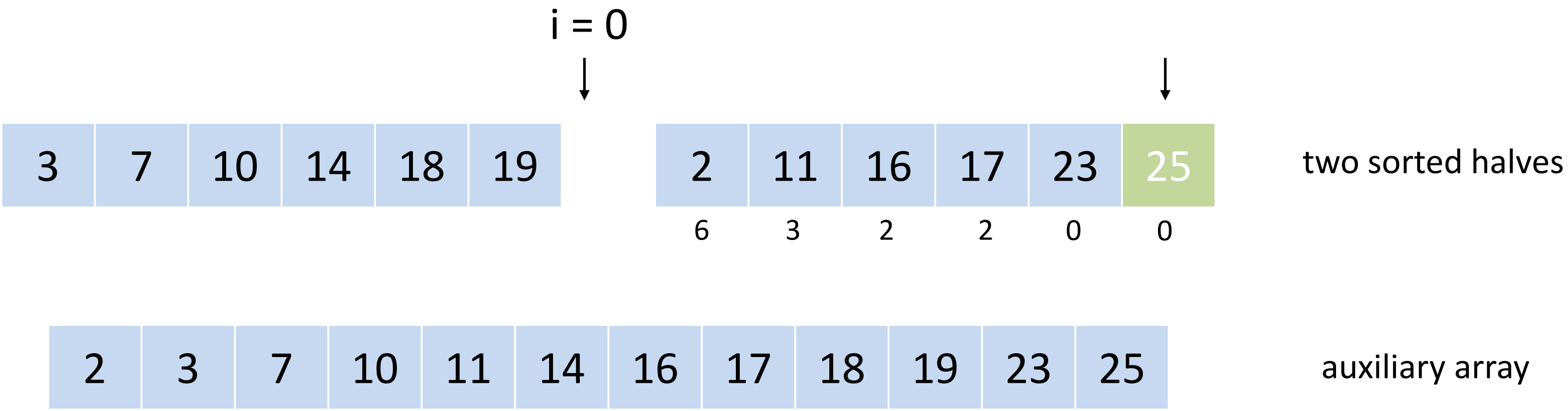
auxiliary array

Total: 6 + 3 + 2 + 2 + 0

Merge and Count

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 = 0

two sorted halves

3	7	10	14	18	19	2	11	16	17	23	25
						6	3	2	2	0	0

auxiliary array

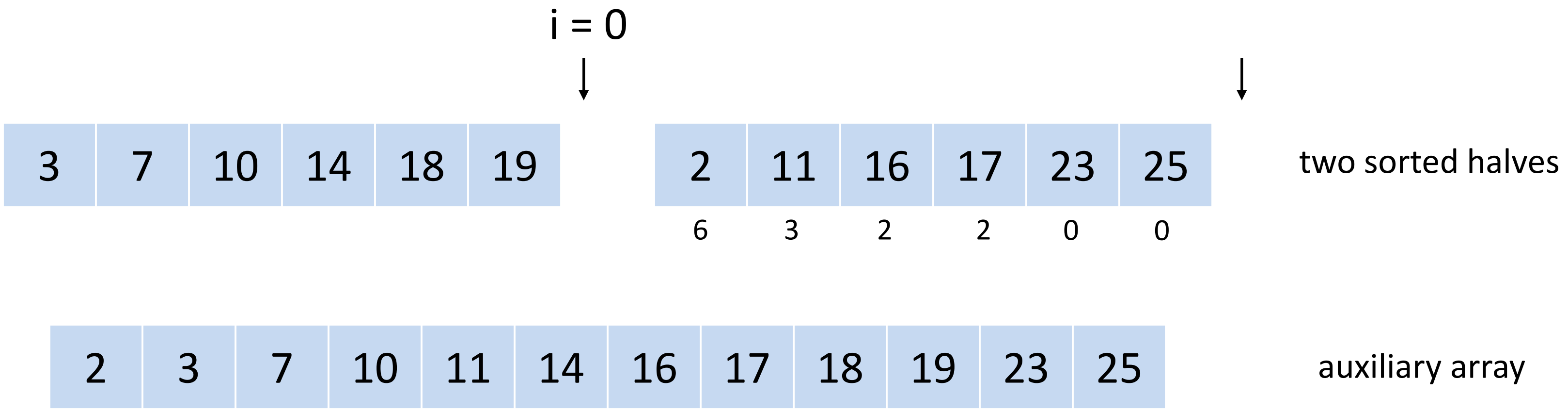
2	3	7	10	11	14	16	17	18	19	23	25

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

Merge and Count

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 = 0

3	7	10	14	18	19	2	11	16	17	23	25	two sorted halves
						6	3	2	2	0	0	

2	3	7	10	11	14	16	17	18	19	23	25	auxiliary array

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