COMP 355
Advanced Algorithms

Divide and Conquer: Inversion Counting
KT: 5.1-5.3

1

Divide-and-Conquer

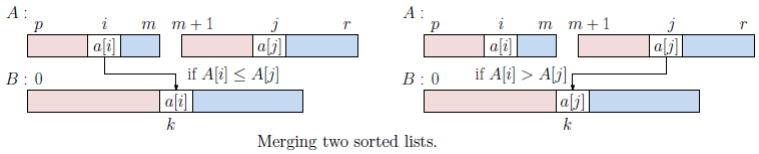
- Divide-and-conquer.
  - **Divide**: Break up problem into several parts.
  - **Conquer**: Solve each part recursively.
  - **Combine**: Merge solutions to sub-problems into overall solution.
- Most common usage.
  - Break up problem of size n into two equal parts of size ½n.
  - Solve two parts recursively.
  - Combine two solutions into overall solution in linear time.
- Consequence.
  - Brute force: n^2.
  - Divide-and-conquer: n log n.

2

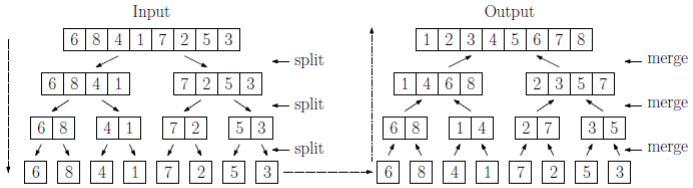
MergeSort

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.

3

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.

4

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.

Rhodes College

5

Divide-and-conquer solution

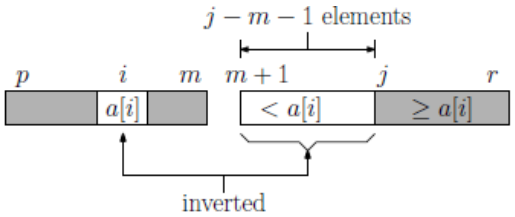
Basis case: If size$(A) = 1$, then there are no inversions.

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 $A[i] \leq A[j]$.

Rhodes College

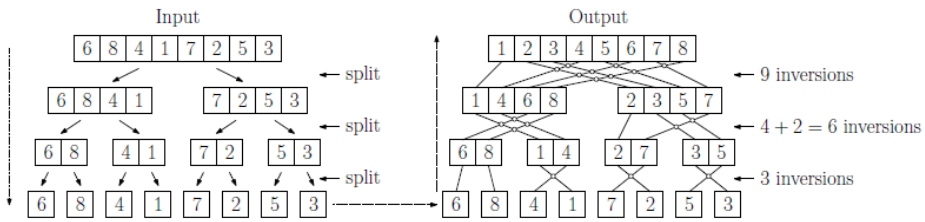
6

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
```

Rhodes College

Inversion Counting Example



Inversion counting by divide and conquer.

Rhodes College

Counting Inversions: Divide-and-Conquer

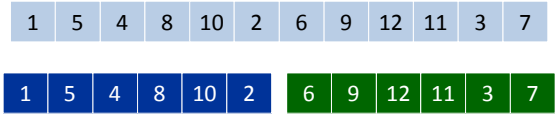
Divide-and-conquer.

9

Counting Inversions: Divide-and-Conquer

Divide-and-conquer.

- Divide: separate list into two pieces.



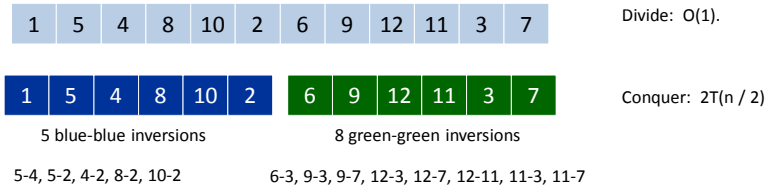
Divide: O(1).

10

Counting Inversions: Divide-and-Conquer

Divide-and-conquer.

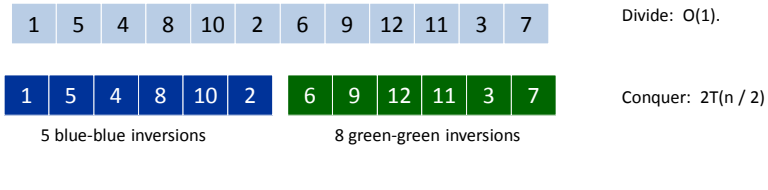
- Divide: separate list into two pieces.
- Conquer: recursively count inversions in each half.



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.



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

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

Count: O(n)

Merge: O(n)

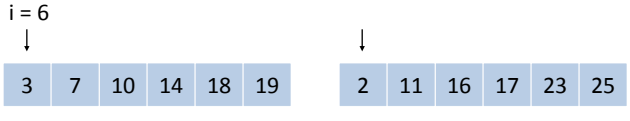
$$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \Rightarrow T(n) = O(n \log n)$$

13

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.



two sorted halves

auxiliary array

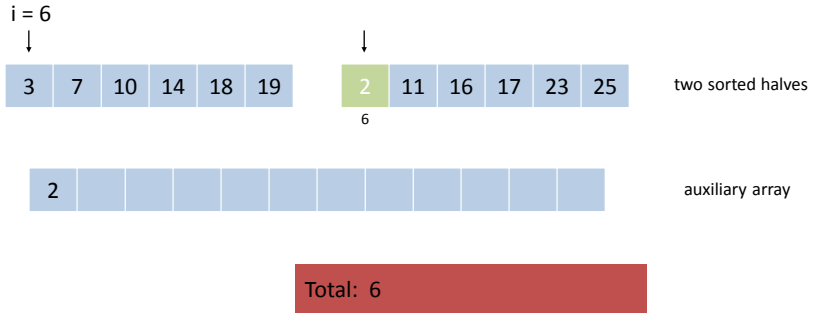
Total:

14

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.

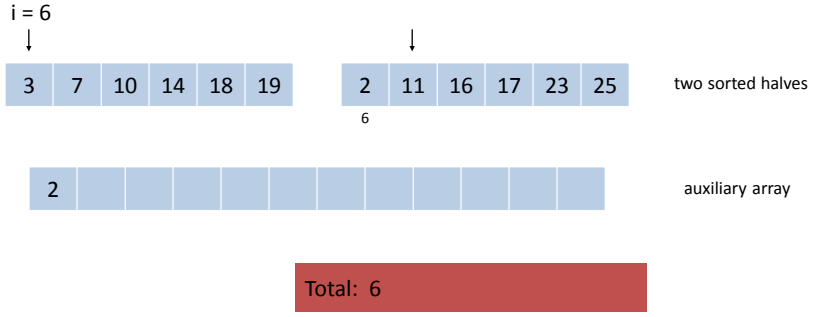


15

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.

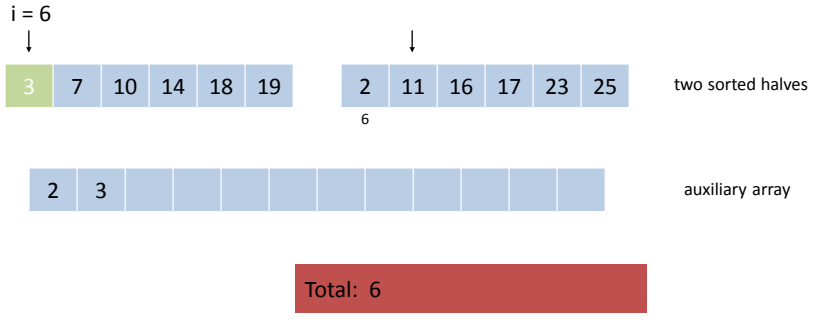


16

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.

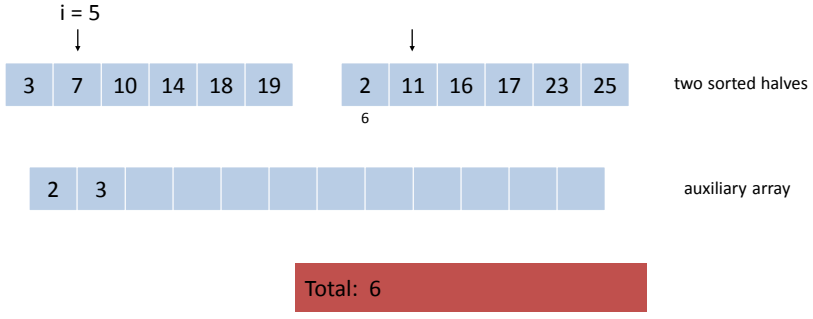


17

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.

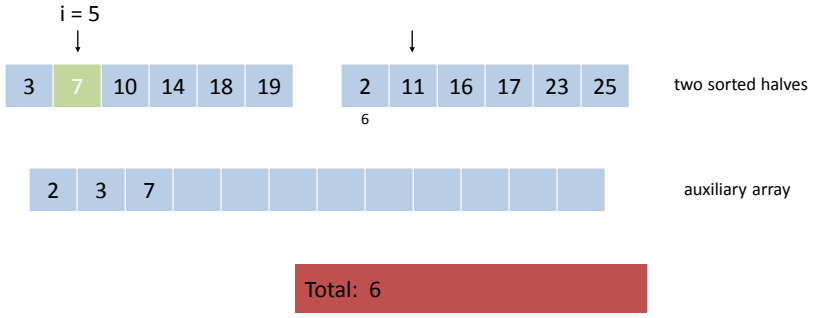


18

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.

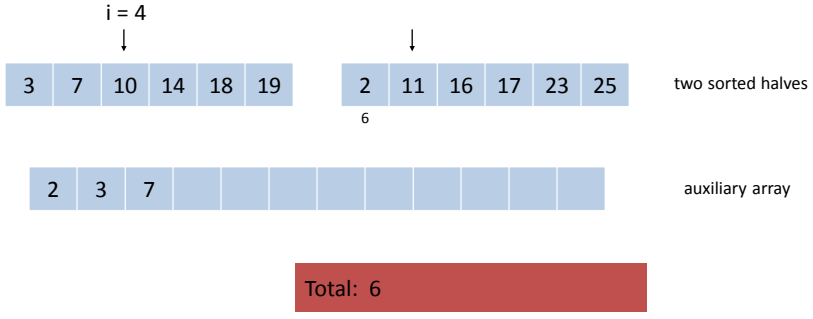


19

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.

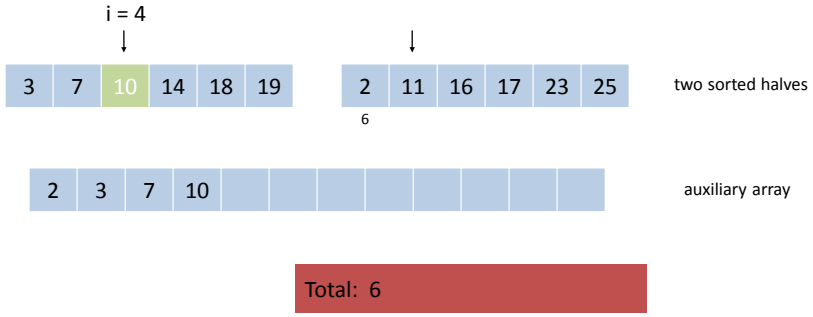


20

Merge and Count

Merge and count step.

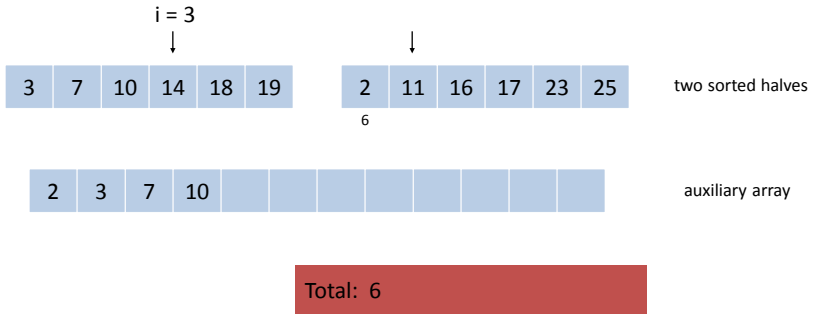
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

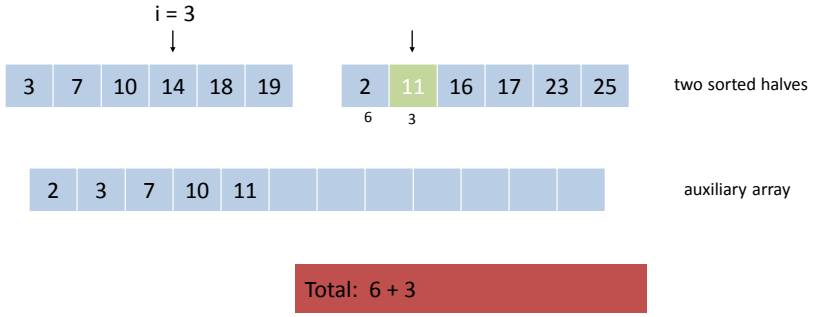
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

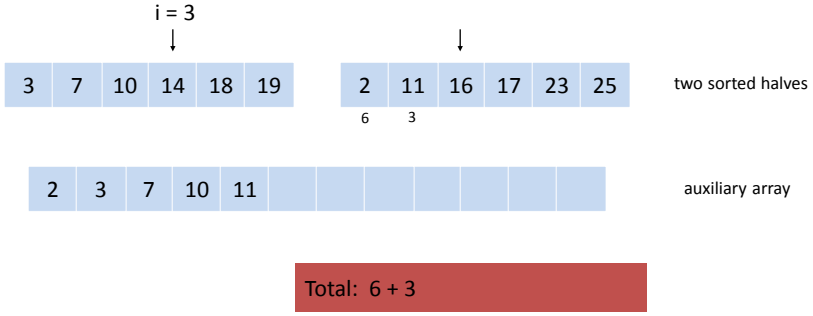
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

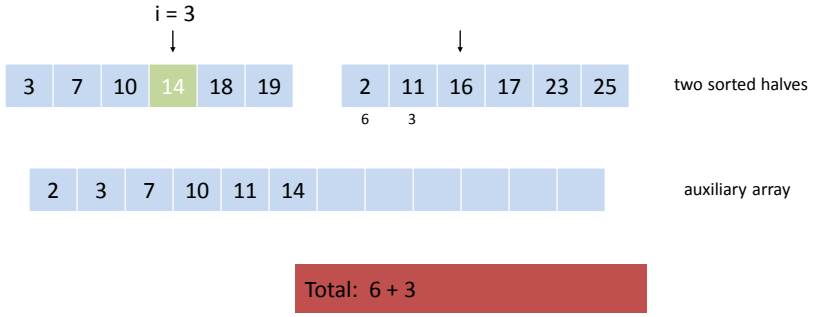
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



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.

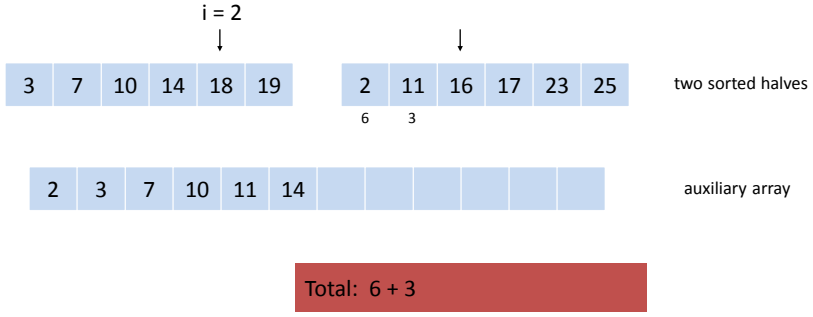


25

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.

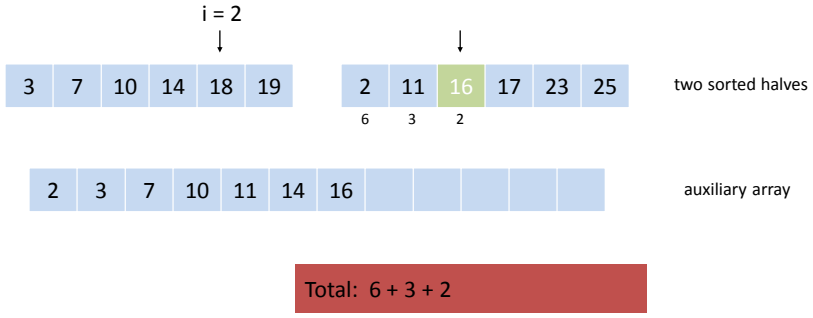


26

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.

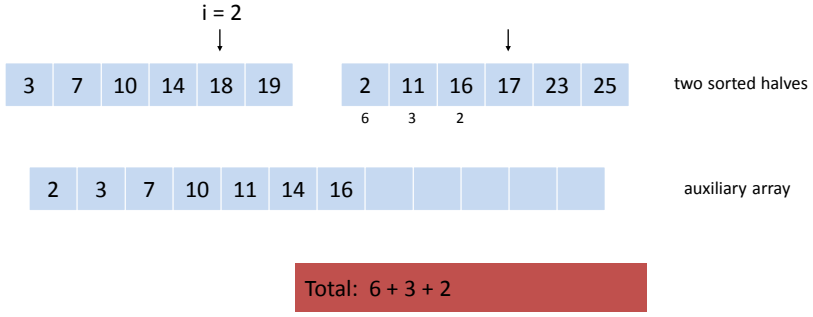


27

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.

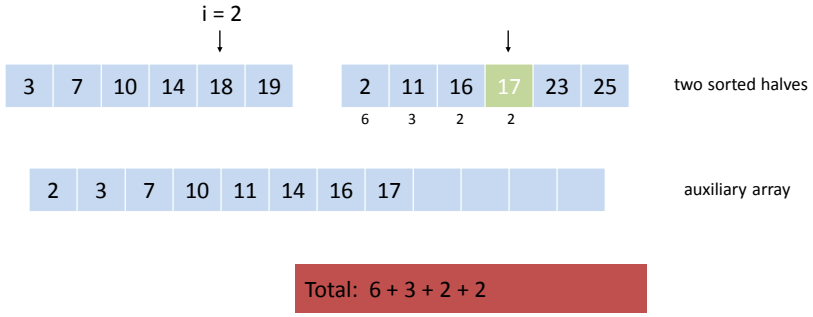


28

Merge and Count

Merge and count step.

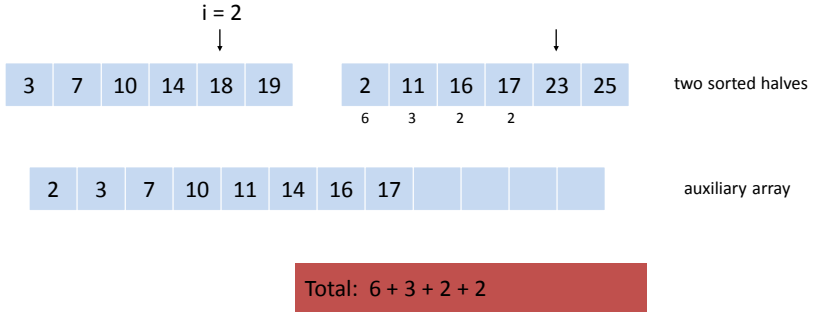
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

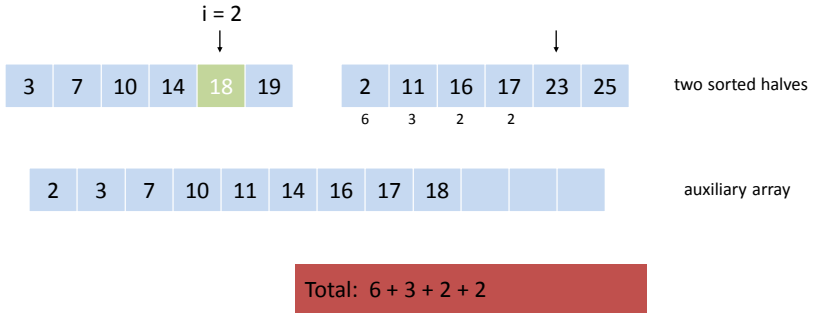
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

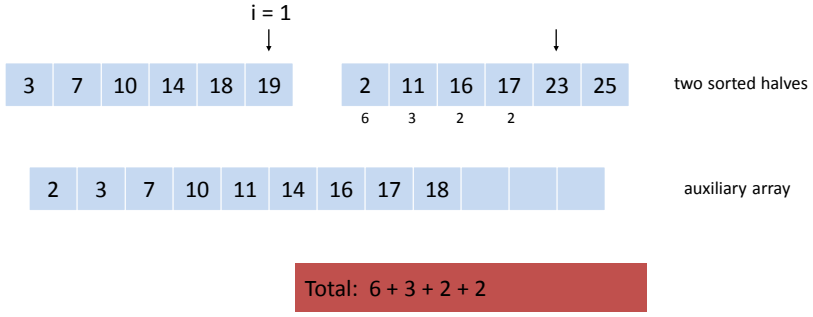
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

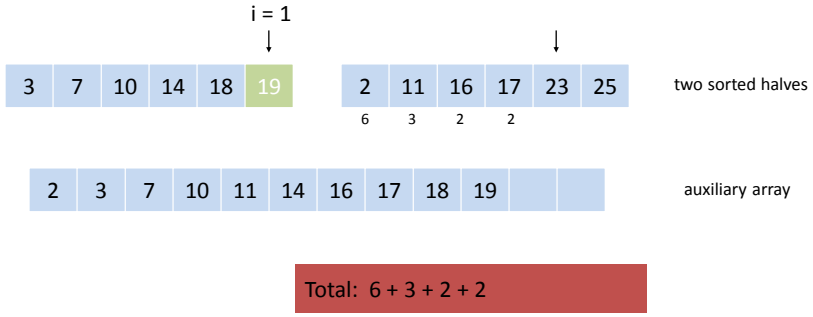
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

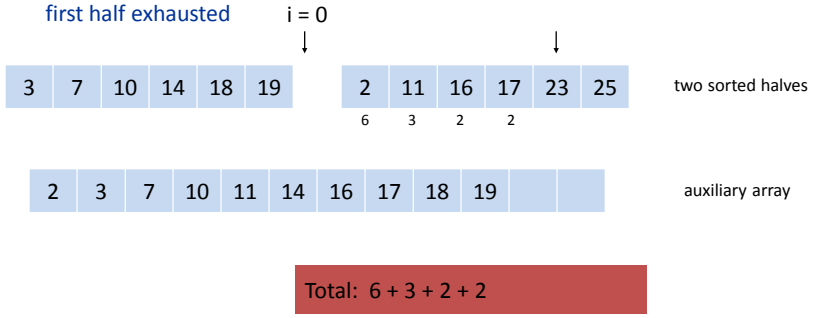
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.



Merge and Count

Merge and count step.

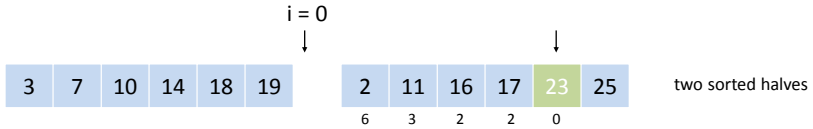
- Given two sorted halves, count number of inversions where a_i and a_j are in different halves.
- Combine two sorted halves into sorted whole.

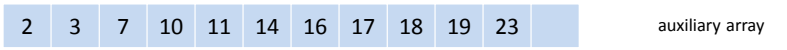


Merge and Count

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.



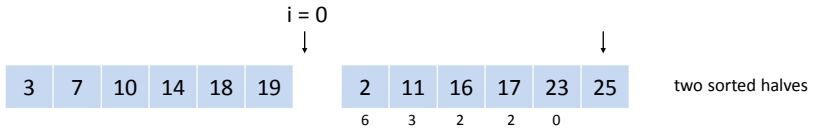


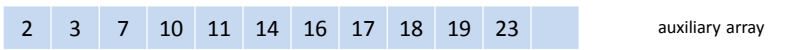
Total: 6 + 3 + 2 + 2 + 0

35

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.





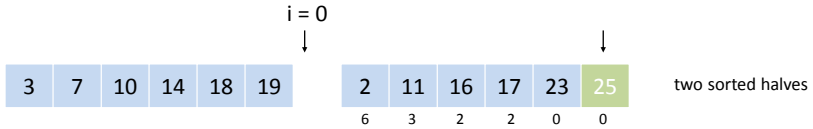
Total: 6 + 3 + 2 + 2 + 0

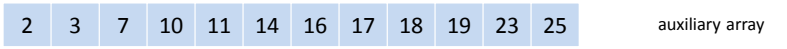
36

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.



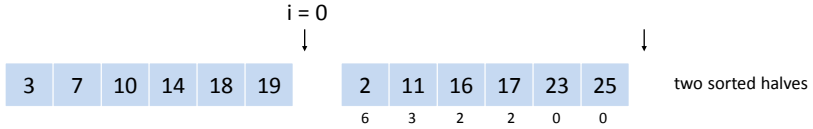


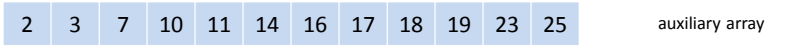
Total: 6 + 3 + 2 + 2 + 0 + 0

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.





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

Next Time

Divide-and-Conquer: Closest Pair