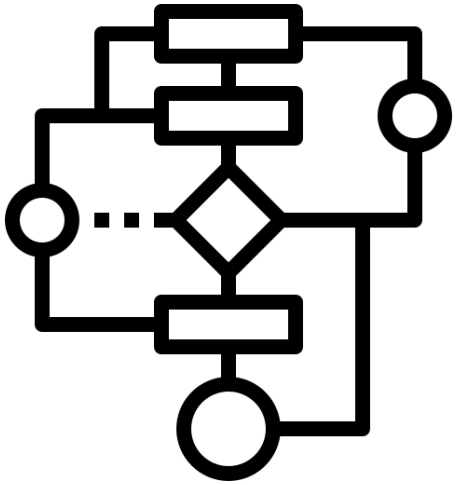
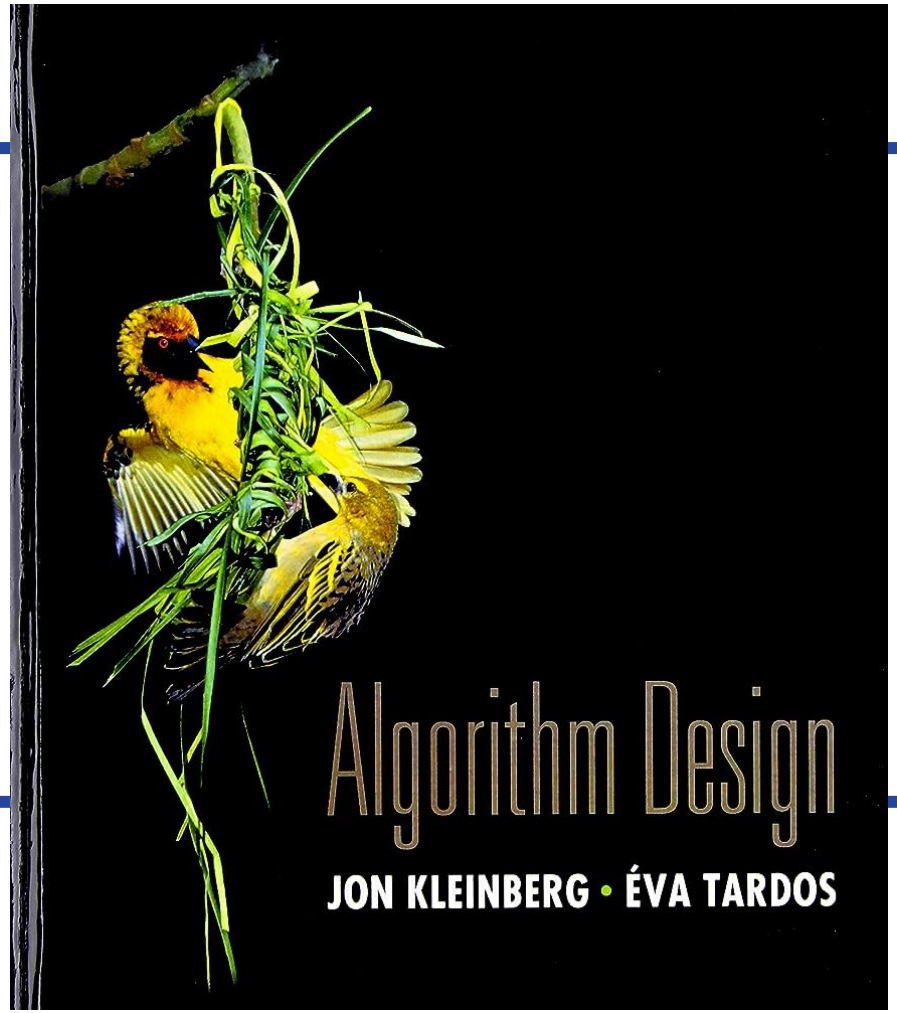


CS 310: Algorithms

Lecture 11

Instructor: Naveed Anwar Bhatti

Slides by Kevin Wayne (**heavily modified by Naveed Bhatti**).
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

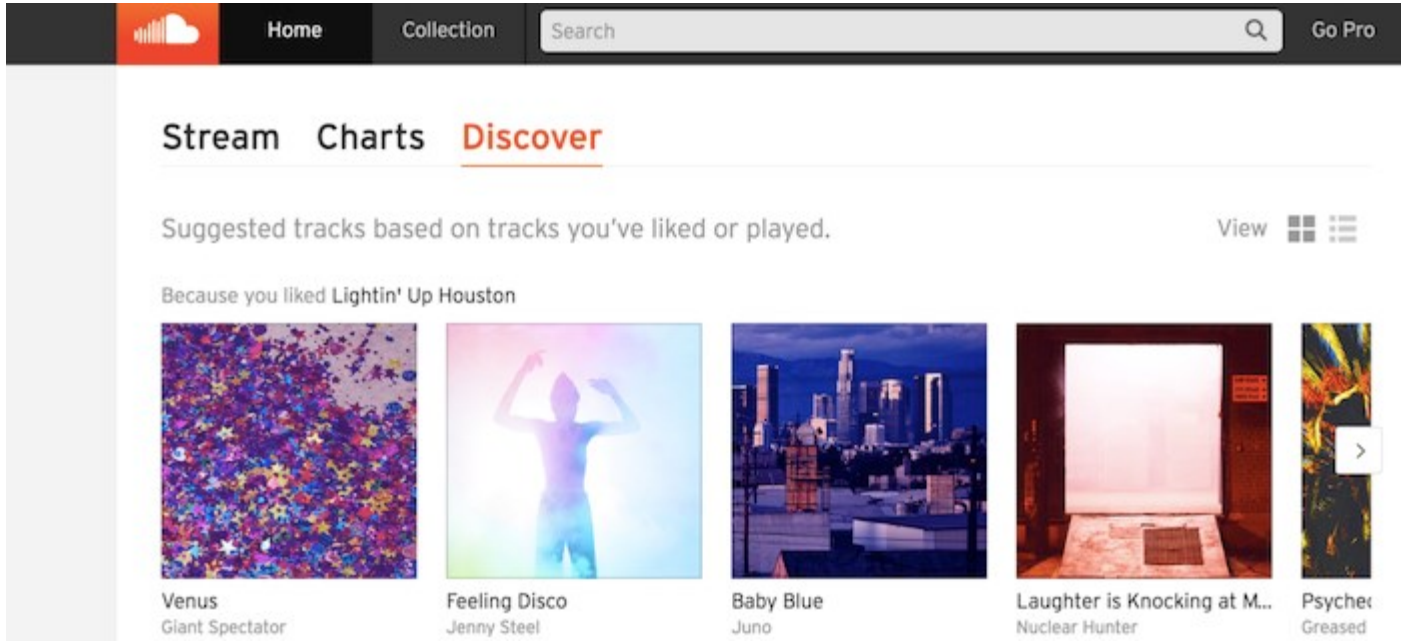


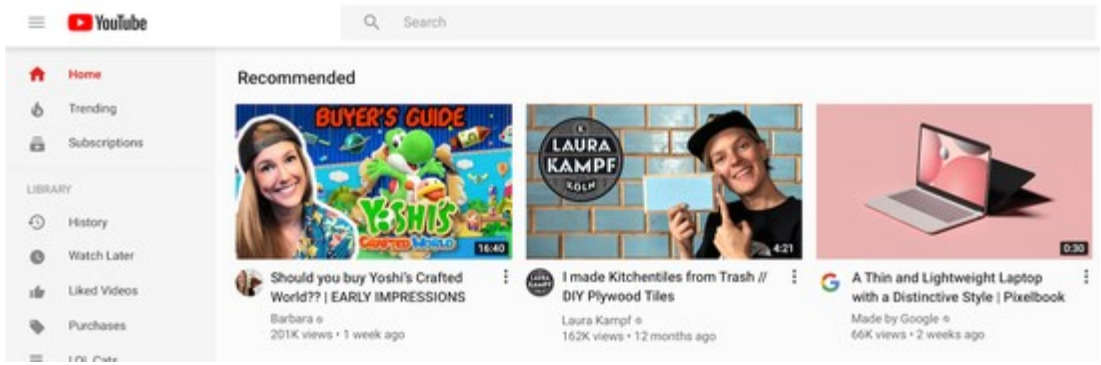
Chapter 5:
Divide and Conquer

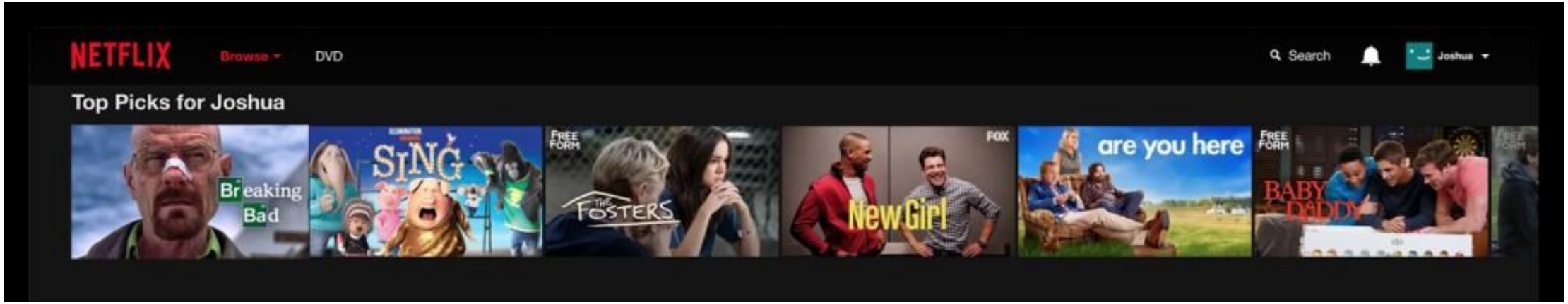
Section 5.3:
Counting Inversions

Slides by Kevin Wayne (**heavily modified by Naveed Bhatti**).
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

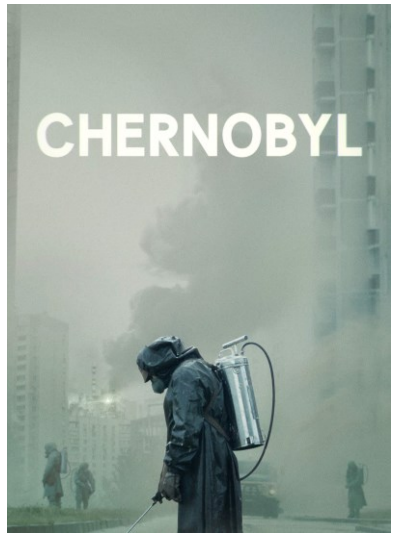
Motivation: Recommendation Systems



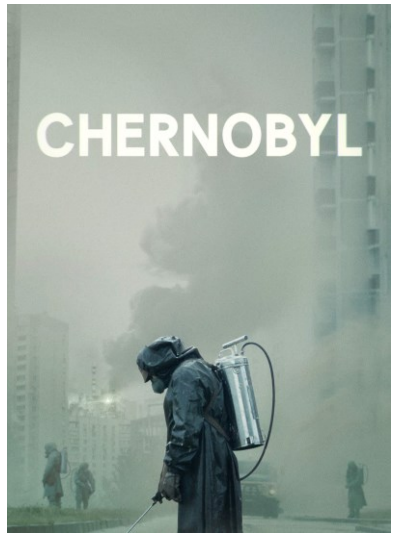




Motivation: Recommendation Systems

1	2	3	4	5
				
2	4	1	3	5

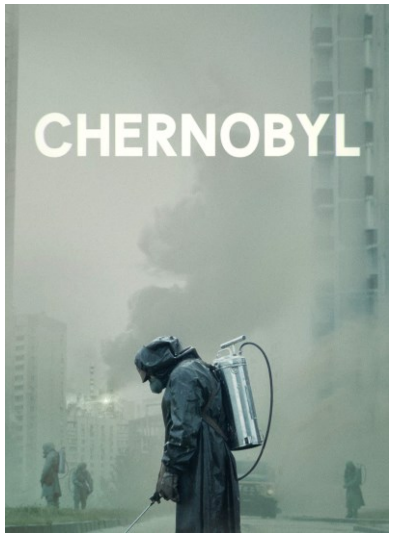
Counting inversions

1	2	3	4	5
				
2	4	1	3	5

(2,1)

Counting inversions

1 2 3 4 5

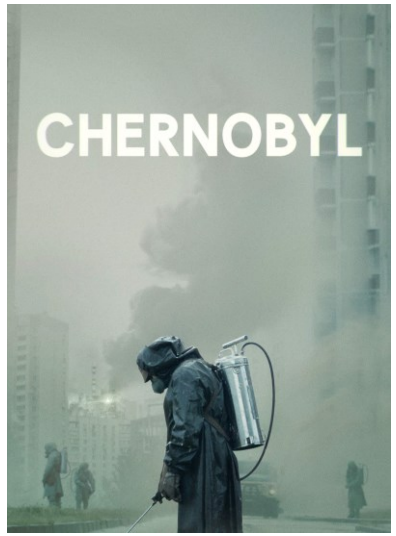


2 4 1 3 5

(2,1) (4,1)

Counting inversions

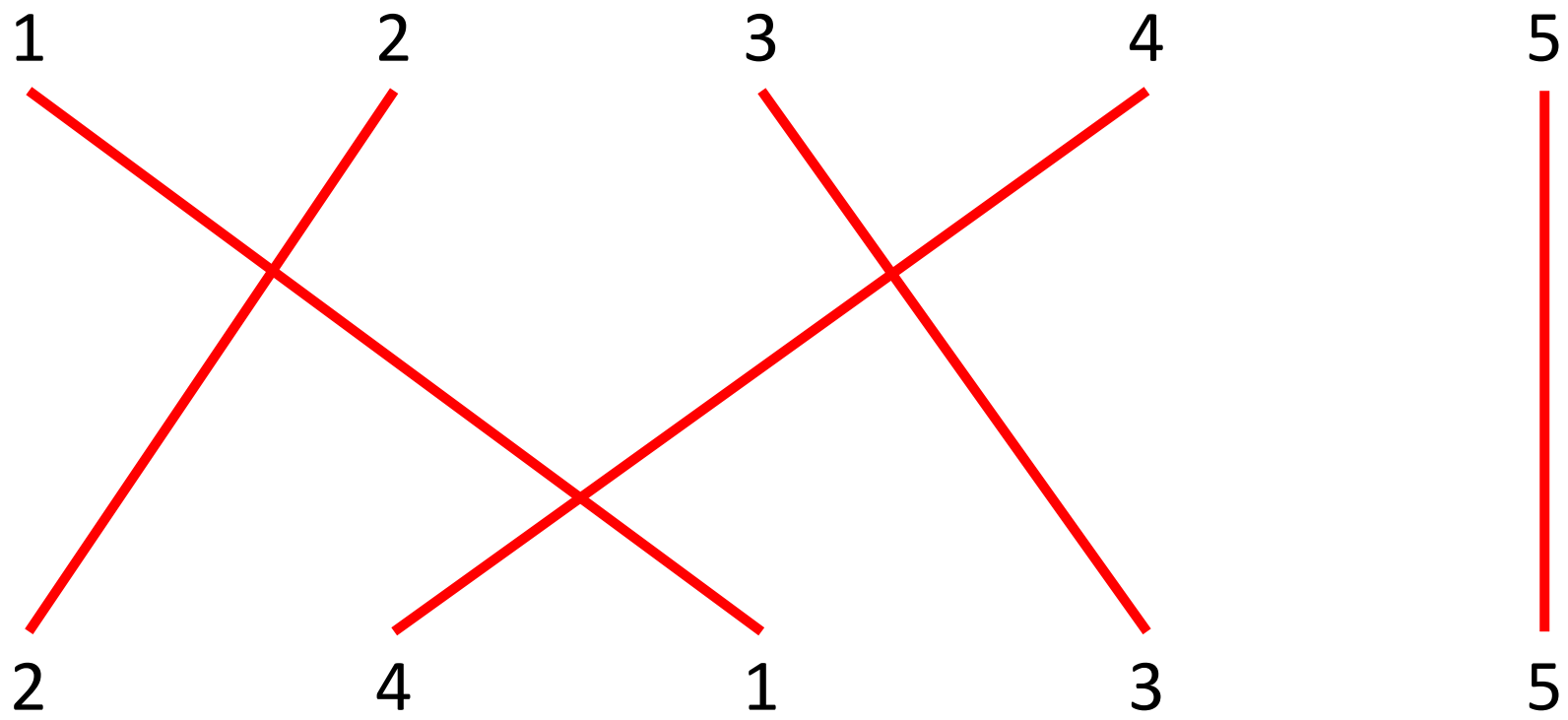
1 2 3 4 5



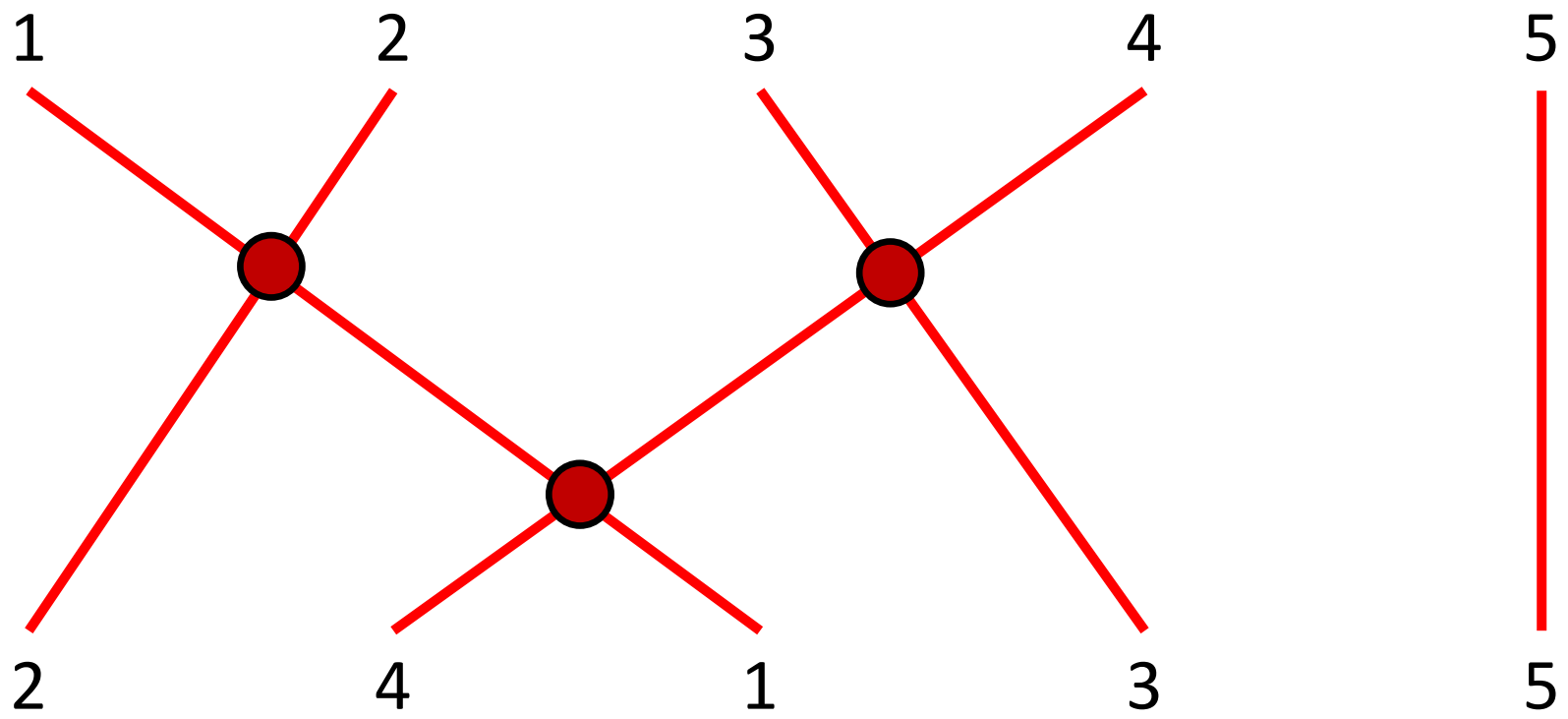
2 4 1 3 5

(2,1) (4,1) (4,3)

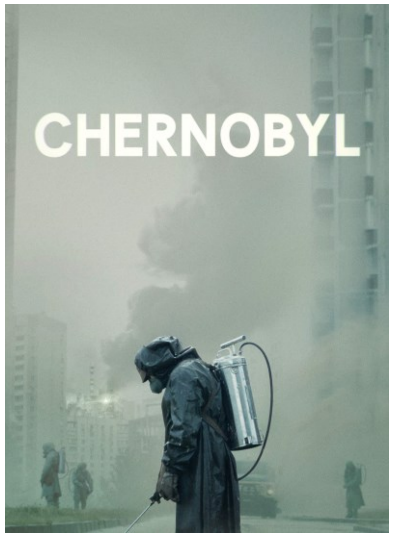
Counting inversions



Counting inversions



Motivation: Recommendation Systems

1	2	3	4	5
				
2	4	1	3	5

Counting inversions: Divide and Conquer (Warm Up!)

- Divide: separate list into two halves A and B.
- Conquer: recursively count inversions in each list.
- Combine: count inversions (a, b) with $a \in A$ and $b \in B$.
- Return sum of three counts.

3	7	11	5	9	1	6	13	4	2	15	8	10	14	12	16

(3,1) (7,5) (7,1) (7,6) (11,5) (11,9) (11,1) (11,6) (5,1) (9,1) (9,6) = **11**

3	7	11	5	9	1	6	13

(4,2) (15,8) (15,10) (15,14) (15,12) (14,12) = **6**

4	2	15	8	10	14	12	16

(3,2) (7,4) (7,2) (11,4) (11,2) (11,8) (11,10) (5,4) (5,2) (9,4) (9,2) (9,8) (6,4) (6,2) (13,4) (13,2) (13,8) (13,10) (13,12) = **19**

3	7	11	5	9	1	6	13

4	2	15	8	10	14	12	16

Return = 11 + 6 +19 = 36

Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B** ***efficiently***?

A. Easy if A and B are sorted!

3	7	11	5	9	1	6	13

4	2	15	8	10	14	12	16

Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.

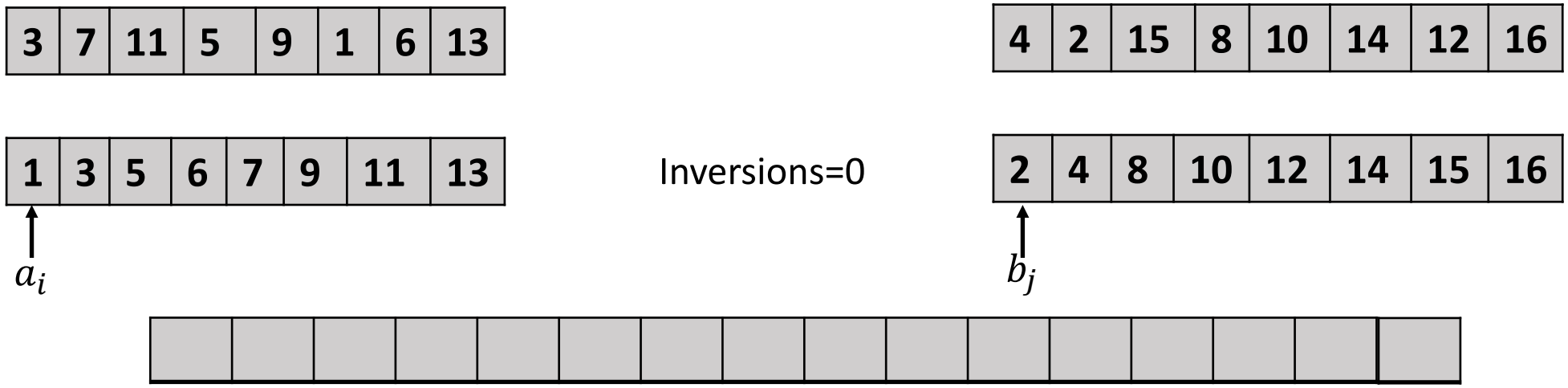
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



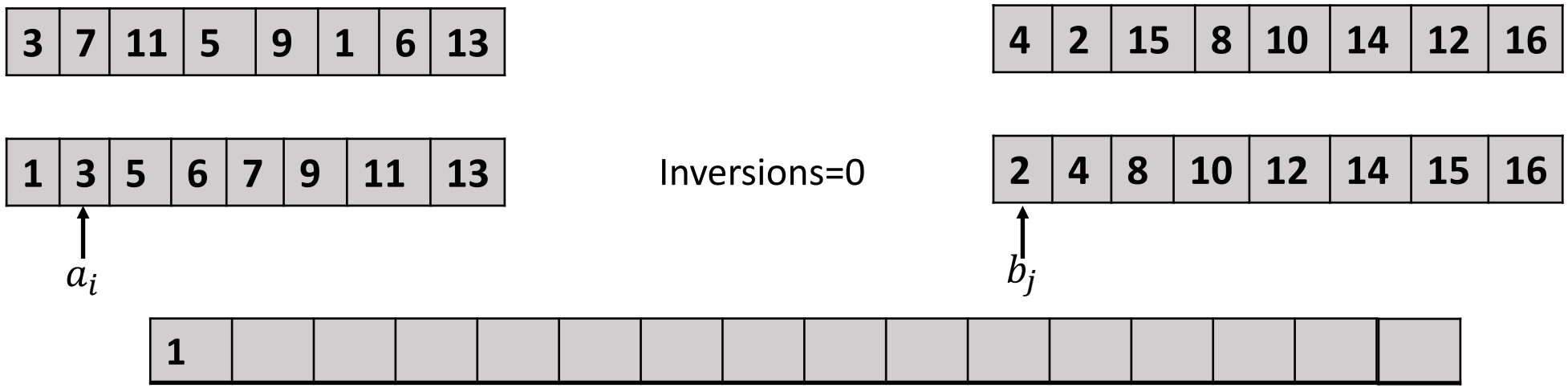
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



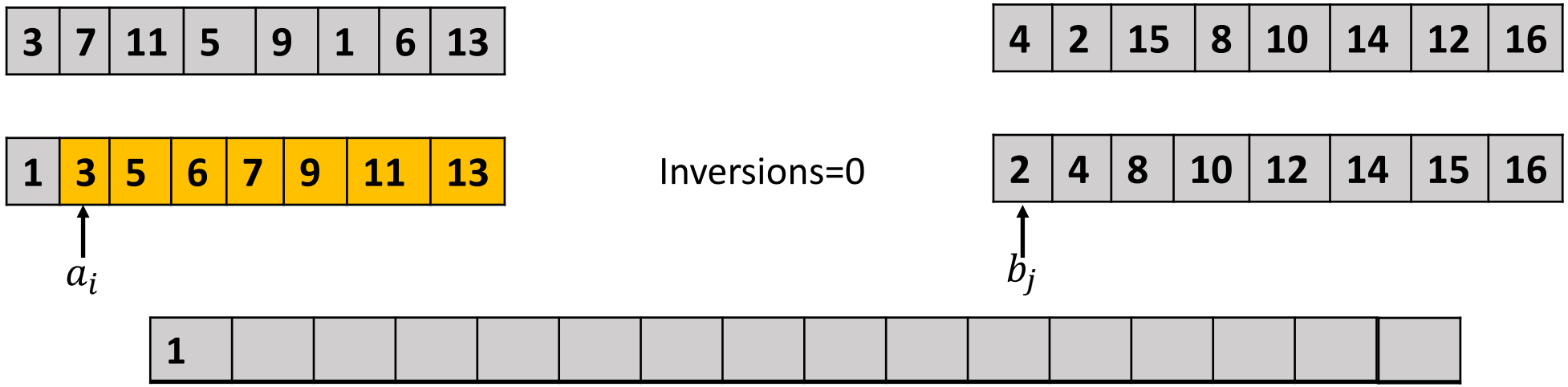
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



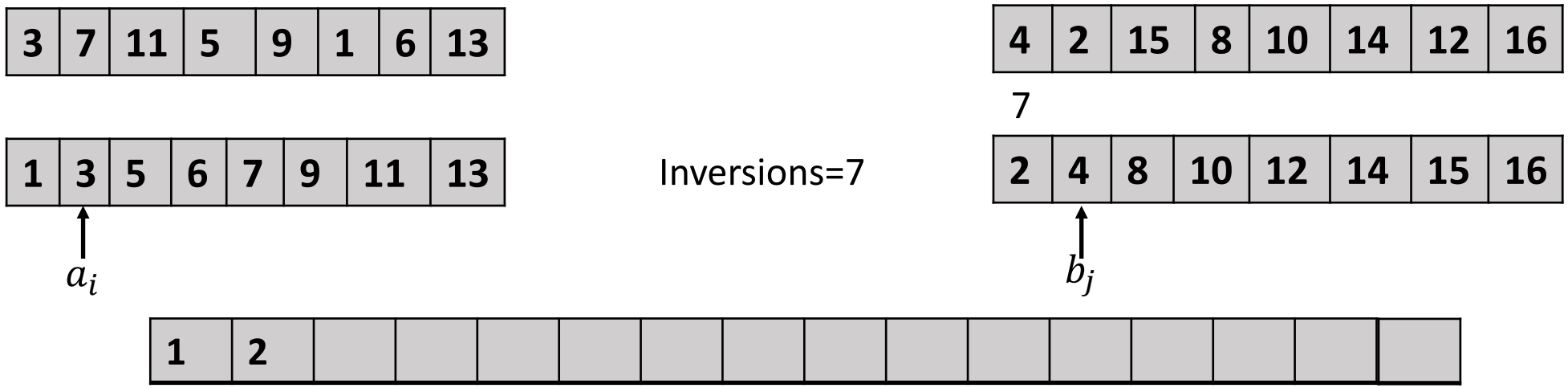
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



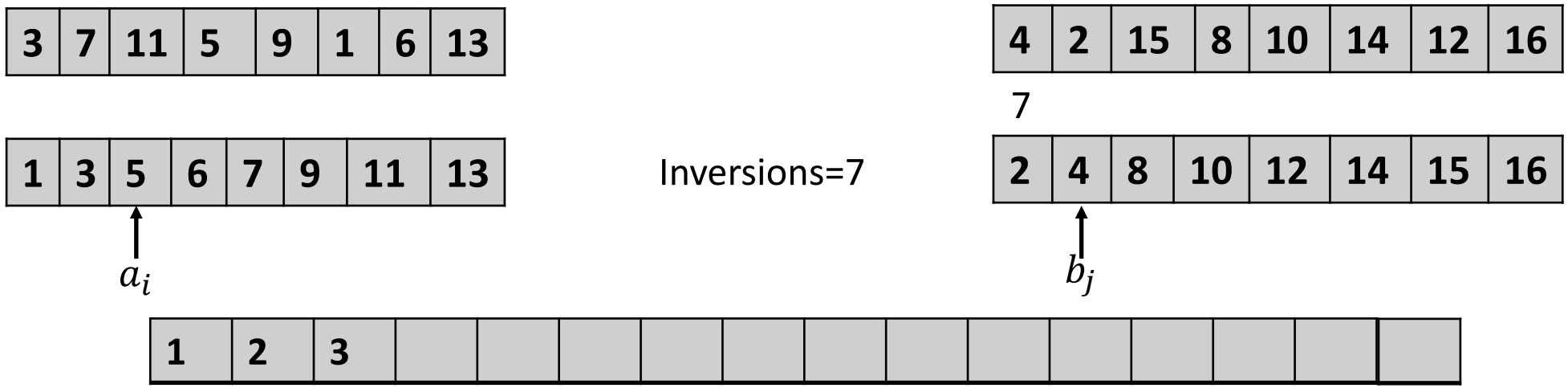
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



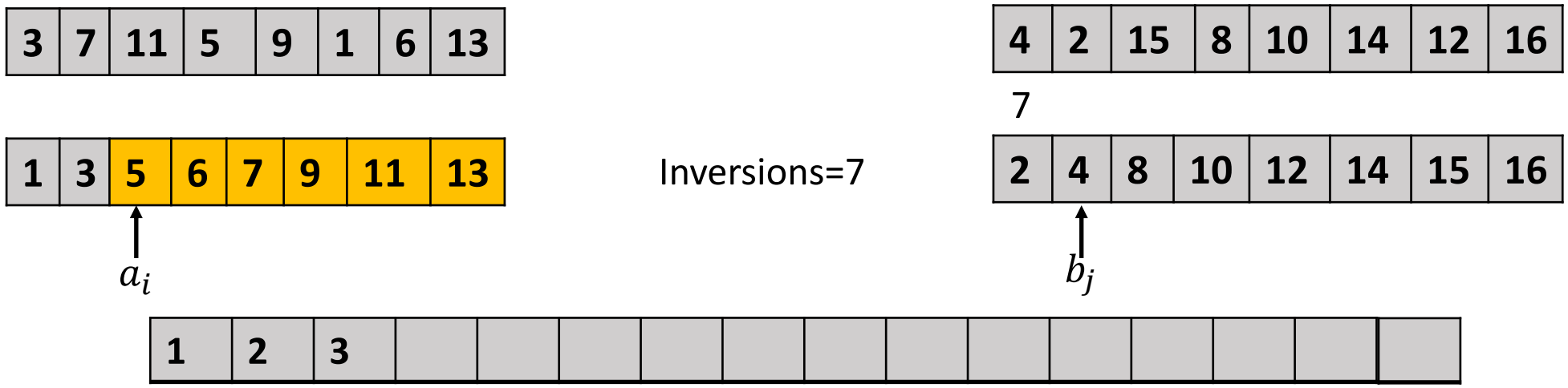
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



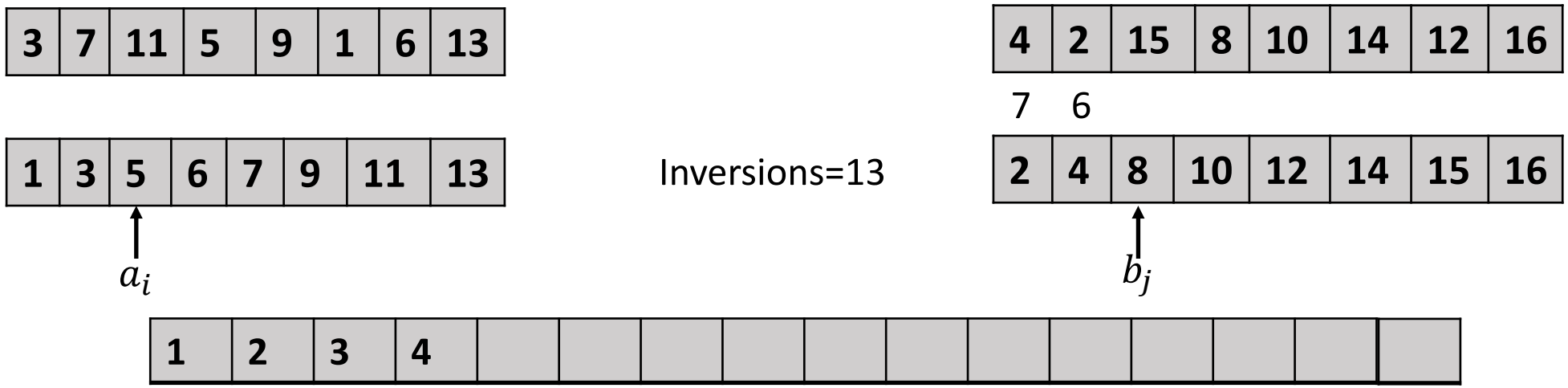
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



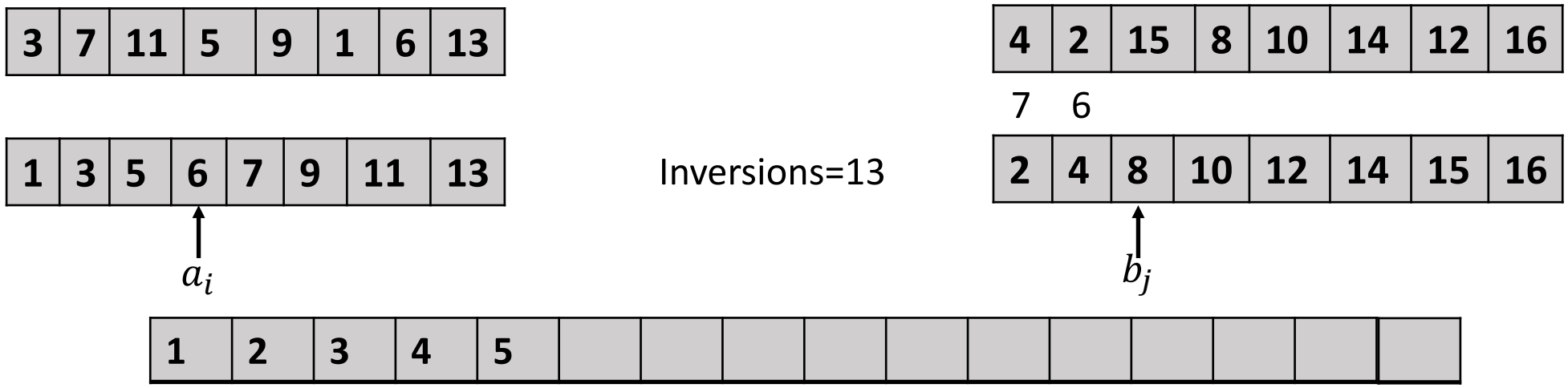
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



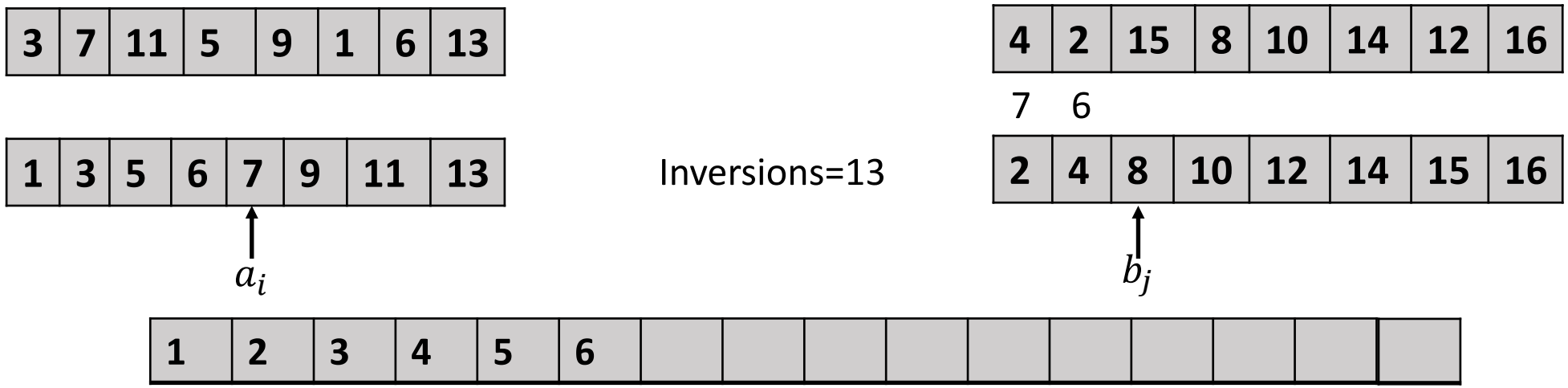
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



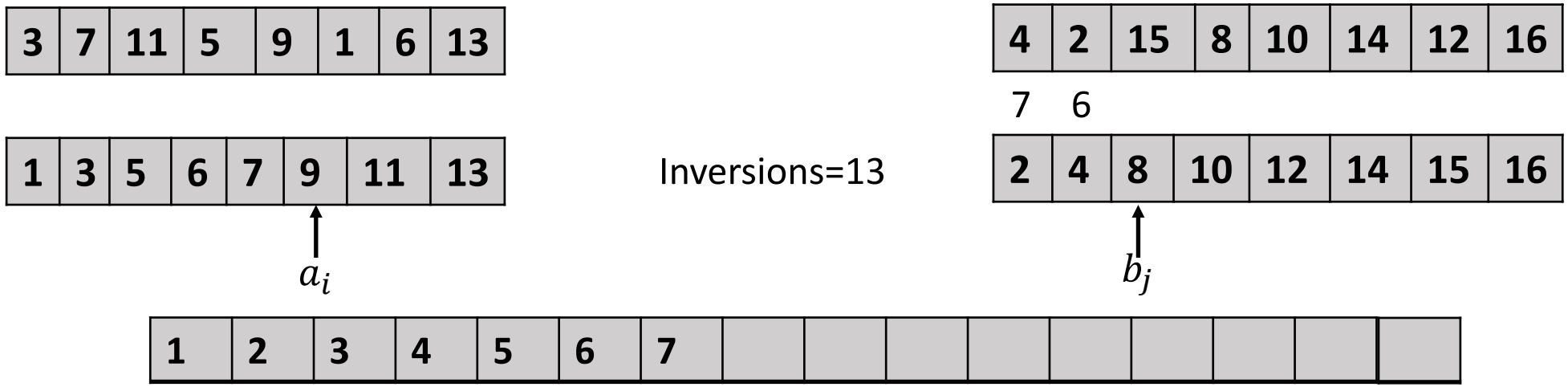
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



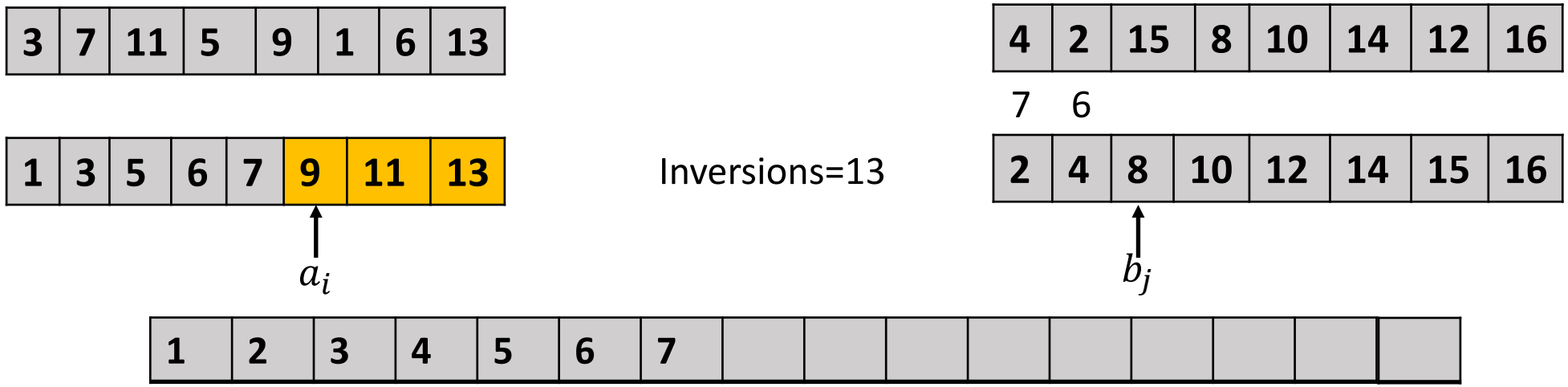
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



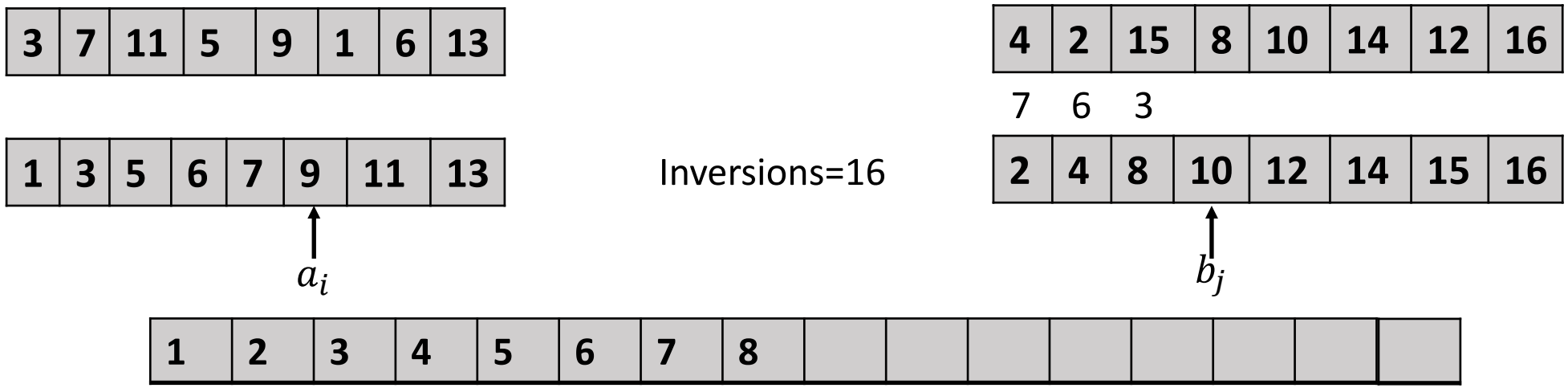
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



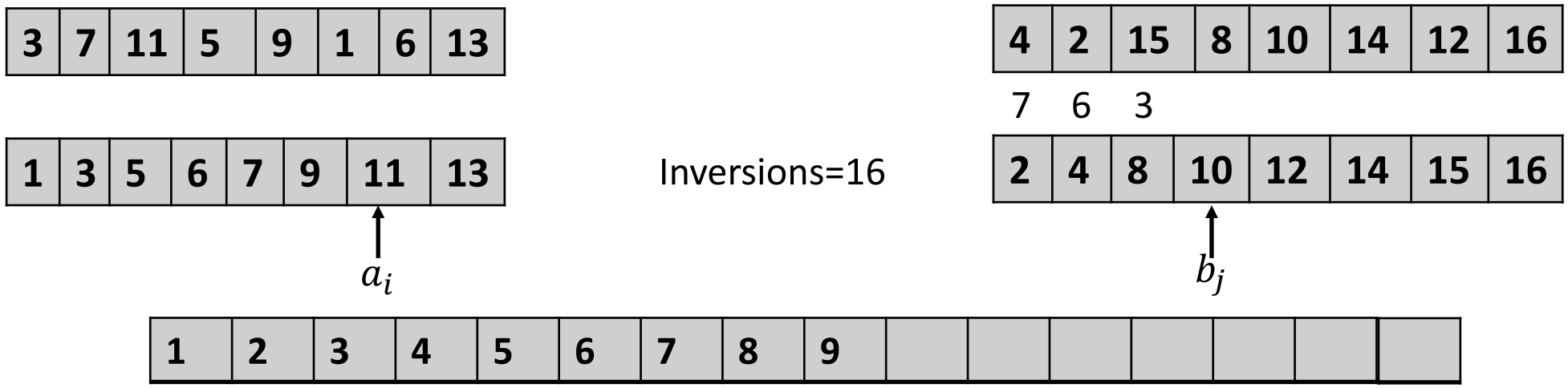
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



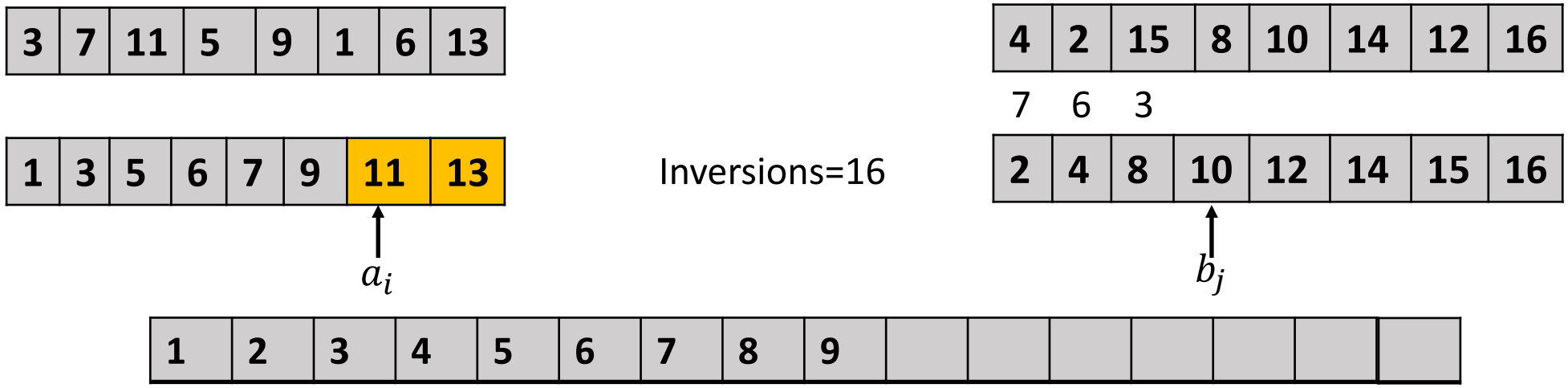
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



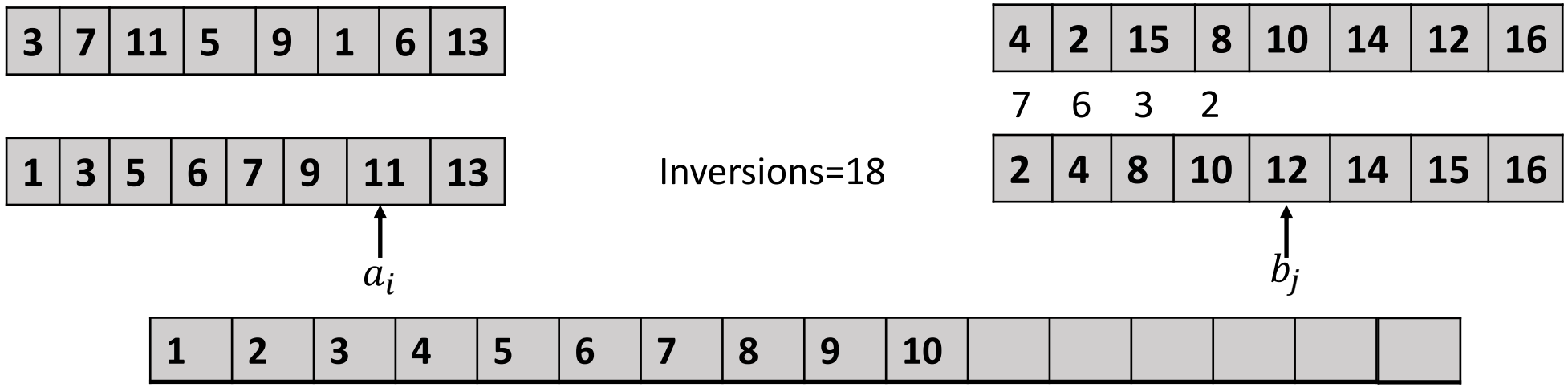
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



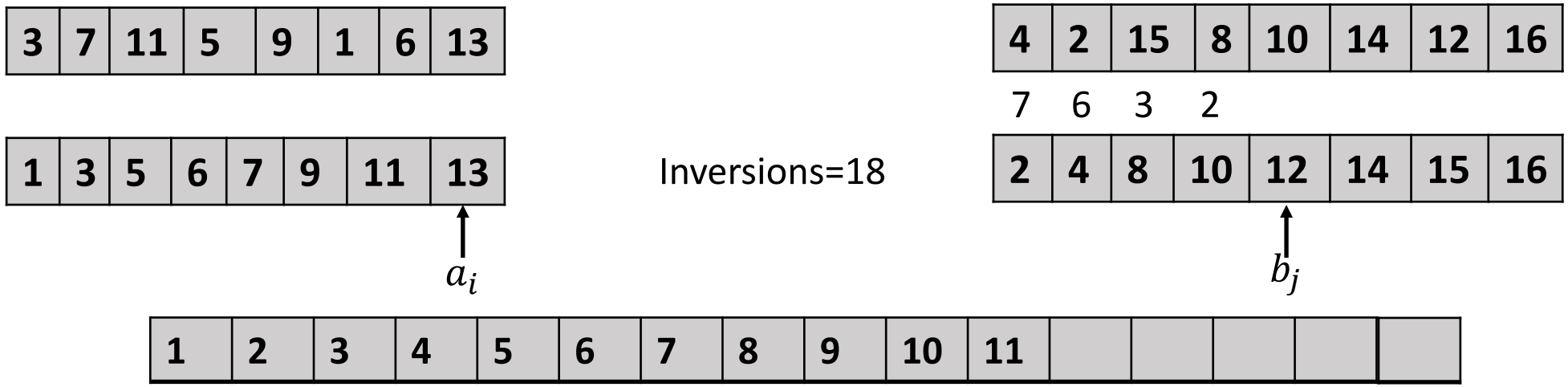
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



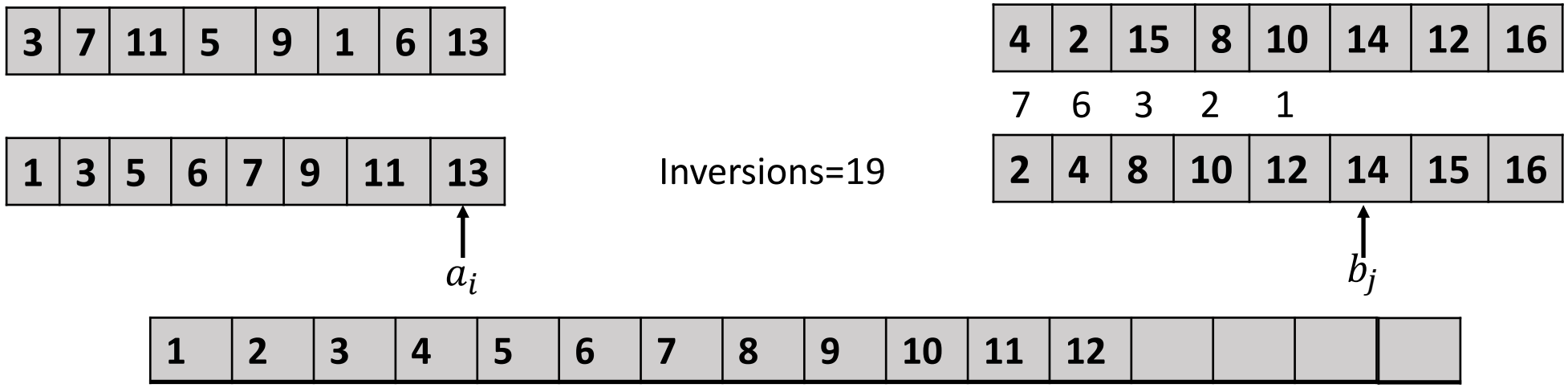
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



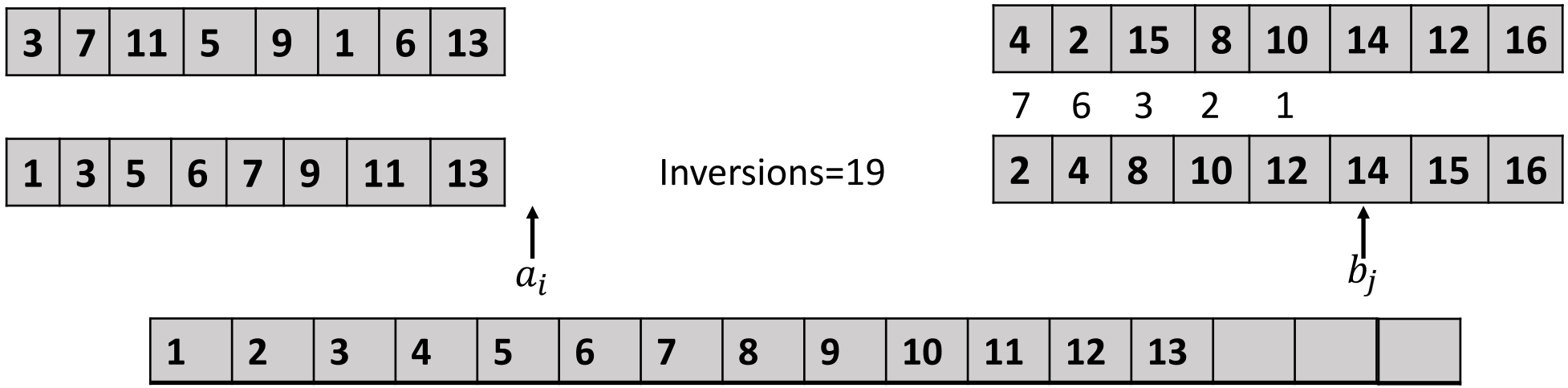
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



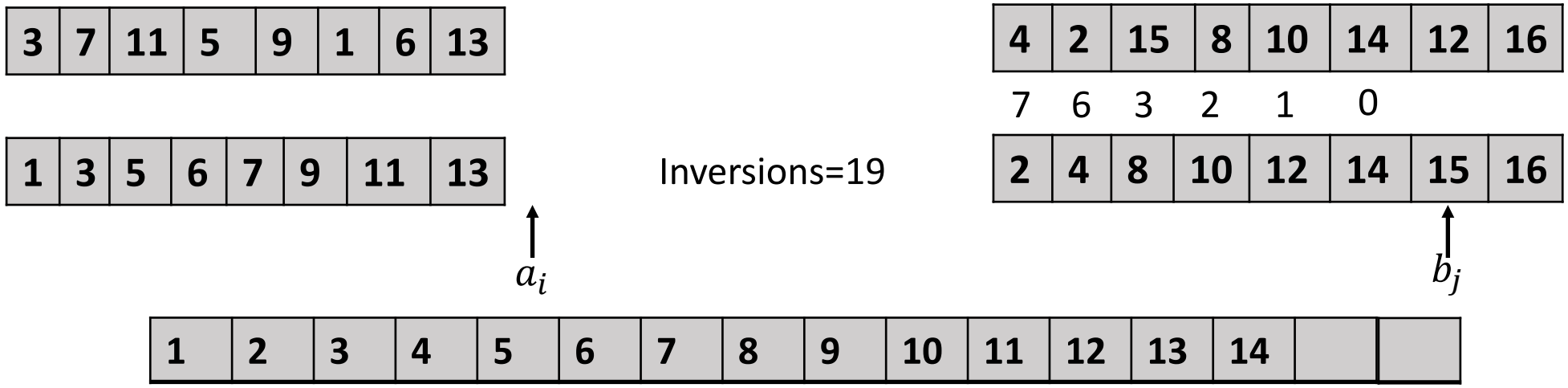
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



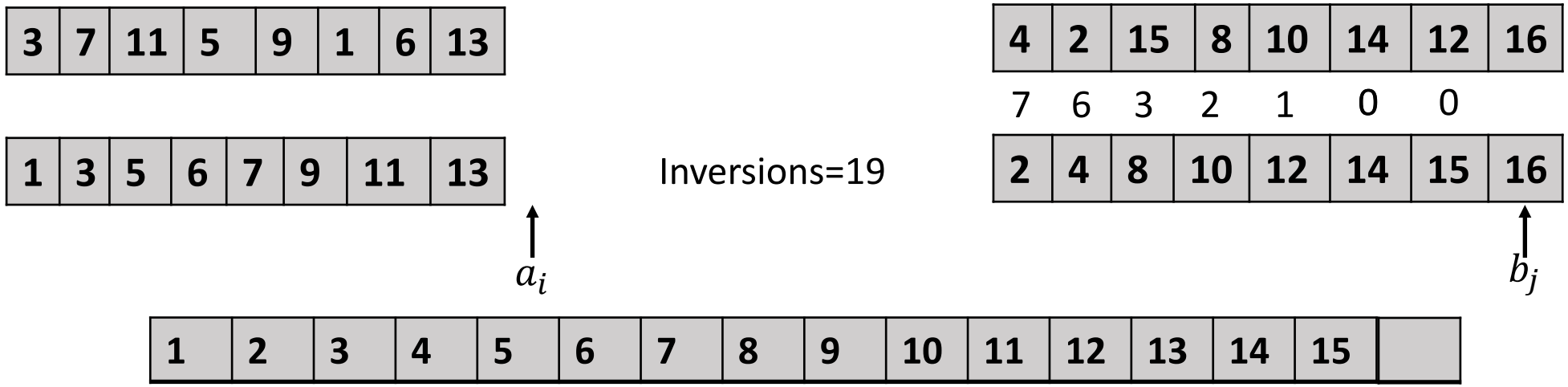
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



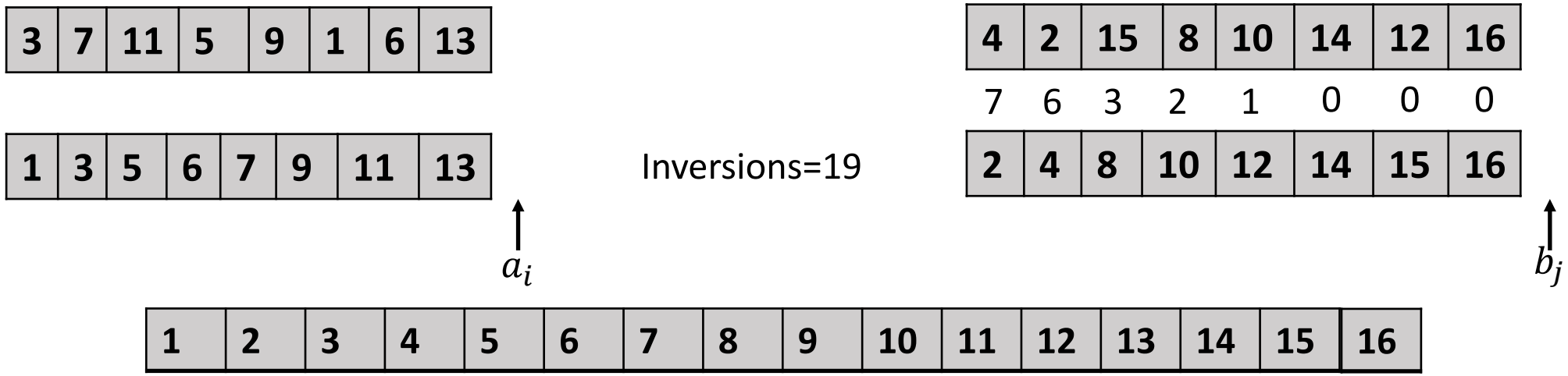
Counting inversions: how to combine two subproblems?

Q. How to count inversions (a, b) with **a ∈ A** and **b ∈ B**?

A. Easy if A and B are sorted!

Algorithm:

- Scan *A* and *B* from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in *B*.
- If $a_i > b_j$, then b_j is inverted with every element left in *A*.
- Append smaller element to sorted list *C*.



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$

2 $x \leftarrow length(A_1)$

3 $inversions \leftarrow 0$

4 $i \leftarrow 1; j \leftarrow 1$

5 **for** $k = 1$ **to** m

6 **if** $A_1[i] \leq A_2[j]$

7 $A[k] \leftarrow A_1[i]$

8 $i \leftarrow i + 1$

9 $x \leftarrow x - 1$

10 **else** $A[k] \leftarrow A_2[j]$

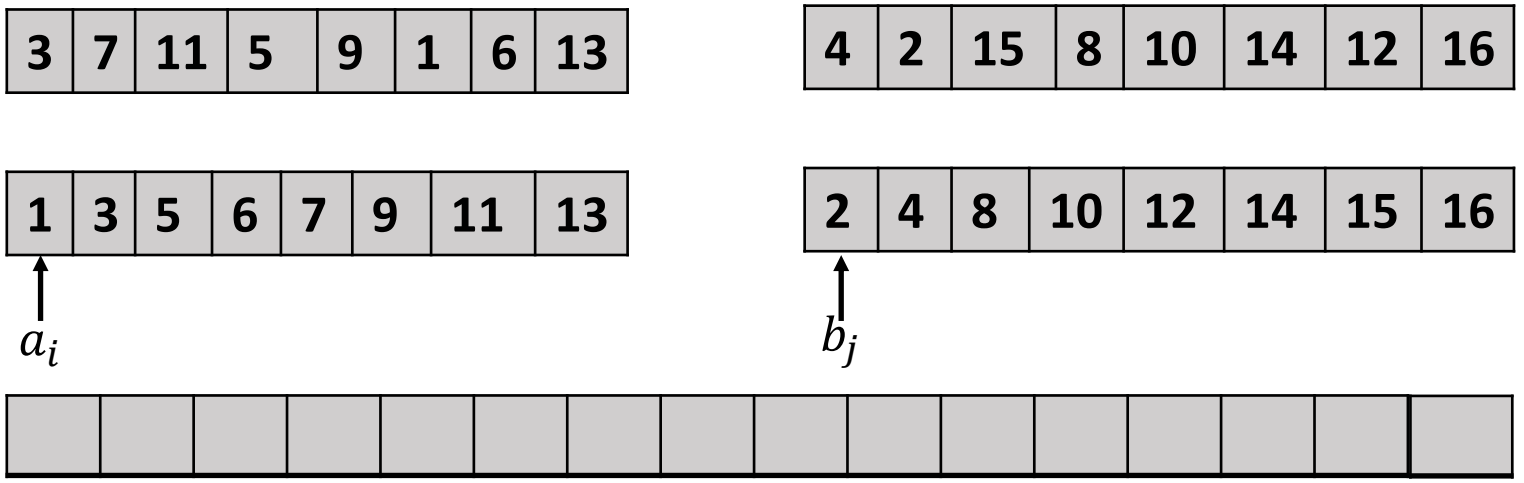
11 $j \leftarrow j + 1$

12 $inversions \leftarrow inversions + x$

13 **return** $(A, inversions)$

$inversions$=0

x=8



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$

2 $x \leftarrow length(A_1)$

3 $inversions \leftarrow 0$

4 $i \leftarrow 1; j \leftarrow 1$

5 **for** $k = 1$ **to** m

6 **if** $A_1[i] \leq A_2[j]$

7 $A[k] \leftarrow A_1[i]$

8 $i \leftarrow i + 1$

9 $x \leftarrow x - 1$

10 **else** $A[k] \leftarrow A_2[j]$

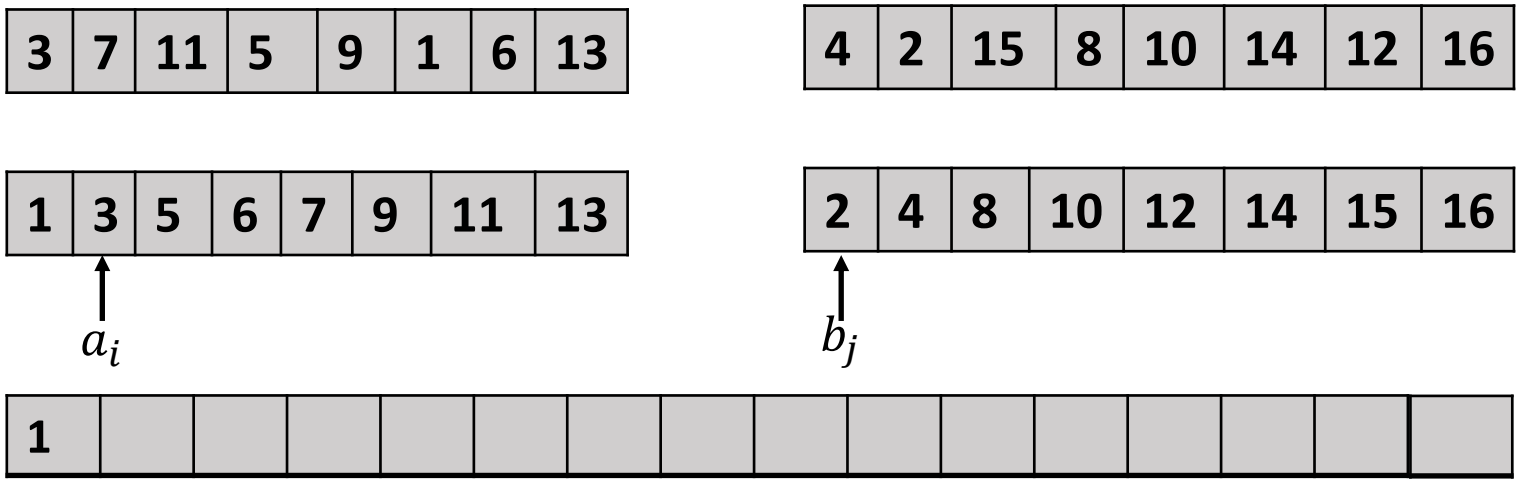
11 $j \leftarrow j + 1$

12 $inversions \leftarrow inversions + x$

13 **return** $(A, inversions)$

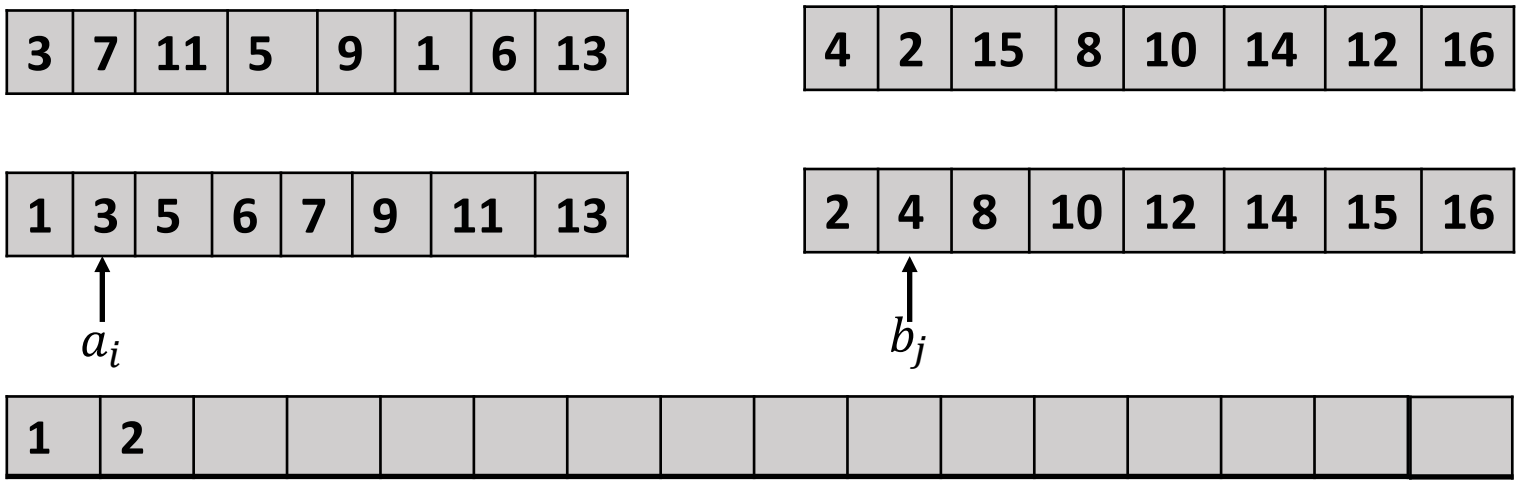
$inversions$=0

x=7



Counting inversions: how to combine two subproblems?

```
Merge − And − Count(A1, A2)
1    m ← length(A1) + length(A2)
2    x ← length(A1)
3    inversions ← 0
4    i ← 1; j ← 1
5    for k = 1 to m
6        if A1[i] ≤ A2[j]
7            A[k] ← A1[i]
8            i ← i + 1
9            x ← x − 1
10       else A[k] ← A2[j]
11           j ← j + 1
12           inversions ← inversions + x
13   return (A, inversions)
```



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$

2 $x \leftarrow length(A_1)$

3 $inversions \leftarrow 0$

4 $i \leftarrow 1; j \leftarrow 1$

5 **for** $k = 1$ **to** m

6 **if** $A_1[i] \leq A_2[j]$

7 $A[k] \leftarrow A_1[i]$

8 $i \leftarrow i + 1$

9 $x \leftarrow x - 1$

10 **else** $A[k] \leftarrow A_2[j]$

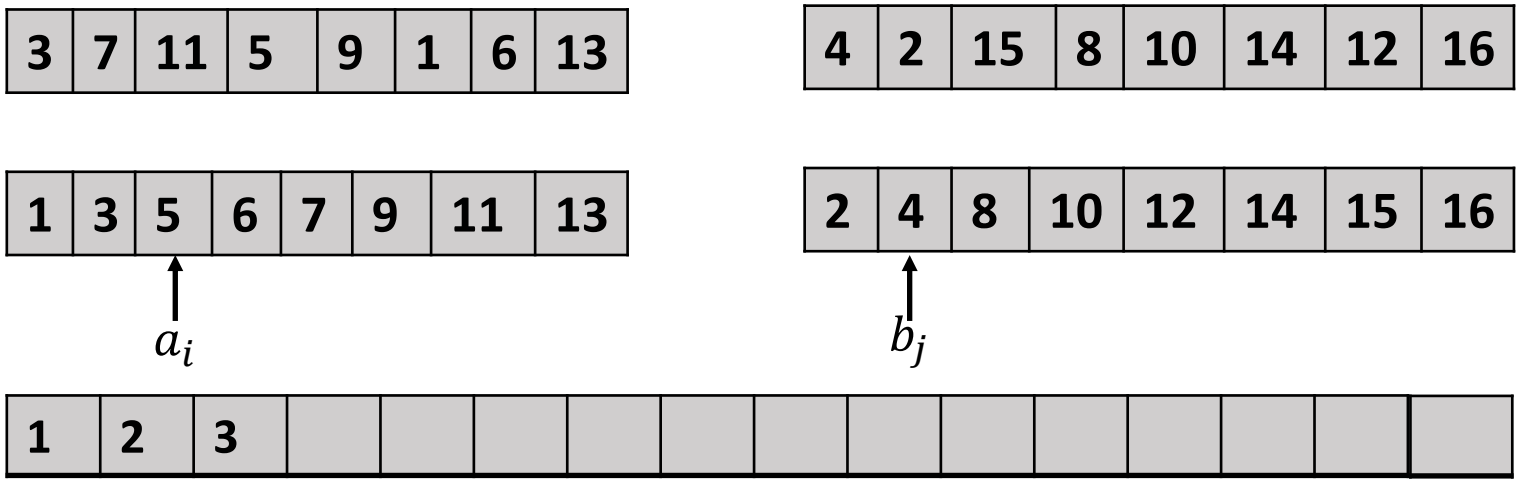
11 $j \leftarrow j + 1$

12 $inversions \leftarrow inversions + x$

13 **return** $(A, inversions)$

$inversions$=7

x=6



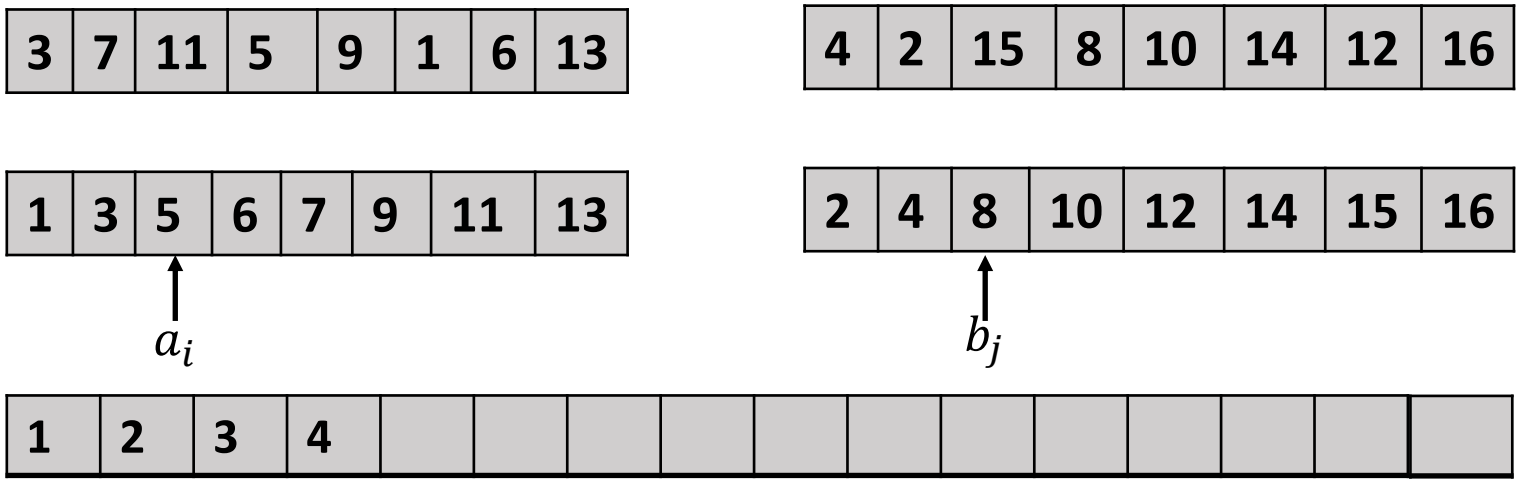
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=13

x=6



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$

2 $x \leftarrow length(A_1)$

3 $inversions \leftarrow 0$

4 $i \leftarrow 1; j \leftarrow 1$

5 **for** $k = 1$ **to** m

6 **if** $A_1[i] \leq A_2[j]$

7 $A[k] \leftarrow A_1[i]$

8 $i \leftarrow i + 1$

9 $x \leftarrow x - 1$

10 **else** $A[k] \leftarrow A_2[j]$

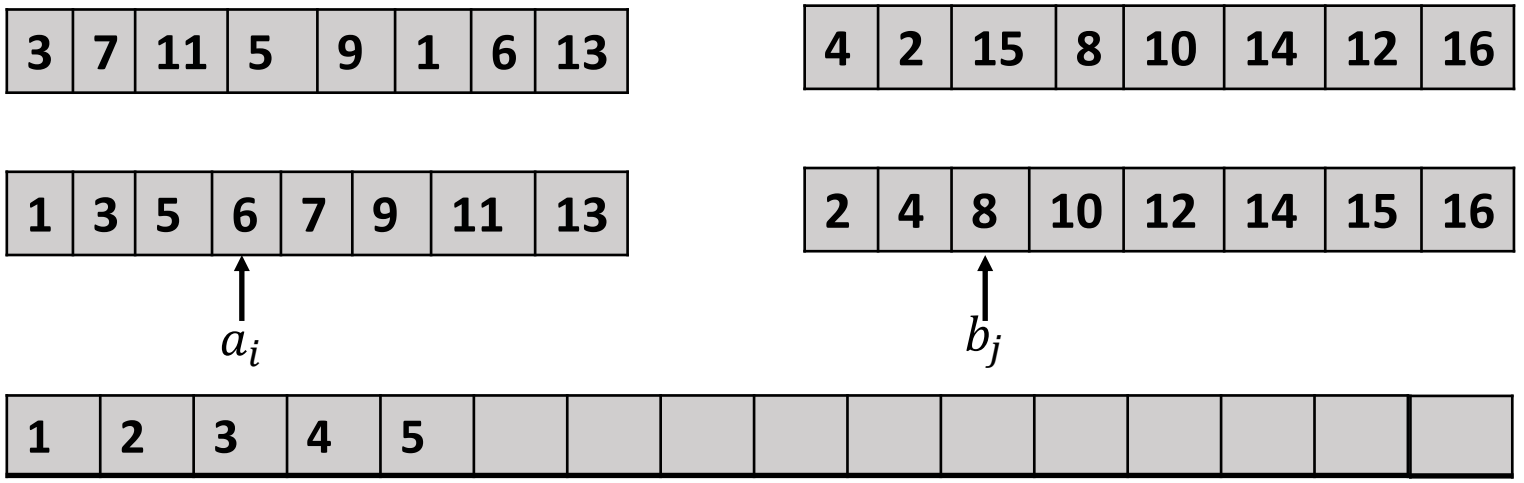
11 $j \leftarrow j + 1$

12 $inversions \leftarrow inversions + x$

13 **return** $(A, inversions)$

$inversions$=13

x=5



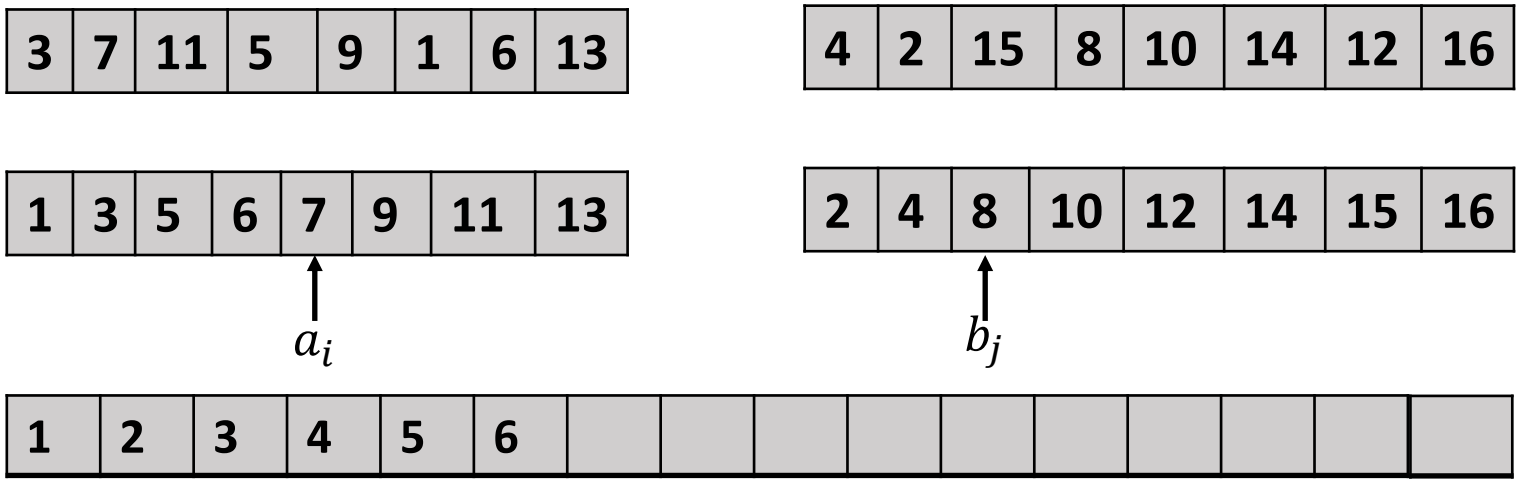
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=13

x=4



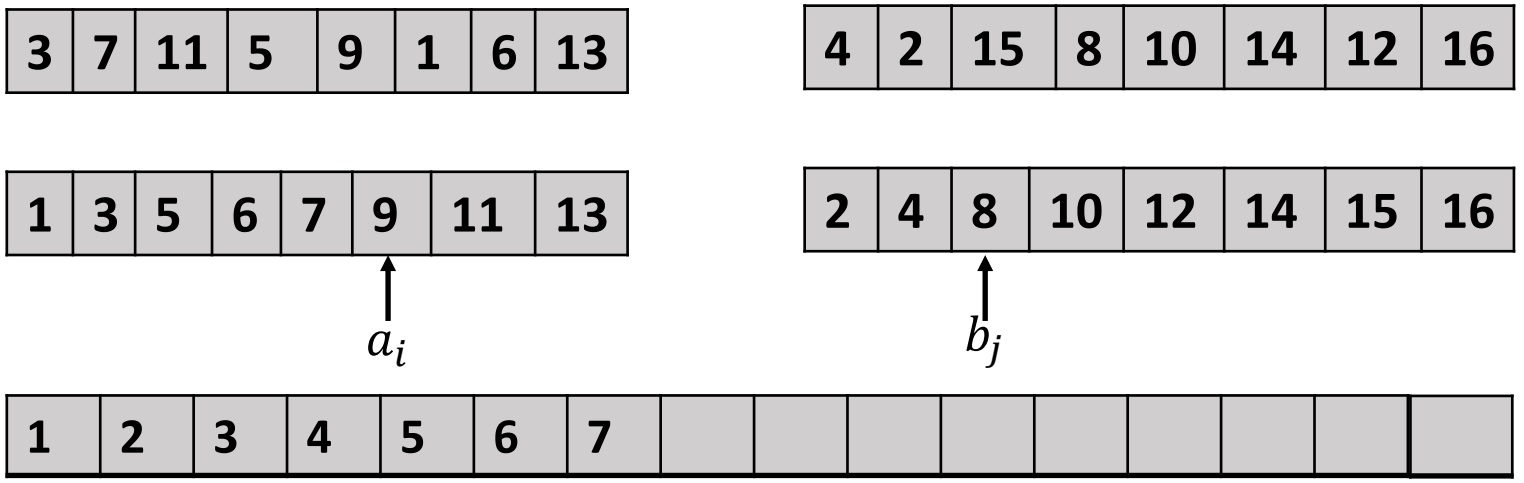
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=13

x=3

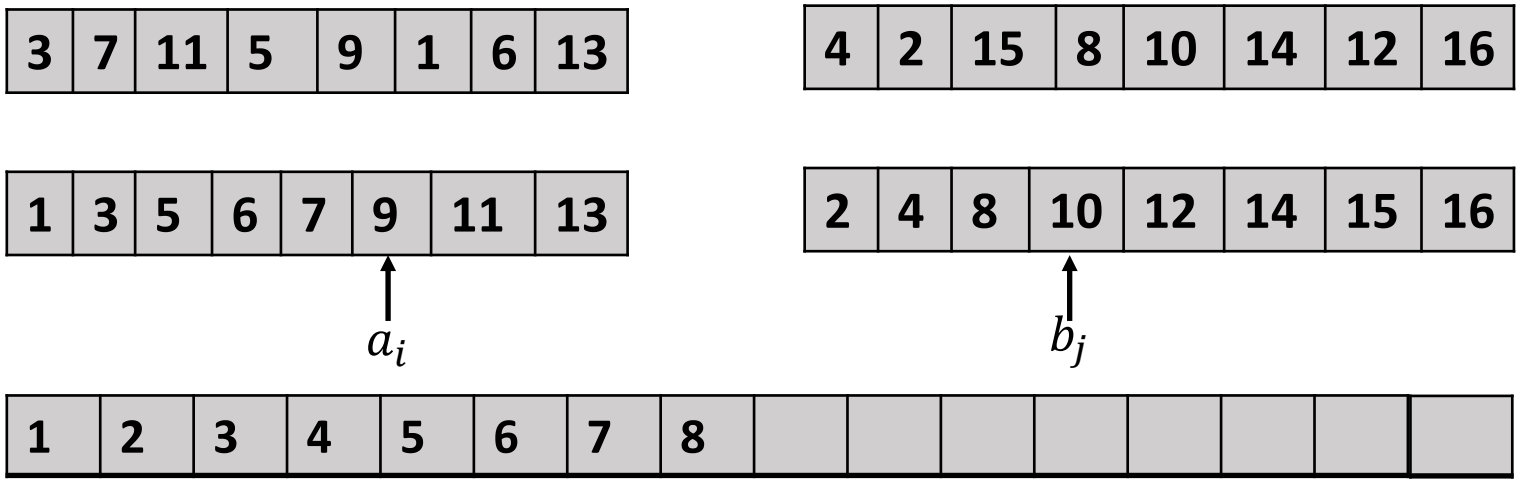


Counting inversions: how to combine two subproblems?

```
Merge − And − Count(A1, A2)
1    m ← length(A1) + length(A2)
2    x ← length(A1)
3    inversions ← 0
4    i ← 1; j ← 1
5    for k = 1 to m
6        if A1[i] ≤ A2[j]
7            A[k] ← A1[i]
8            i ← i + 1
9            x ← x − 1
10       else A[k] ← A2[j]
11           j ← j + 1
12           inversions ← inversions + x
13   return (A, inversions)
```

$inversions$=16

x=3



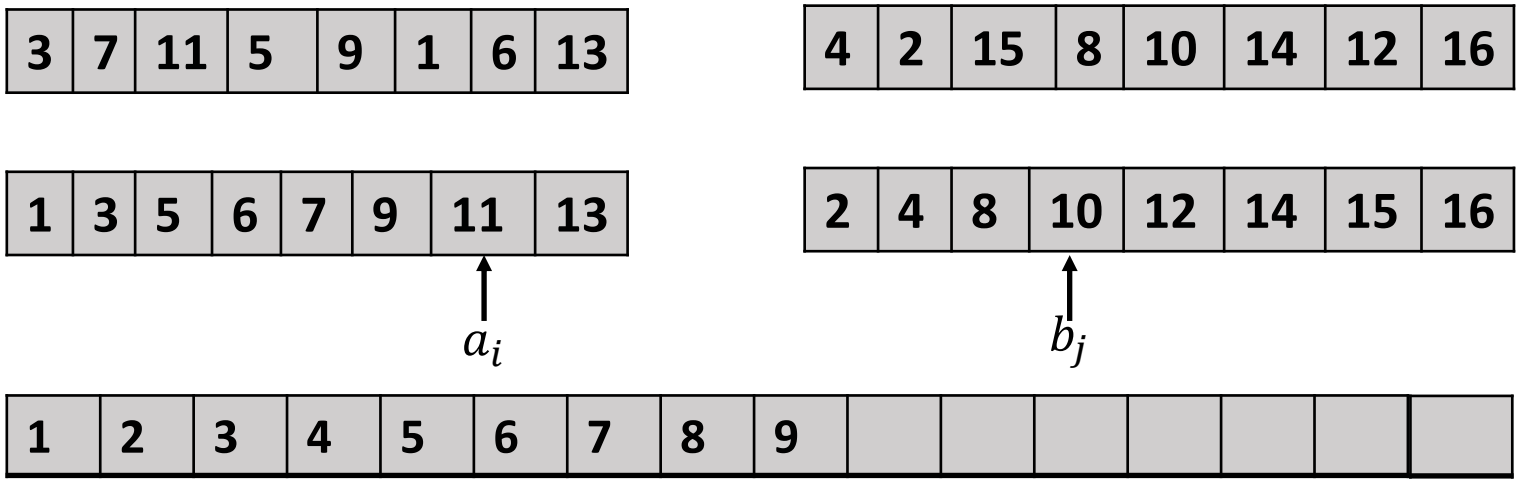
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=16

x=2



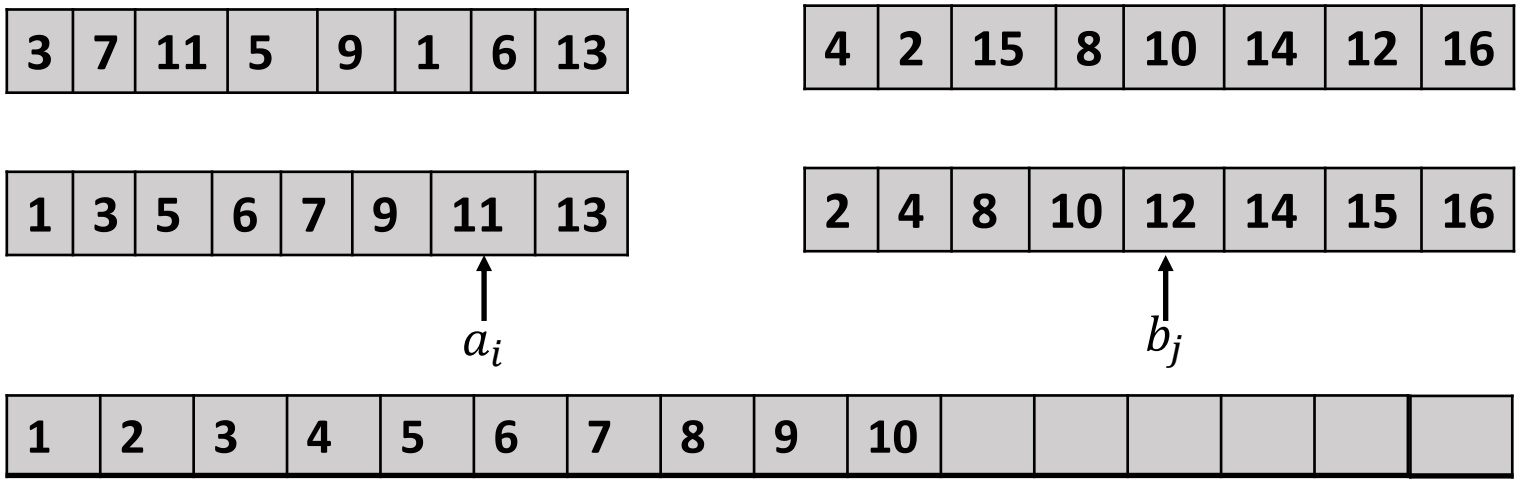
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=18

x=2



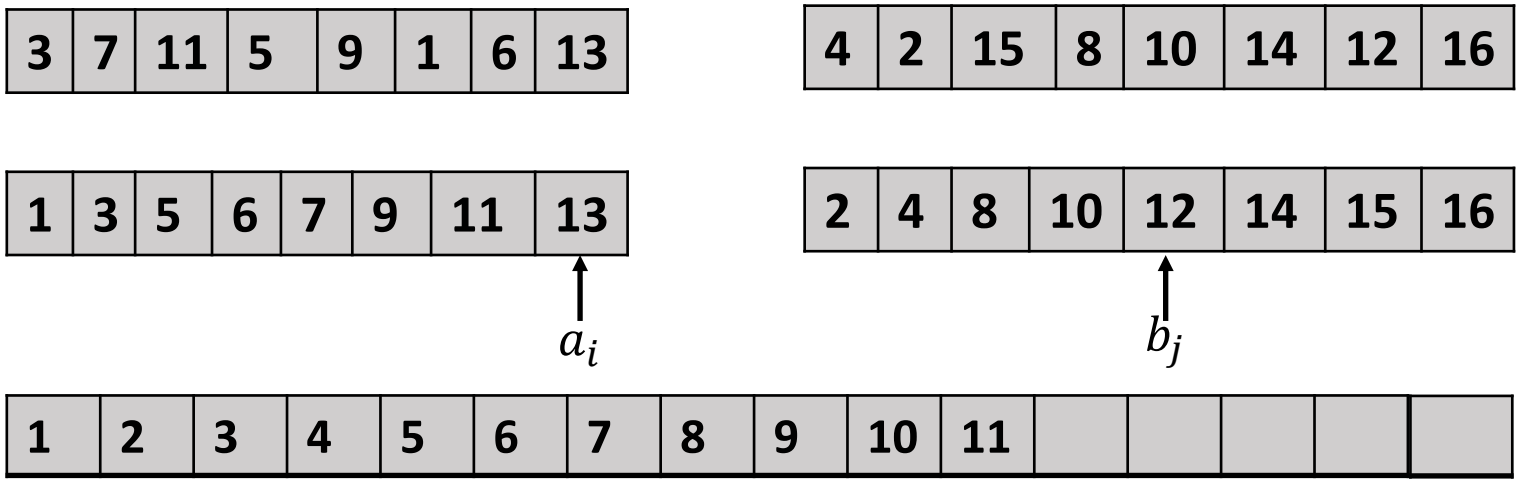
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=18

x=1



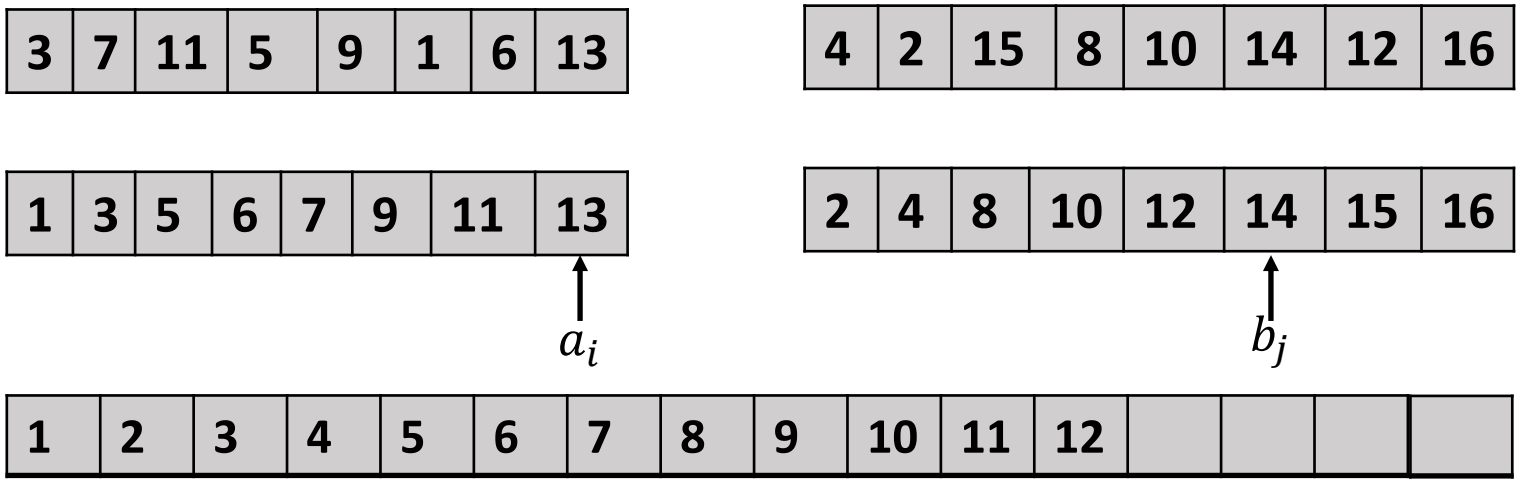
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=19

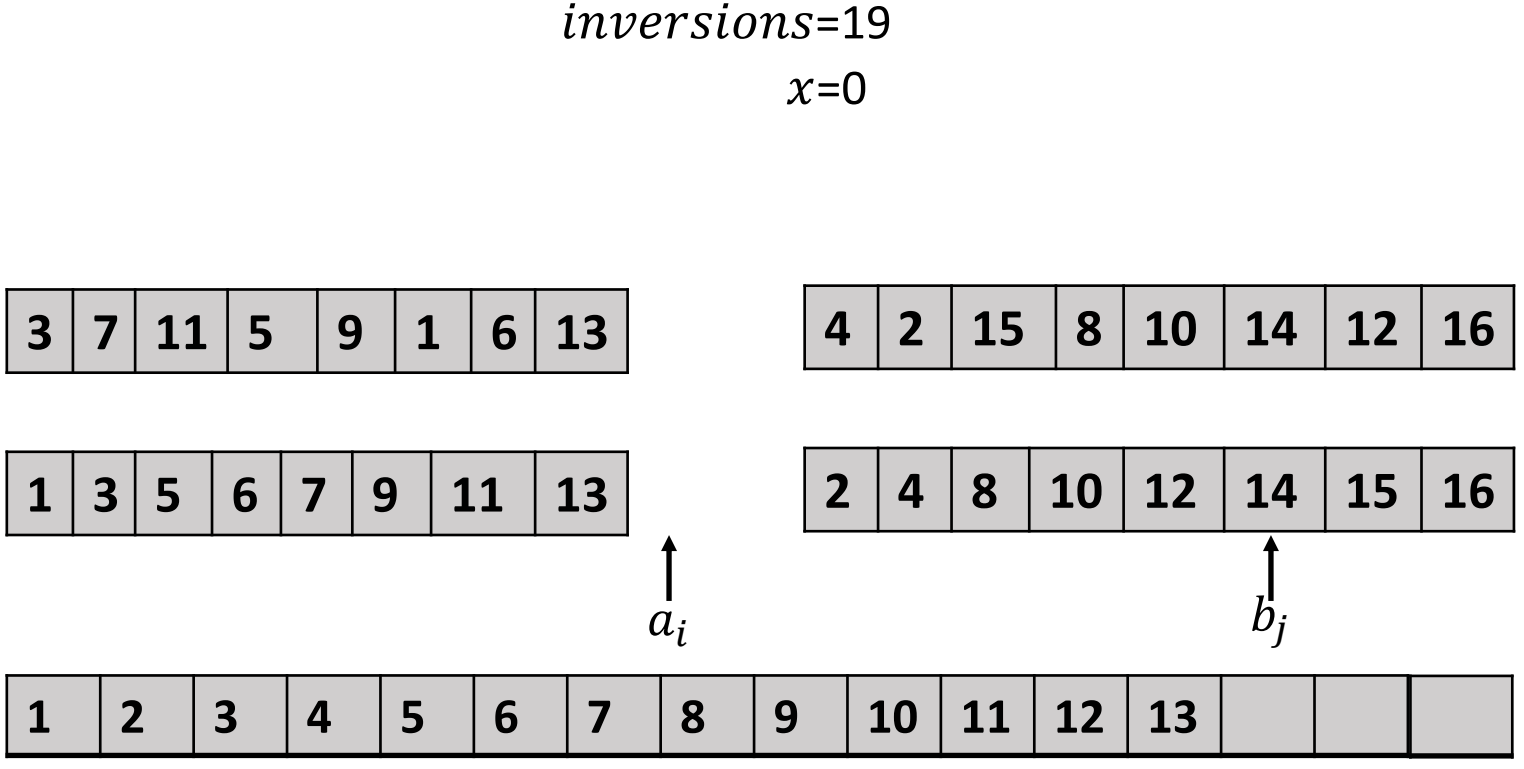
x=1



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

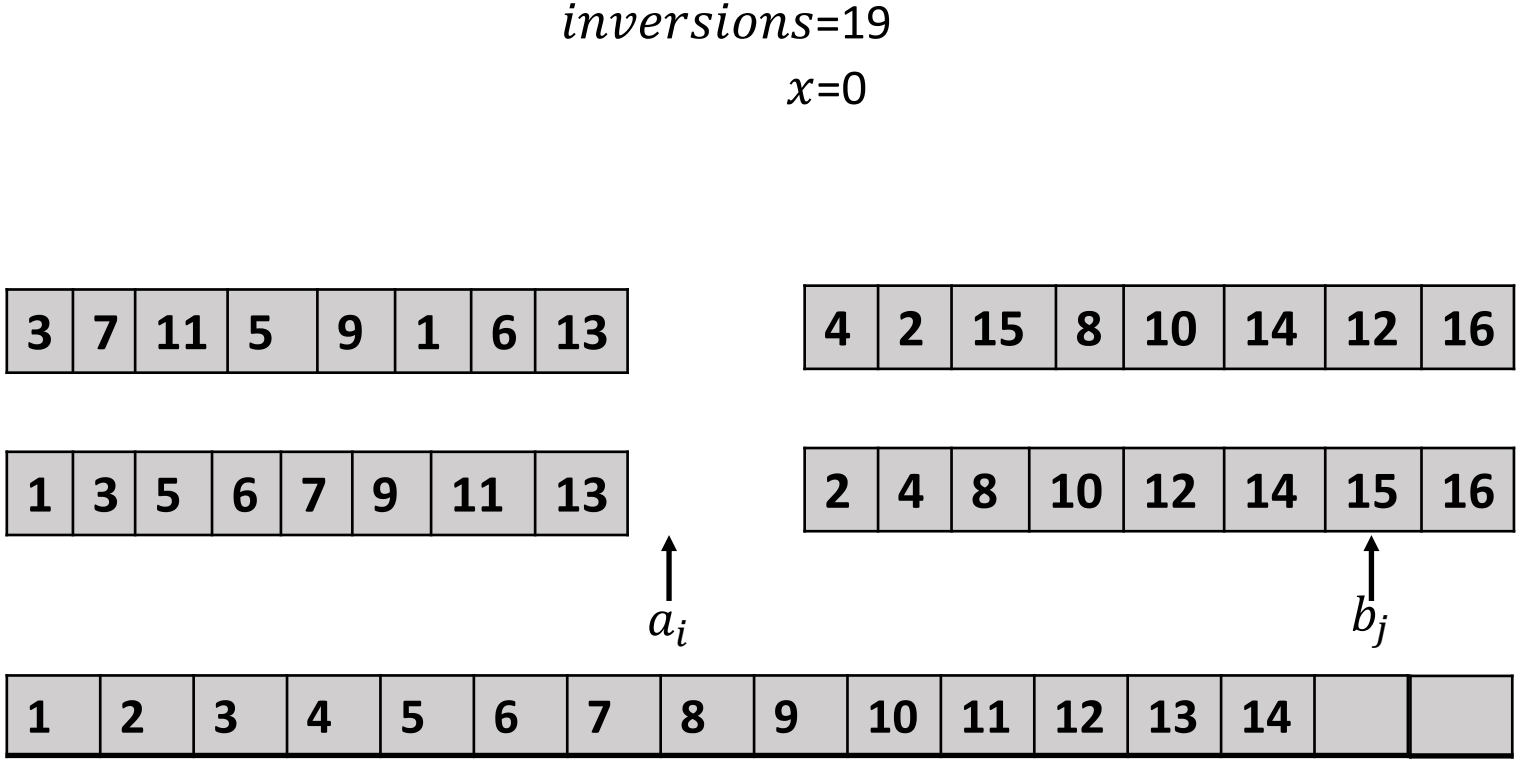
1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

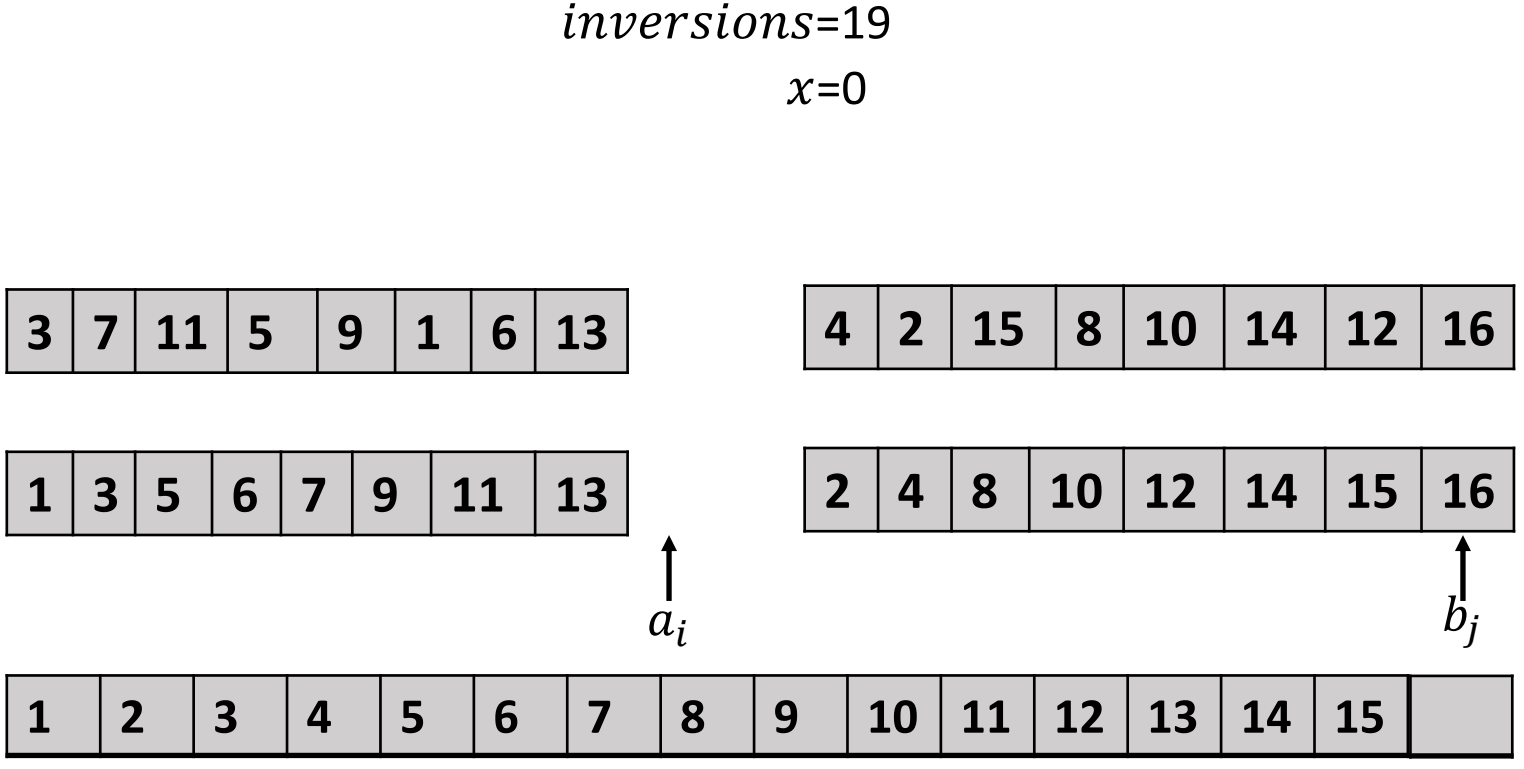
1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$



Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$



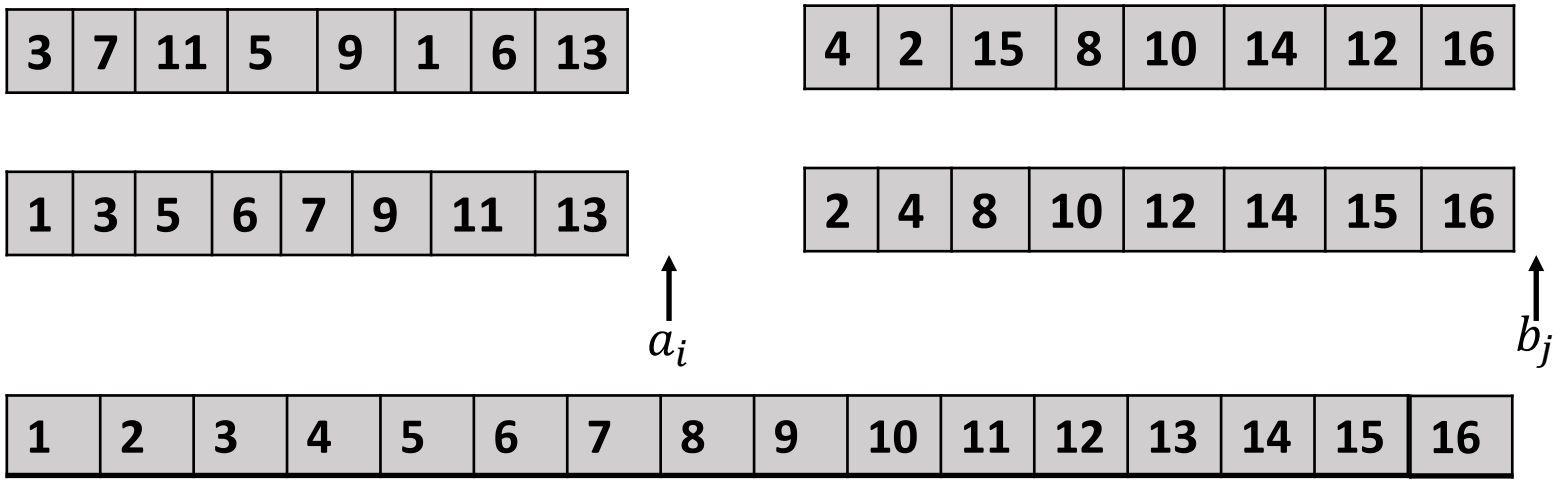
Counting inversions: how to combine two subproblems?

$Merge - And - Count(A_1, A_2)$

1 $m \leftarrow length(A_1) + length(A_2)$
2 $x \leftarrow length(A_1)$
3 $inversions \leftarrow 0$
4 $i \leftarrow 1; j \leftarrow 1$
5 **for** $k = 1$ **to** m
6 **if** $A_1[i] \leq A_2[j]$
7 $A[k] \leftarrow A_1[i]$
8 $i \leftarrow i + 1$
9 $x \leftarrow x - 1$
10 **else** $A[k] \leftarrow A_2[j]$
11 $j \leftarrow j + 1$
12 $inversions \leftarrow inversions + x$
13 **return** $(A, inversions)$

$inversions$=19

x=0



Counting inversions: how to combine two subproblems?

$Sort - And - Count(A)$

1 **if** $(length(A) > 1)$

2 $A_1 \leftarrow A[1 \cdots \lfloor n/2 \rfloor]$ ← $\Theta(1)$

3 $A_2 \leftarrow A[\lfloor n/2 \rfloor + 1 \cdots n]$ ← $\Theta(1)$

4 $(A_1, A_{1\_inversions}) \leftarrow Sort - And - Count(A_1)$ ← $T(n/2)$

5 $(A_2, A_{2\_inversions}) \leftarrow Sort - And - Count(A_2)$ ← $T(n/2)$

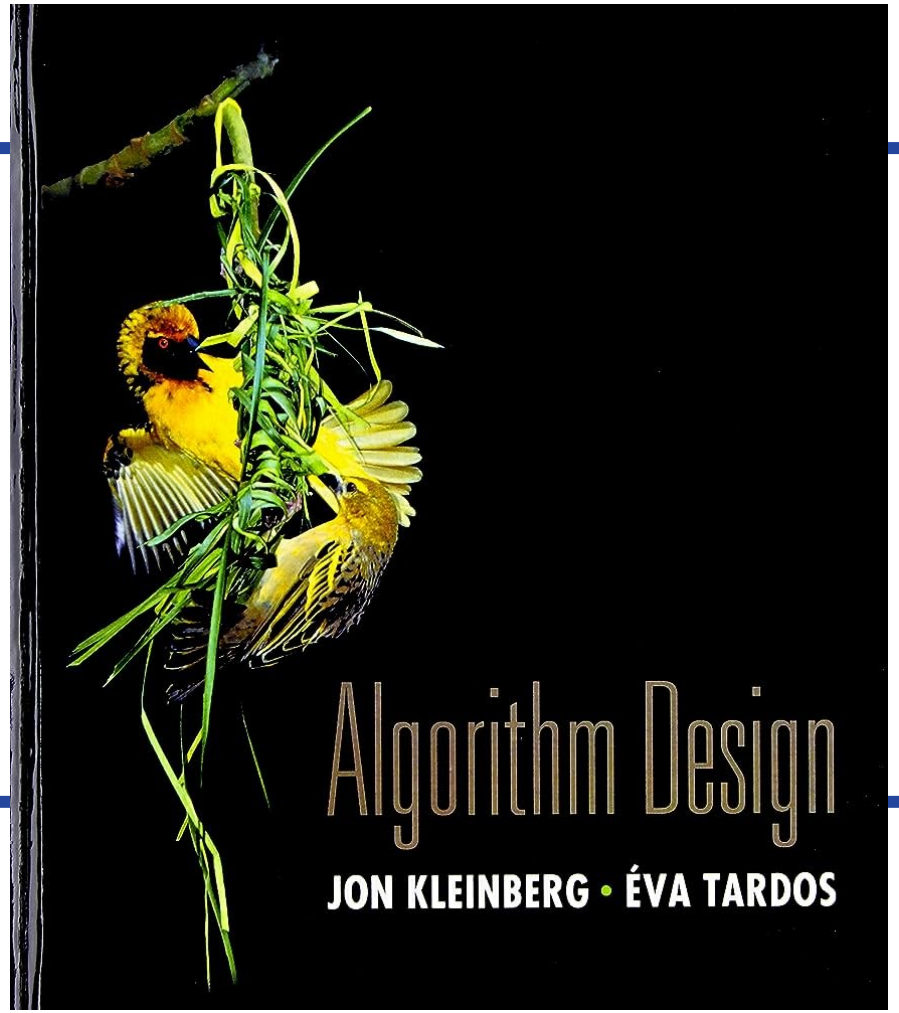
6 $(A, A_{inversions}) \leftarrow Merge - And - Count(A_1, A_2)$ ← $\Theta(n)$

7 **return** A

Counting inversions: divide-and-conquer algorithm analysis

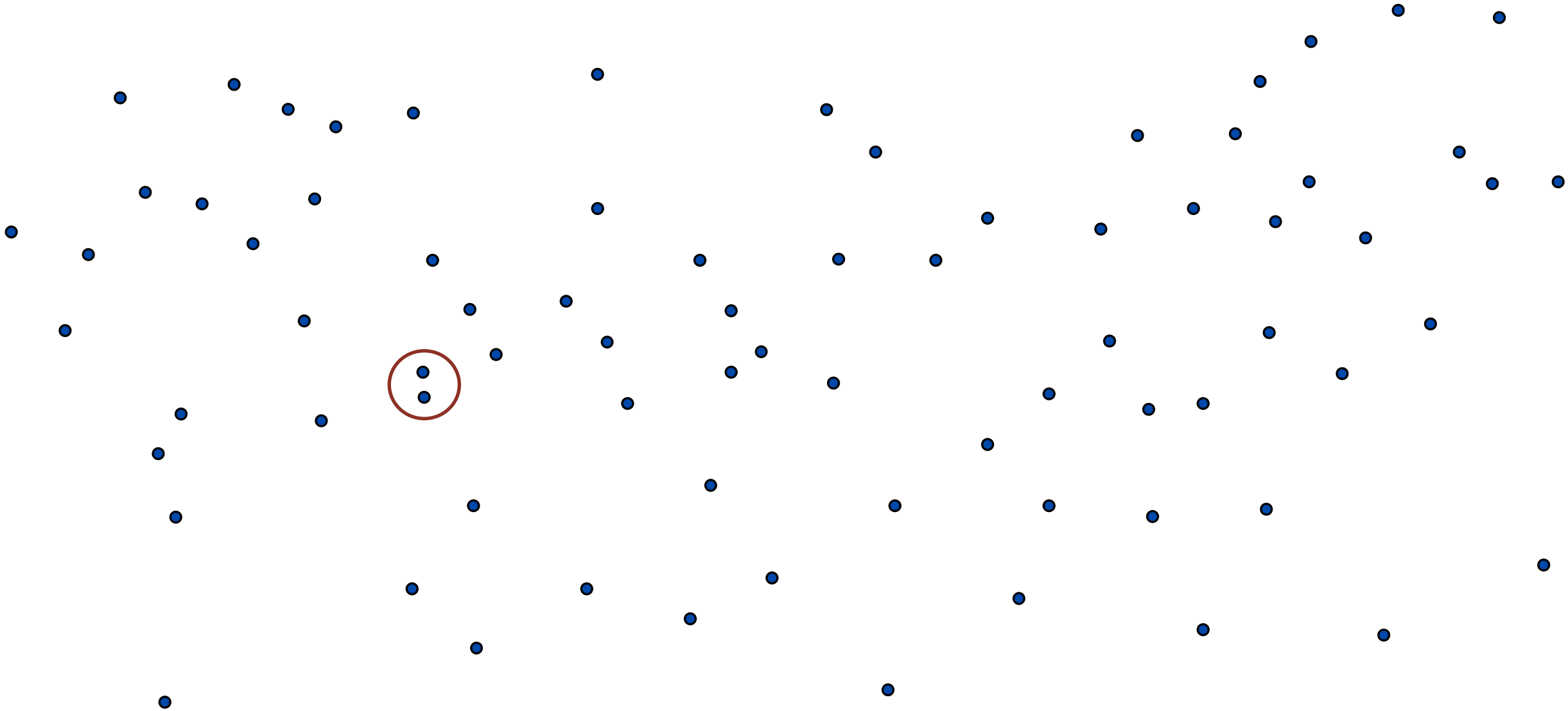
- Proposition. The sort-and-count algorithm counts the number of inversions in a permutation of size n in $O(n \log n)$ time.
- **Proof:** The worst-case running time $T(n)$ satisfies the recurrence:

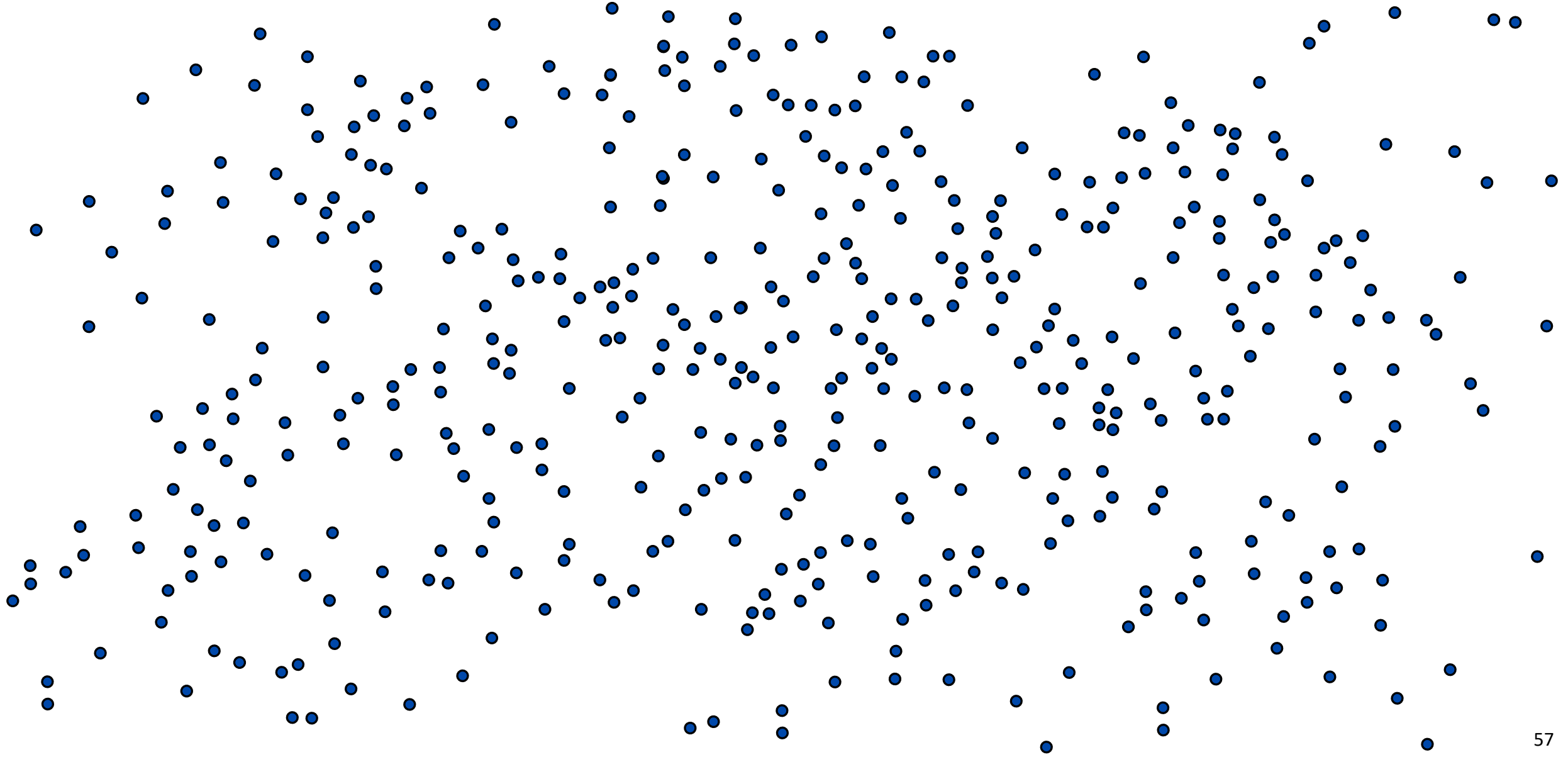
$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases}$$



Chapter 5:
Divide and Conquer

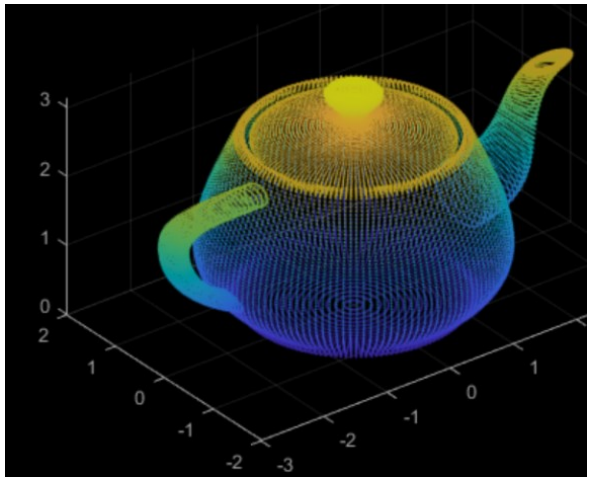
Section 5.4:
Closed Pair of Points

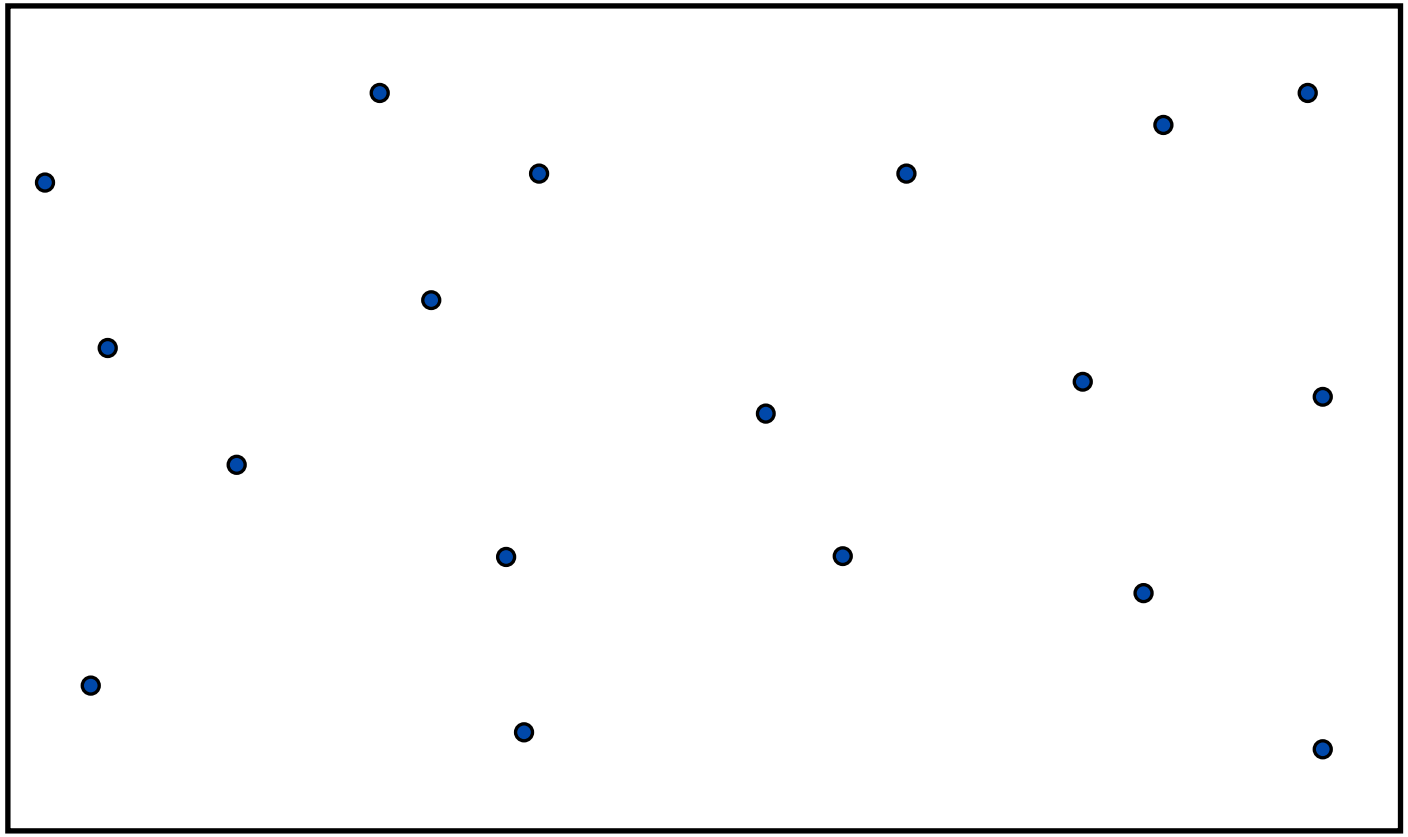


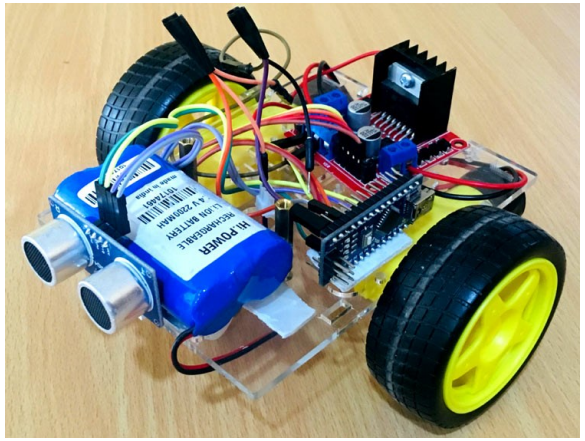


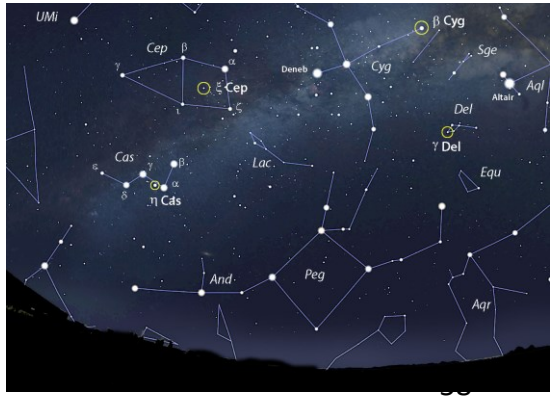
Closest pair of points

- **Closest pair problem:** Given n points in the plane, find a pair of points with the smallest Euclidean distance between them.



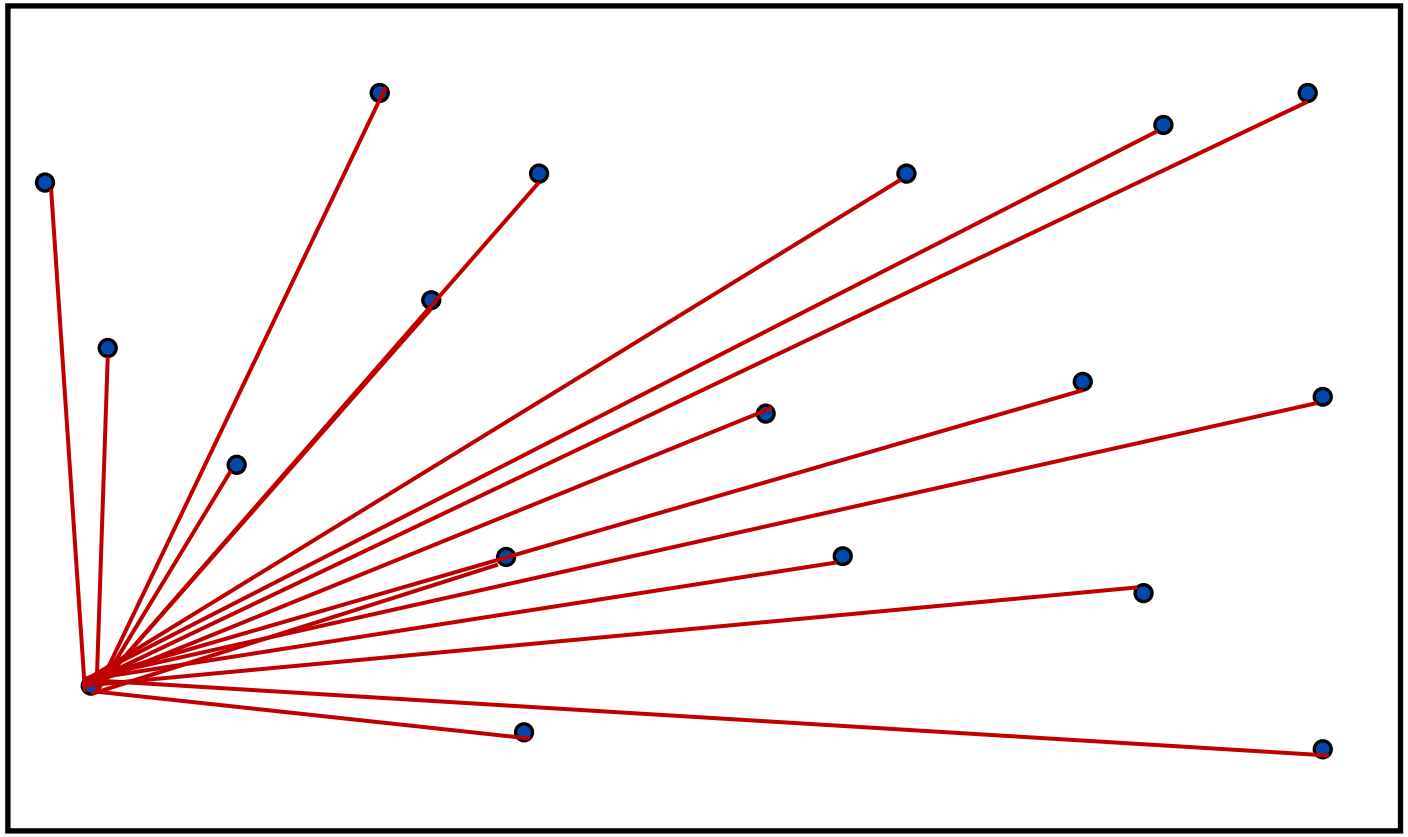






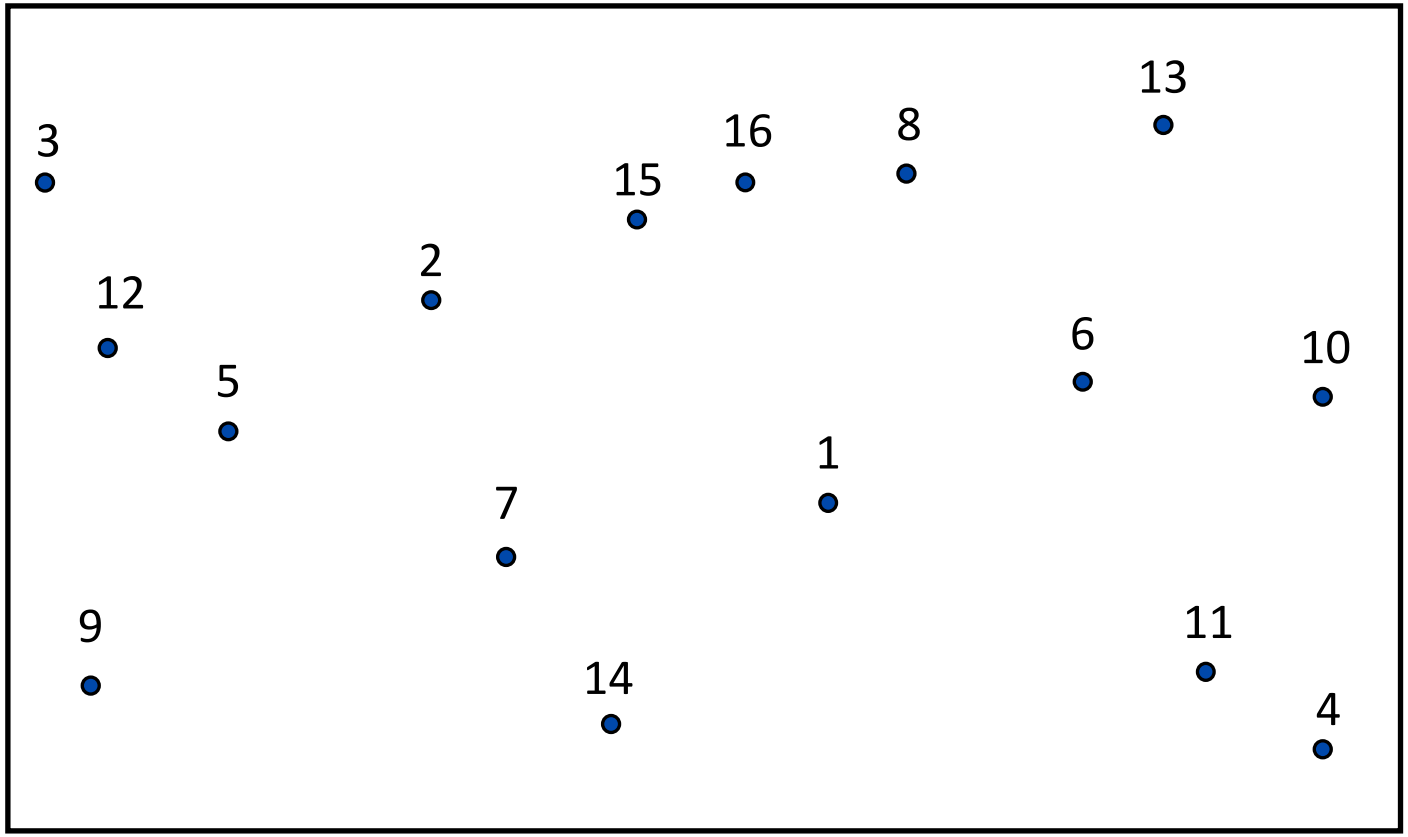
Closest pair of points

- Closest pair problem. Given n points in the plane, find a pair of points with the smallest Euclidean distance between them.



Closest pair of points

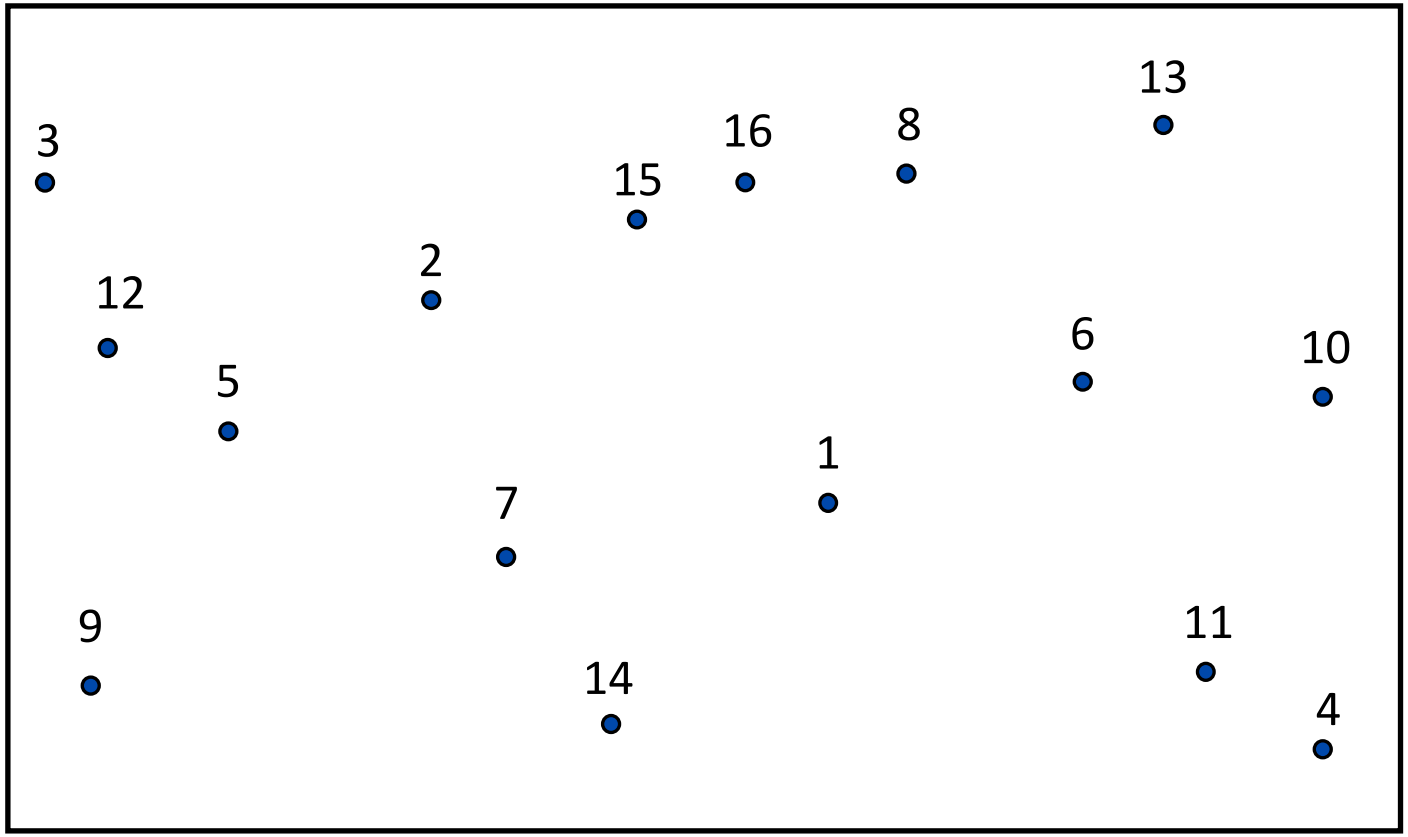
- Closest pair problem. Given n points in the plane, find a pair of points with the smallest Euclidean distance between them.



(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17)

Closest pair of points: Sorting Solution

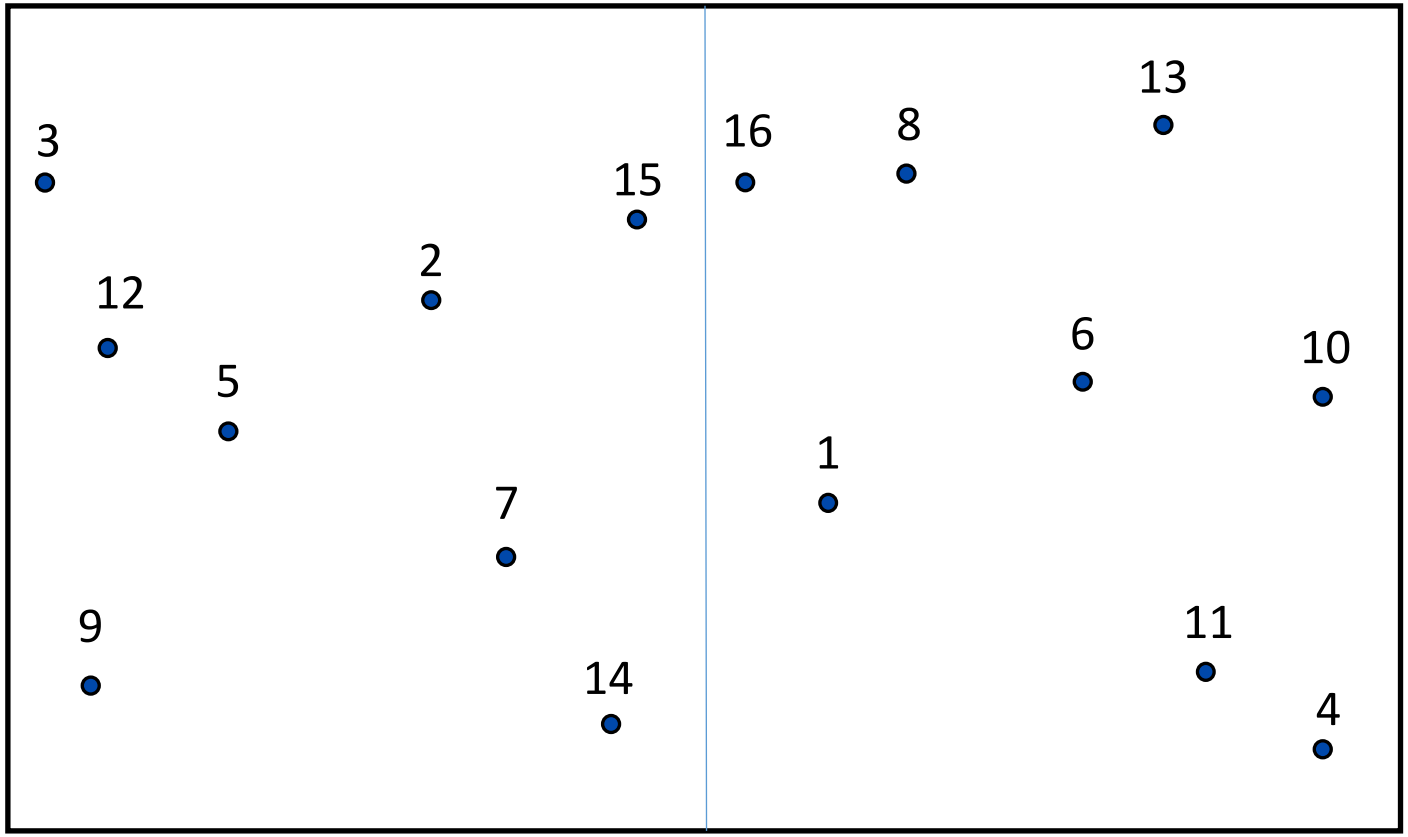
- Sort points by x-coordinate



(3, 9, 12, 5, 2, 7, 14, 15, 16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points

- Divide: draw vertical line L so that n / 2 points on each side.

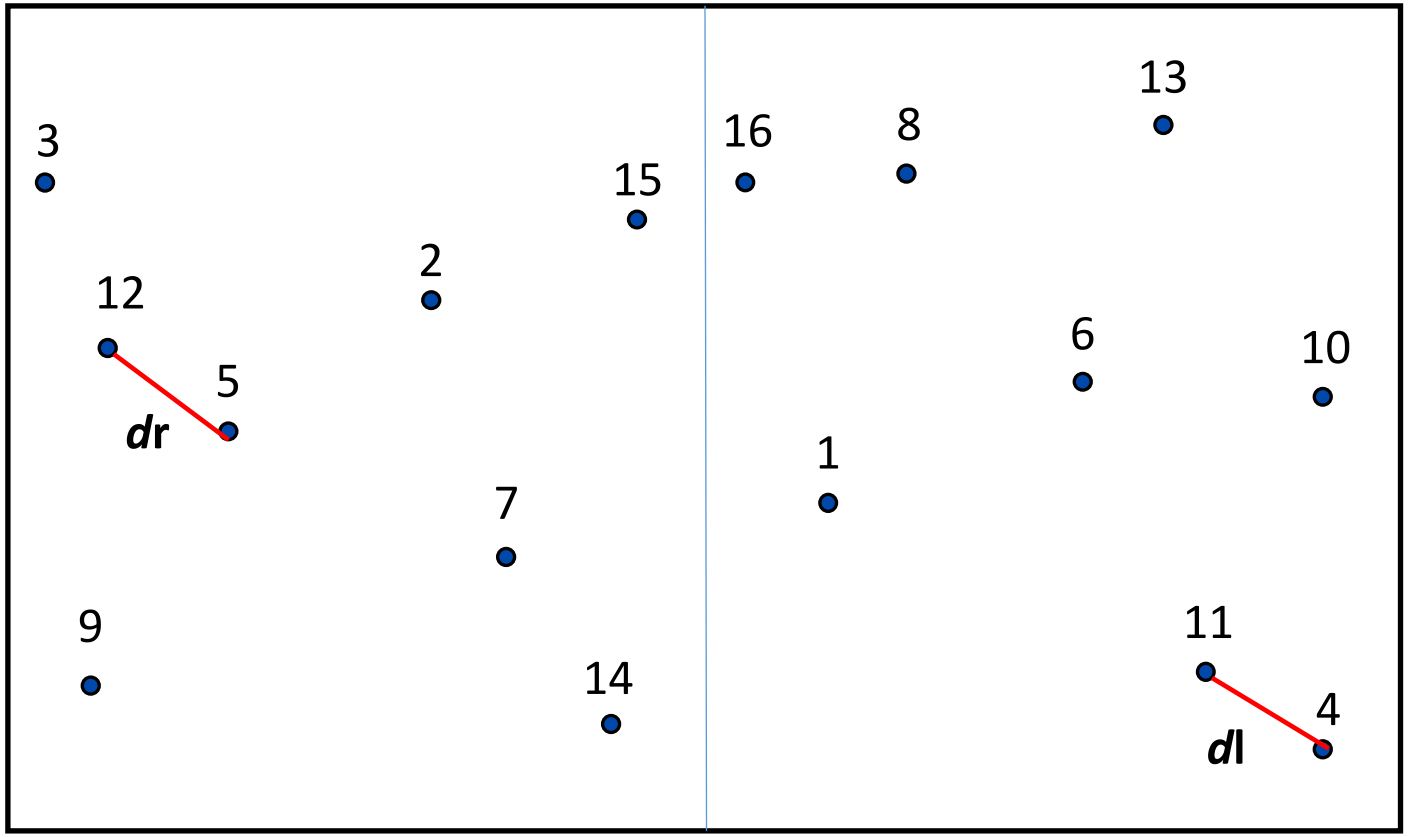


(3, 9, 12, 5, 2, 7, 14, 15)

(16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points

- Conquer: find closest pair in each side recursively.

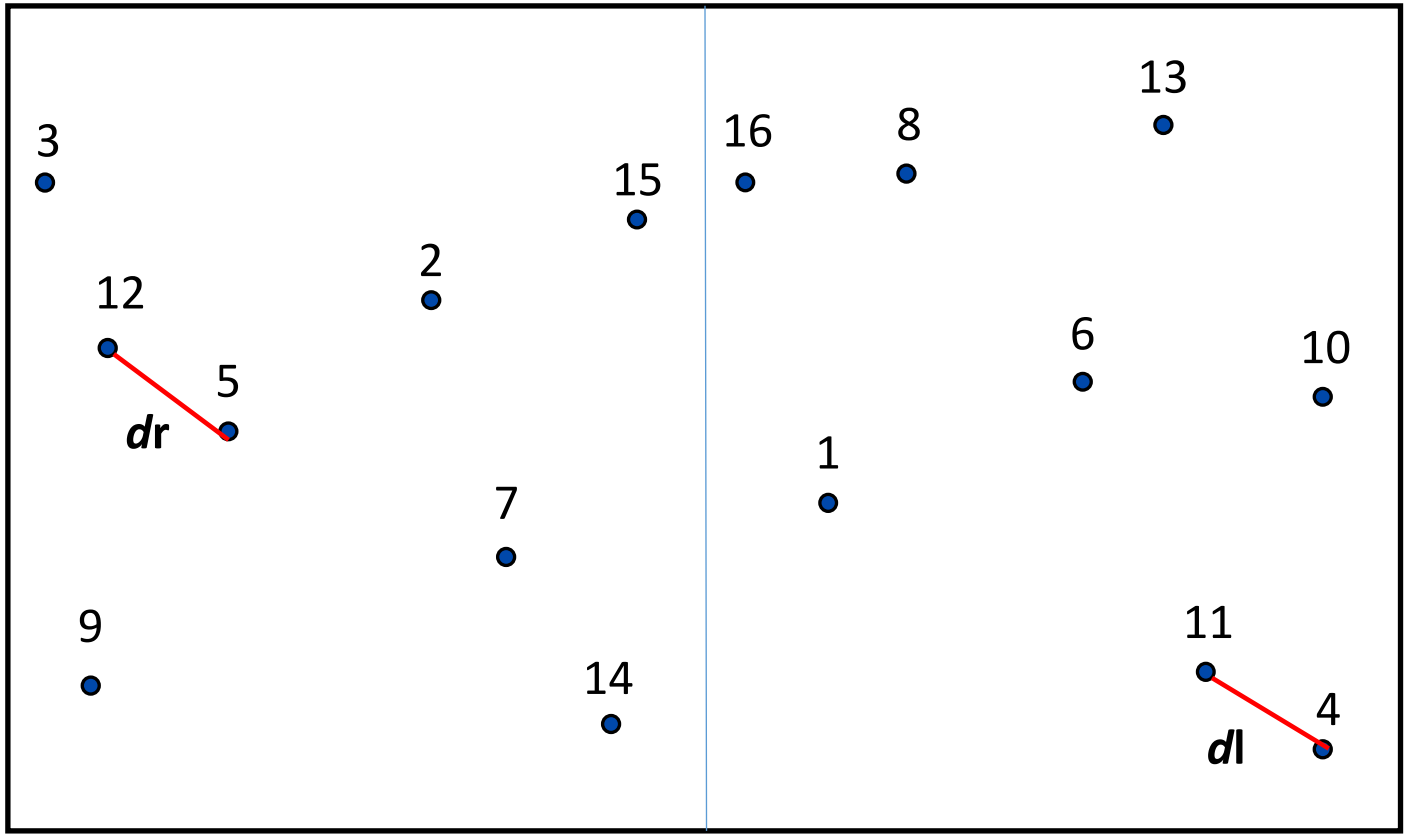


(3, 9, 12, 5, 2, 7, 14, 15)

(16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points

- Conquer: find closest pair in each side recursively.

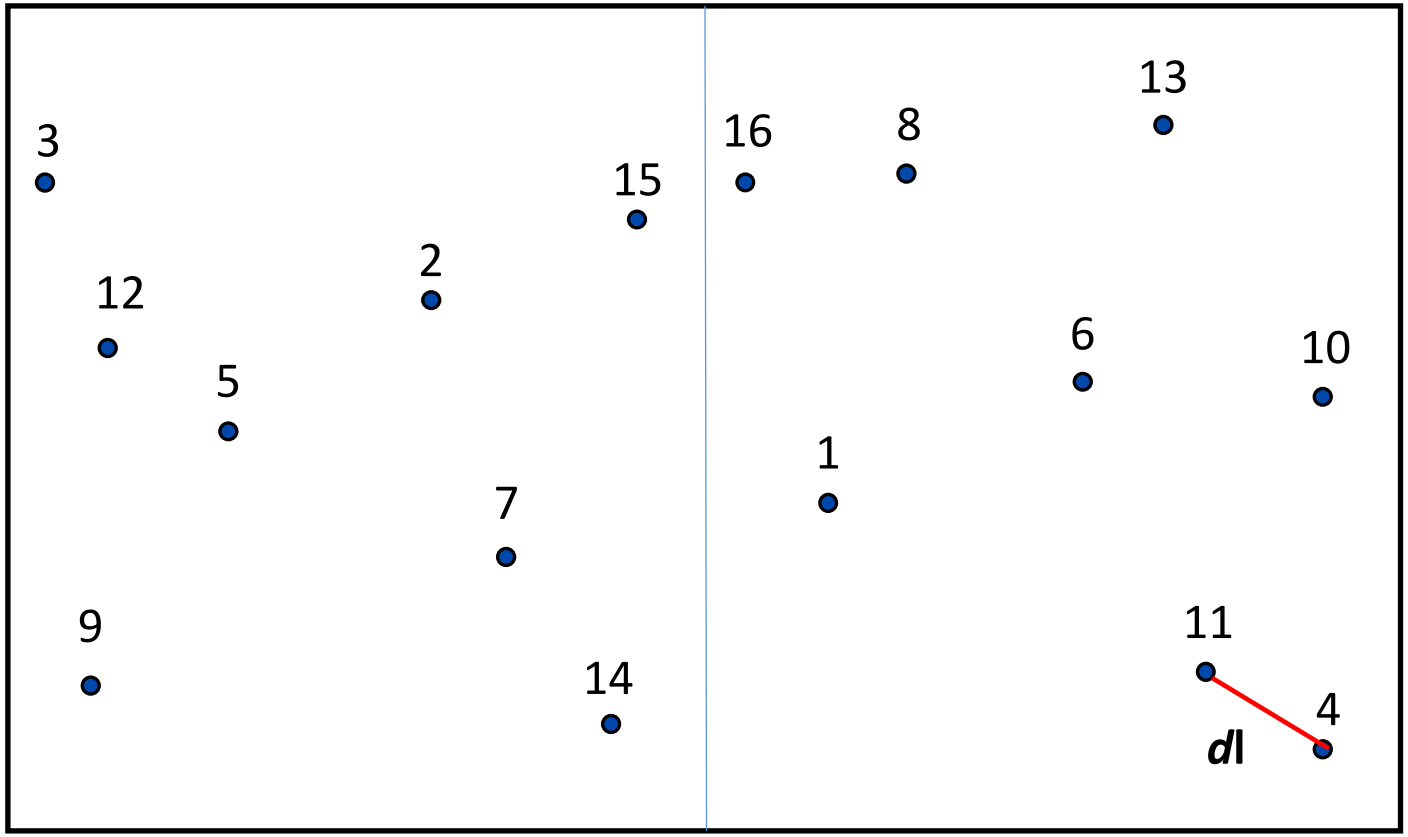


$d = \min(d_l, d_r)$

(3, 9, 12, 5, 2, 7, 14, 15)

(16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points



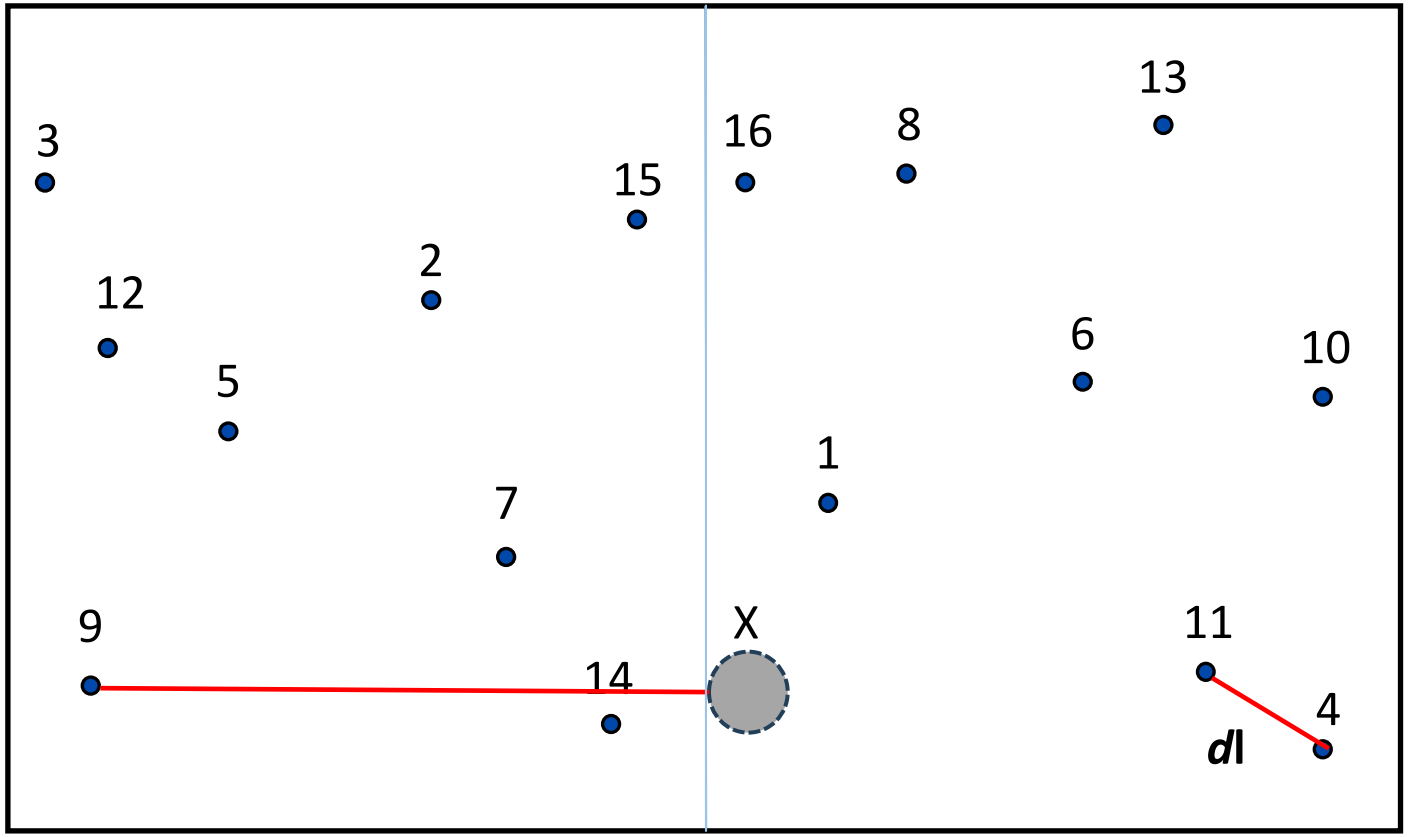
$d = \min(dl, dr)$

(3, 9, 12, 5, 2, 7, 14, 15)

(16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points

- Closest pair problem. Given n points in the plane, find a pair of points with the smallest Euclidean distance between them.



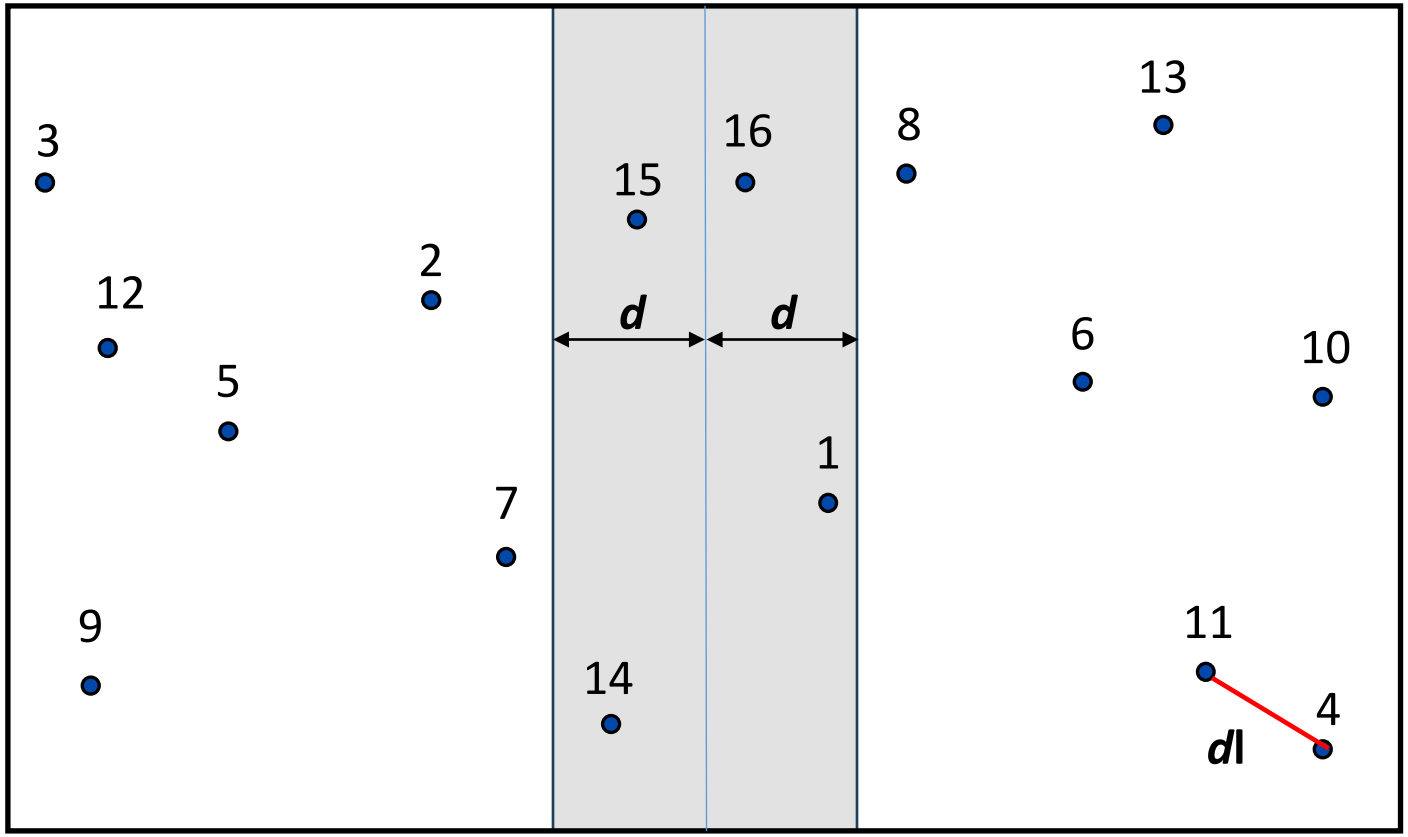
d= min(dl,dr)

(3, 9, 12, 5, 2, 7, 14, 15)

(16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points

- Observation: suffices to consider only those points within d of line L



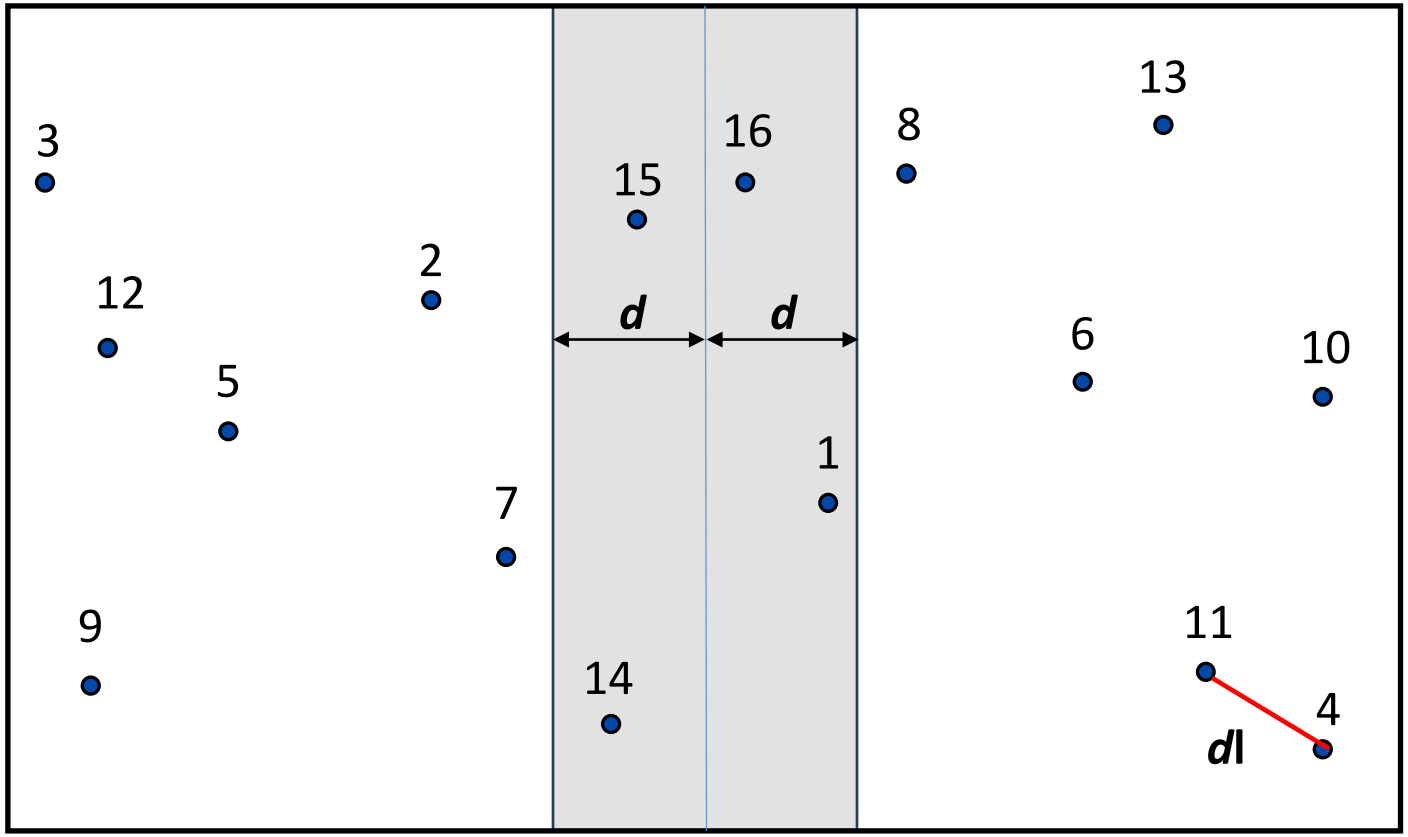
d= min(dl,dr)

(3, 9, 12, 5, 2, 7, 14, 15)

(16, 1, 8, 6, 13, 11, 10, 4)

Closest pair of points

- Observation: suffices to consider only those points within d of line L

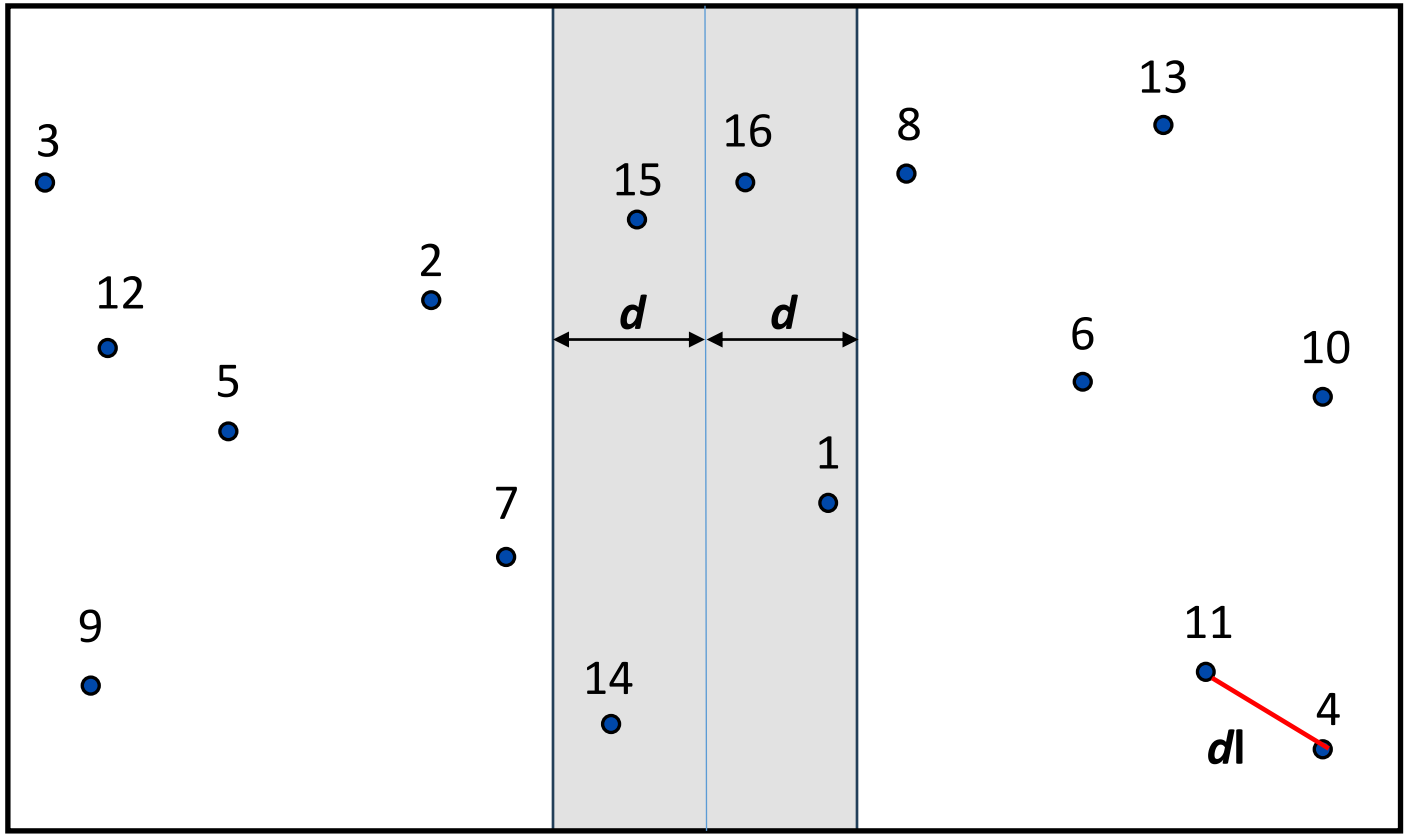


d= min(dl,dr)

(14,15,16,1)

Closest pair of points

- Reorder based on Y-axis

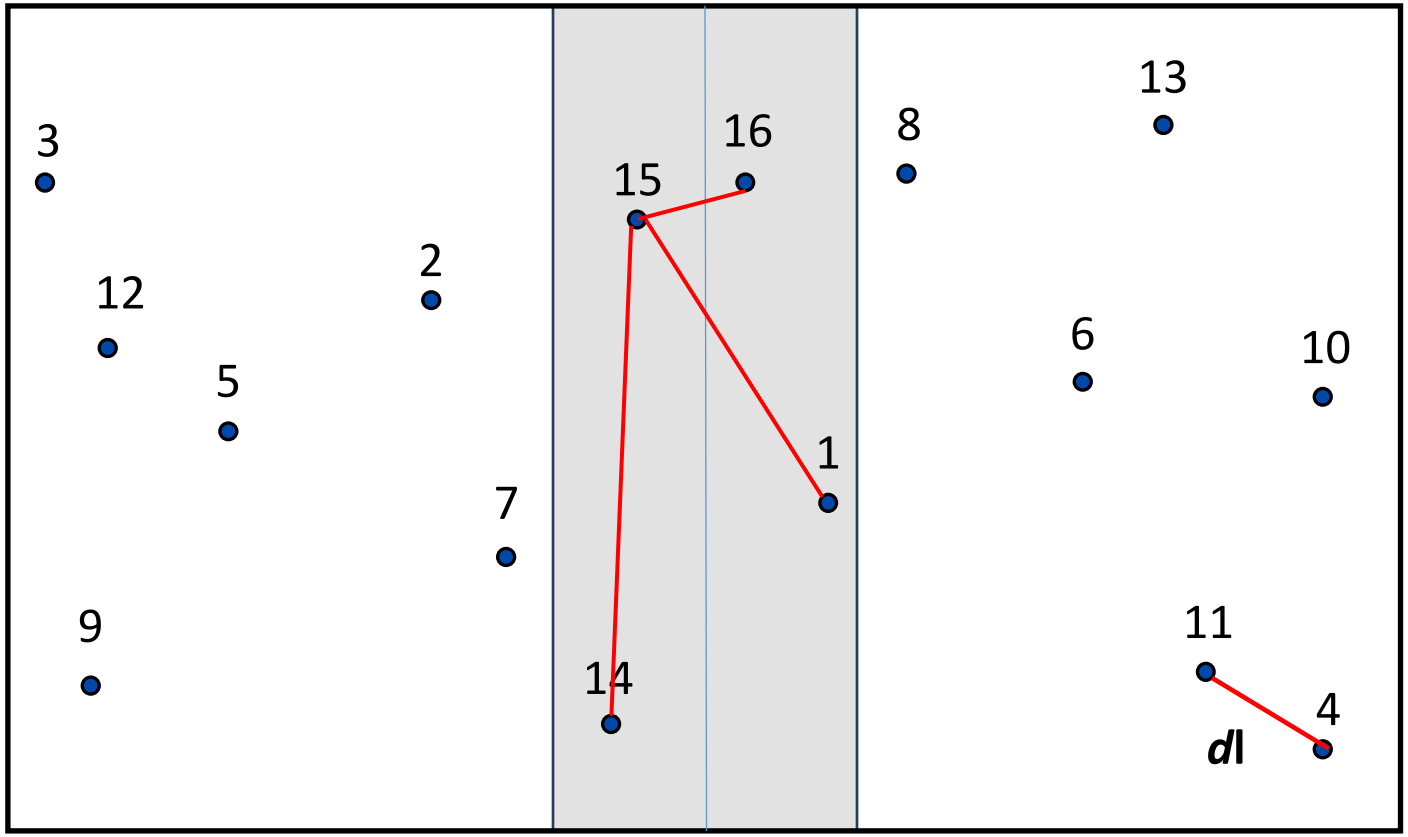


(16,15,1,14)

***d*= min(dl,dr)**

Closest pair of points

- Reorder based on Y-axis

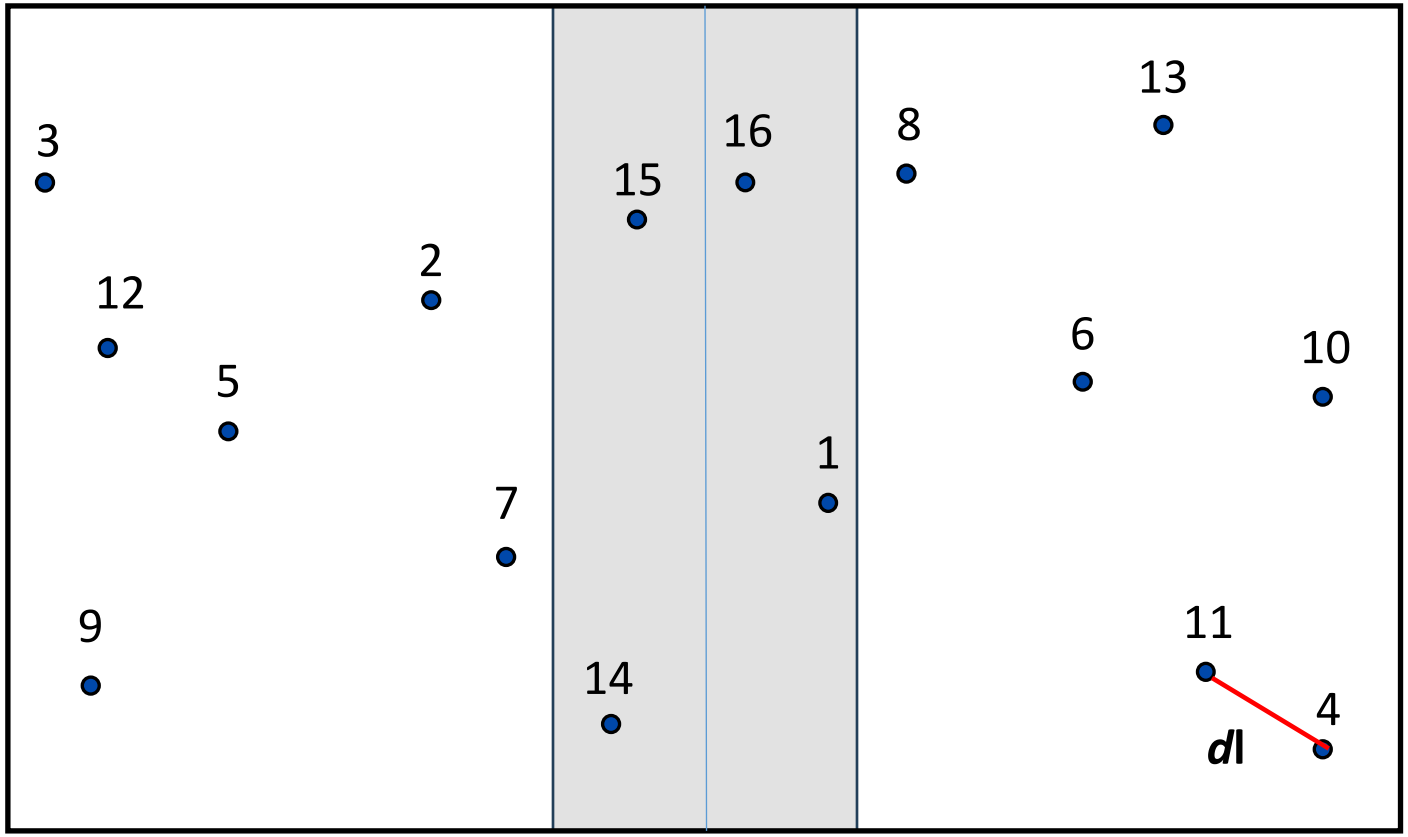


(16,15,1,14)

***d*= min(dl,dr)**

Closest pair of points

- Reorder based on Y-axis

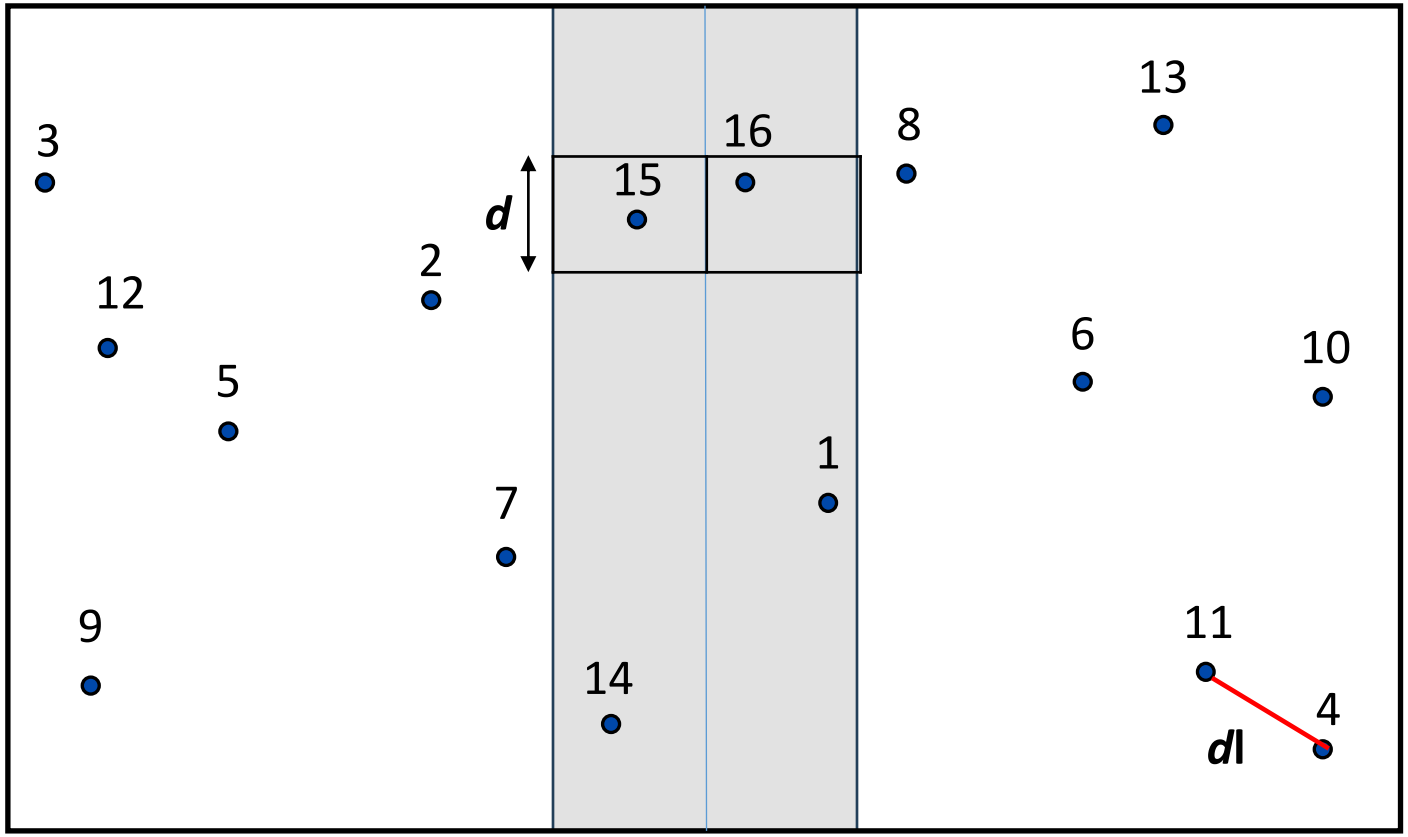


(16,15,1,14)

***d*= min(dl,dr)**

Closest pair of points

- Reorder based on Y-axis

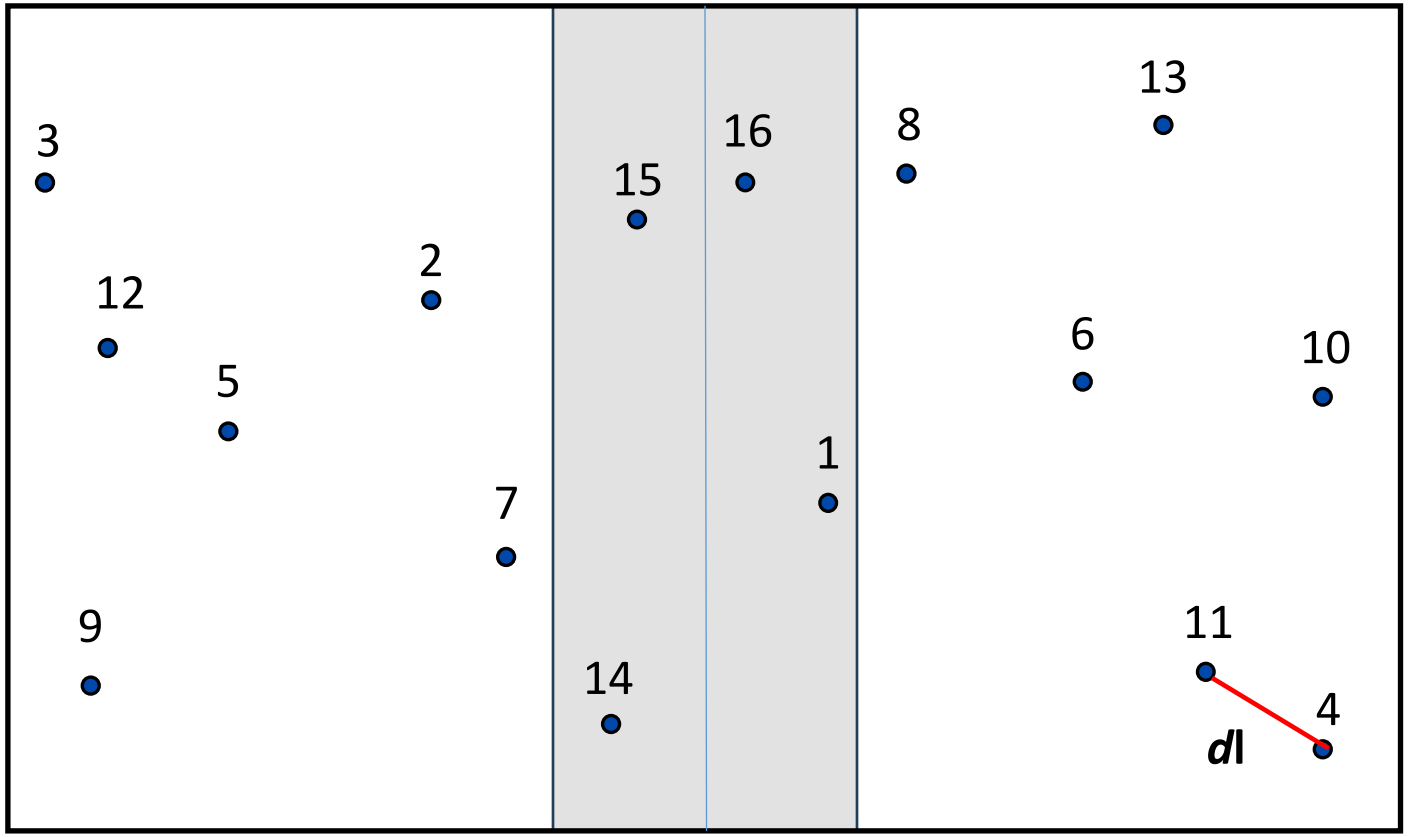


(16,15,1,14)

***d*= min(dl,dr)**

Closest pair of points

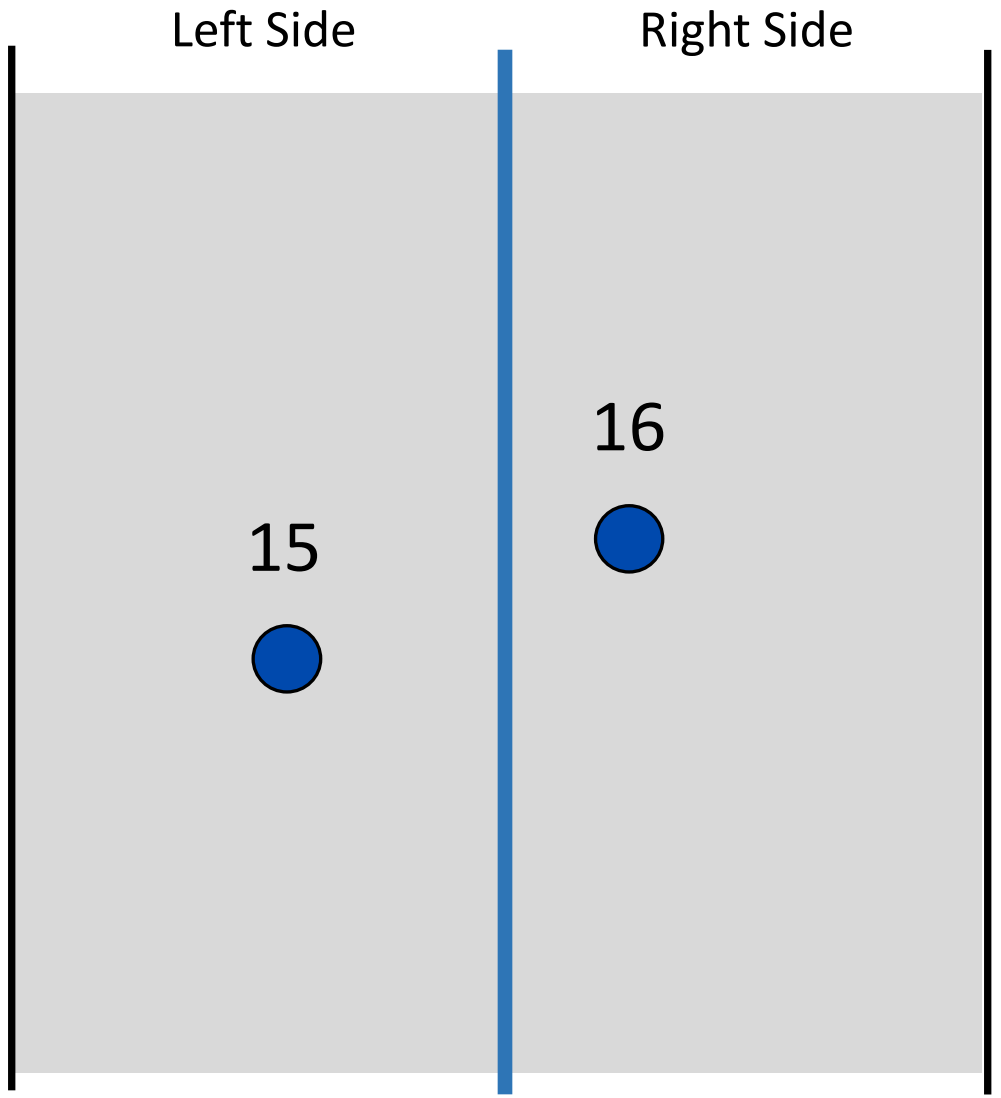
- Reorder based on Y-axis



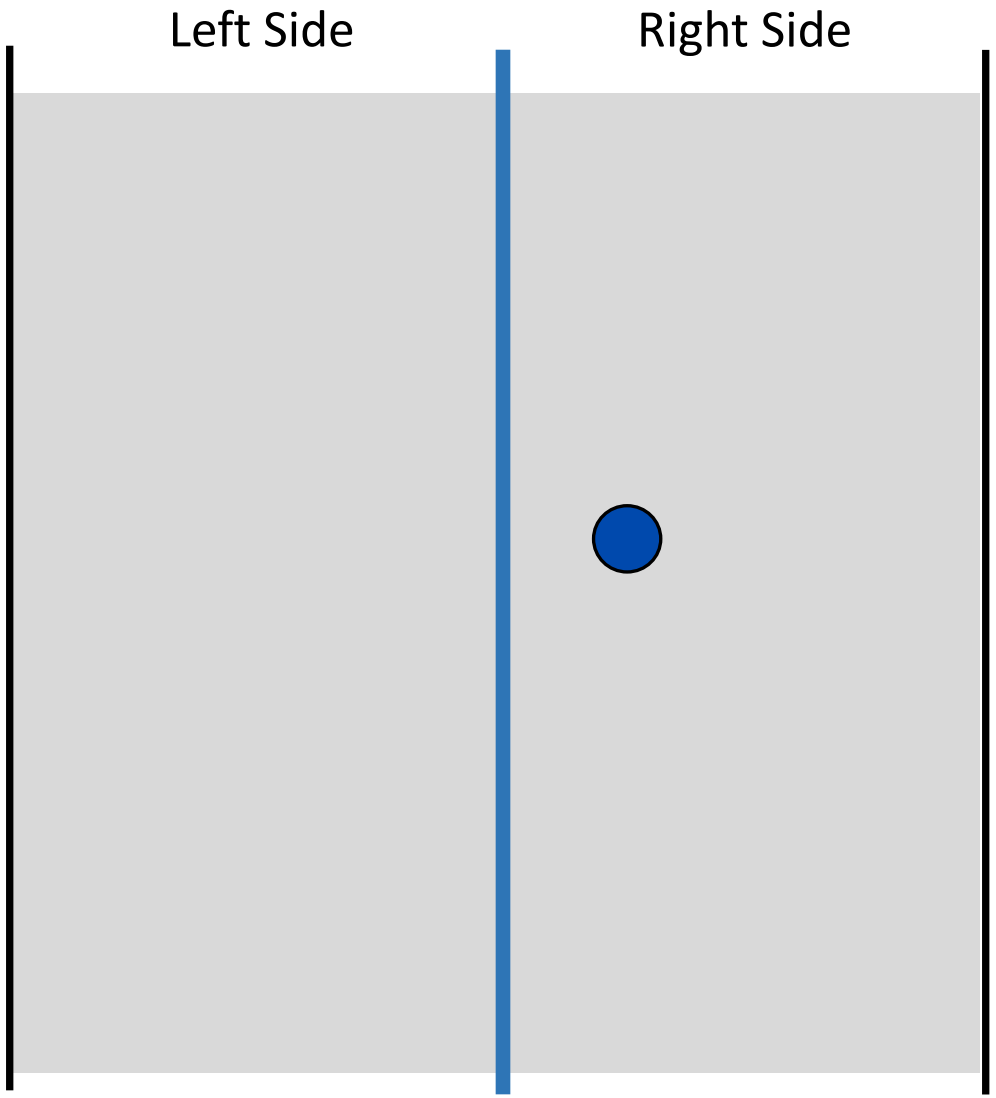
(16,15,1,14)

***d*= min(dl,dr)**

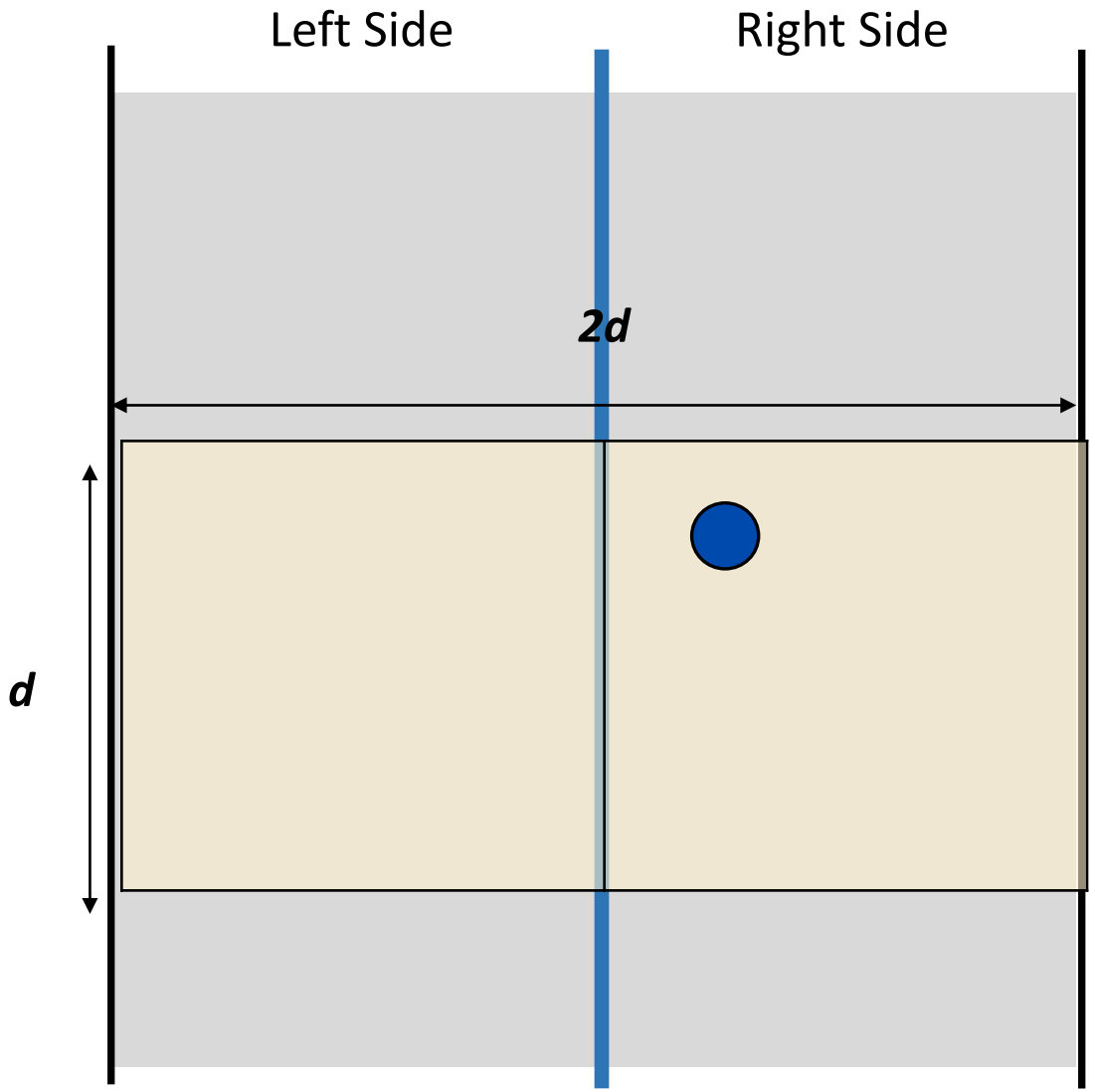
16
15
1
14



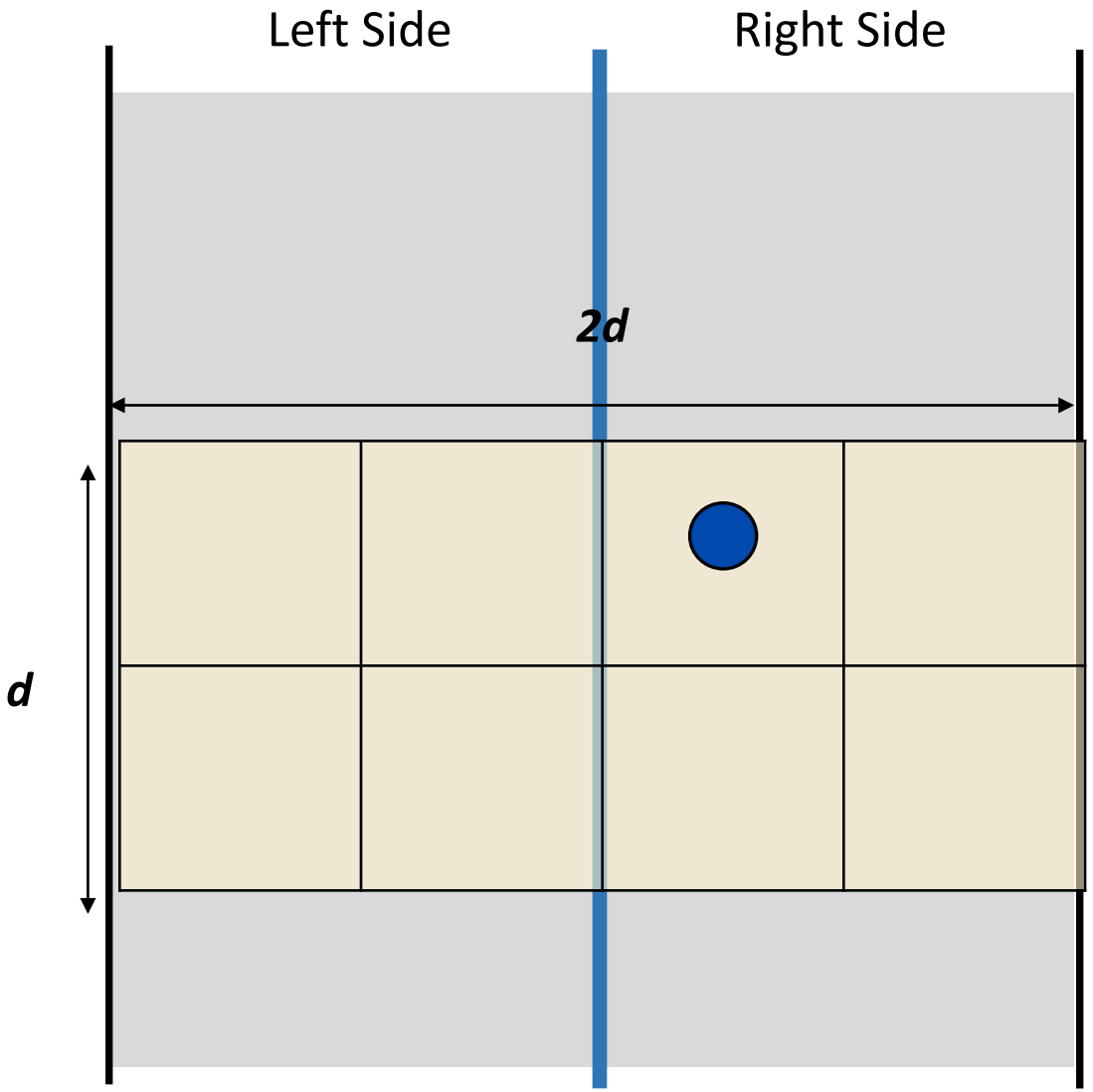
16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$



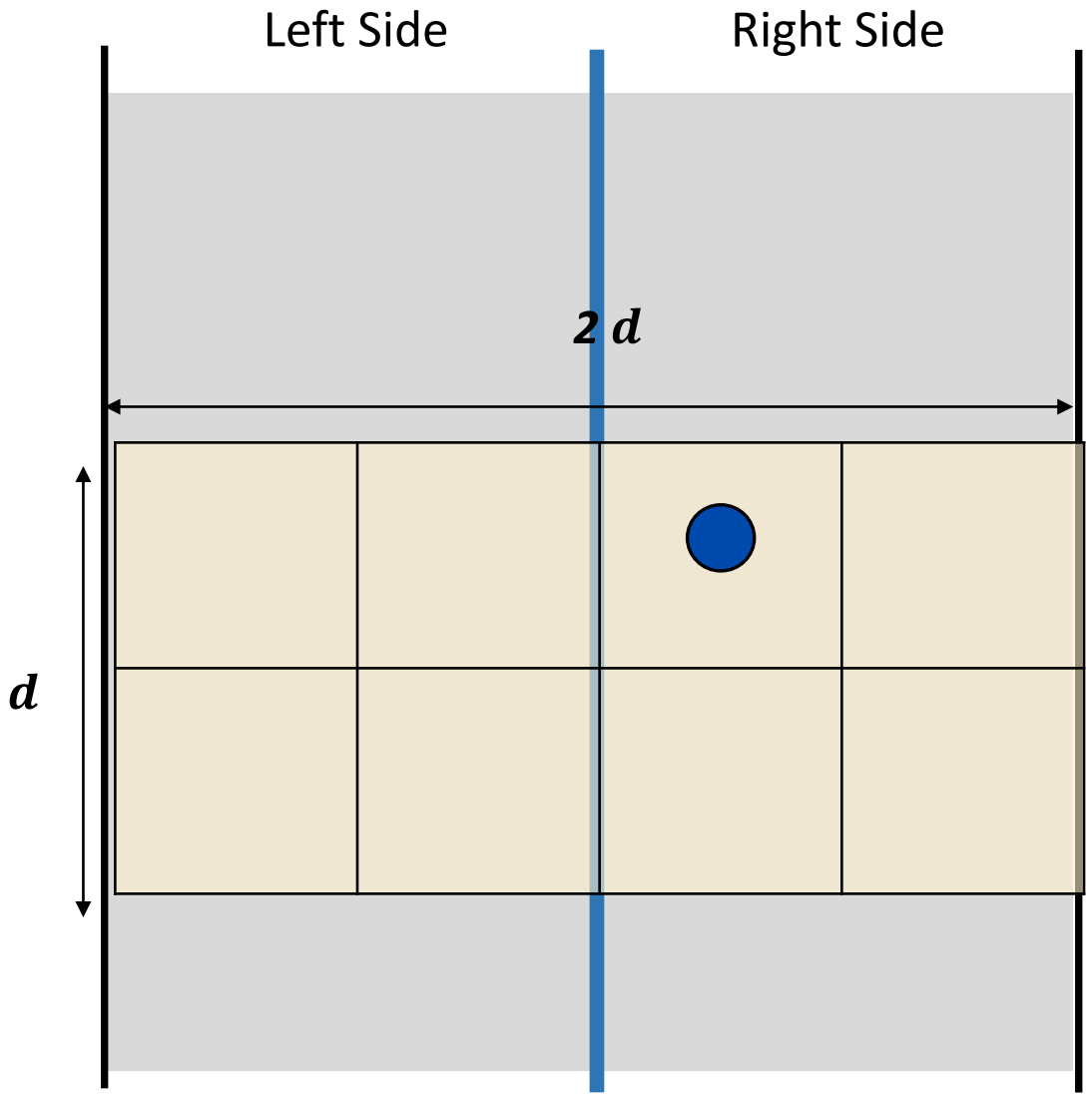
16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$

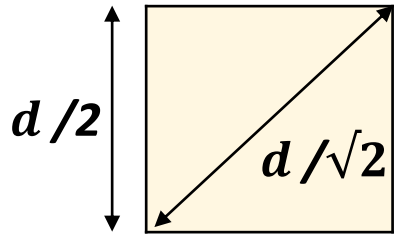


16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$



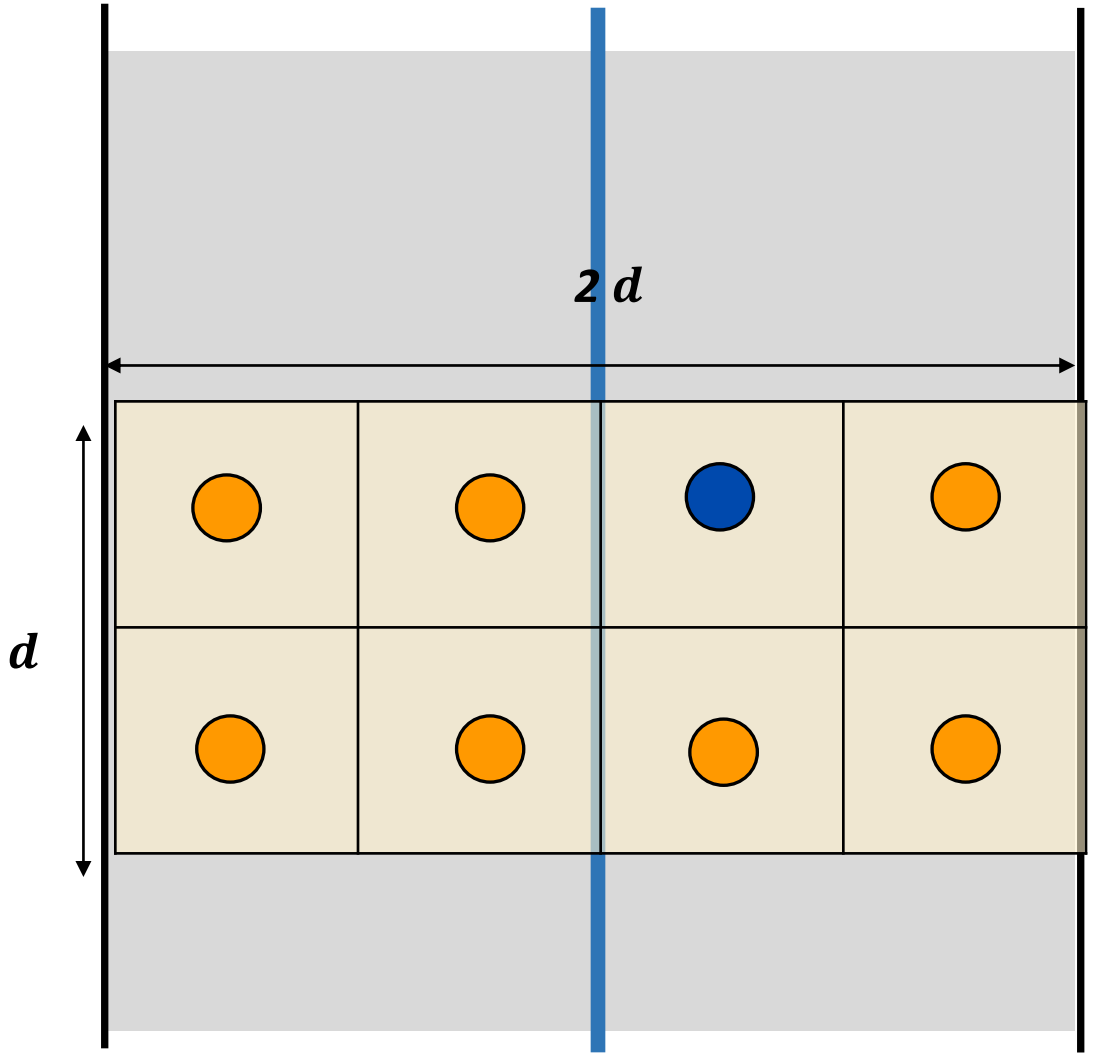
16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$

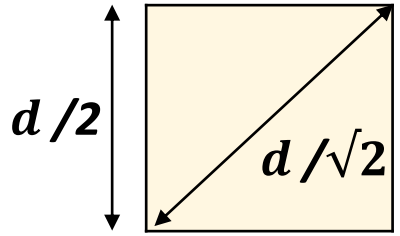




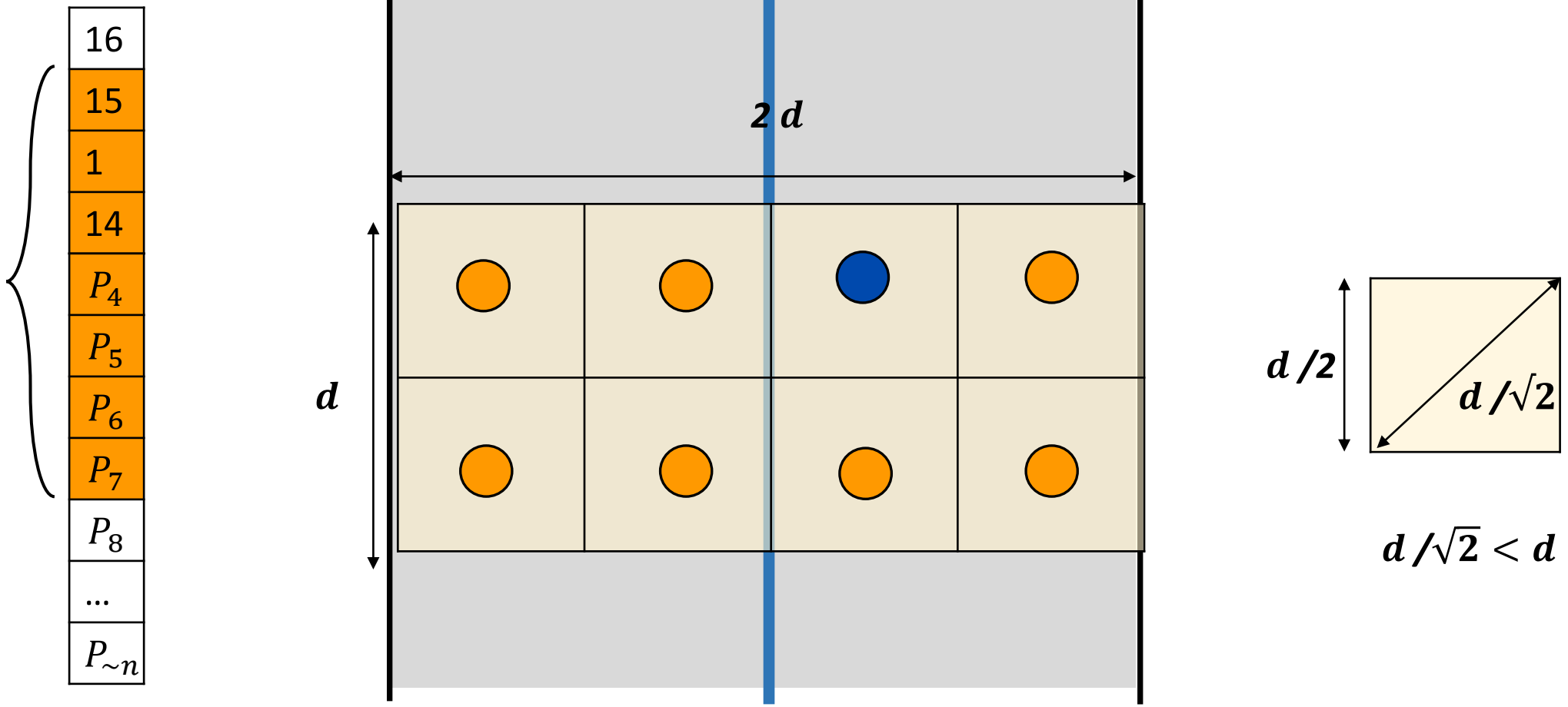
$$d/\sqrt{2} < d$$

16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$

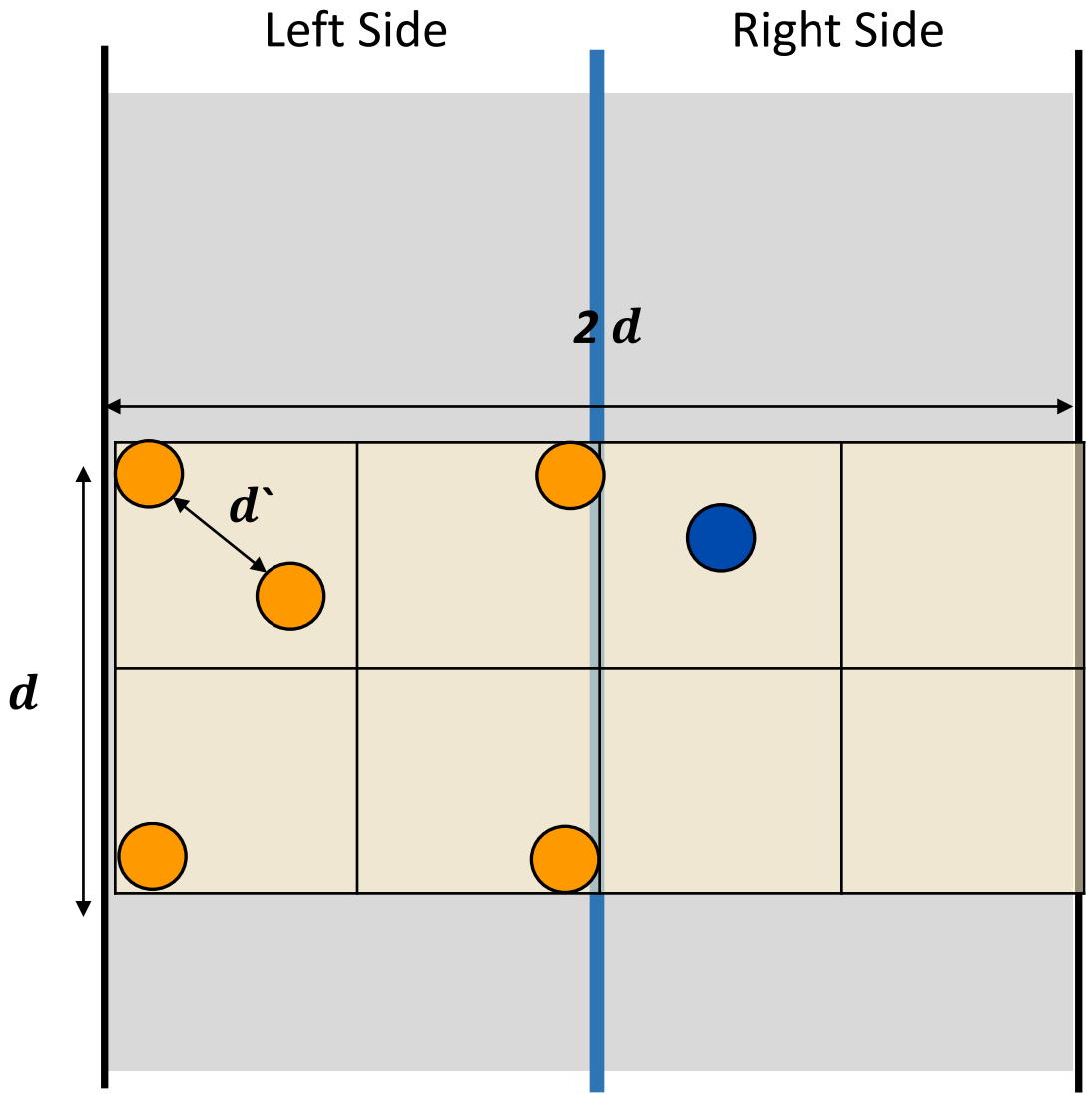




$$d/\sqrt{2} < d$$

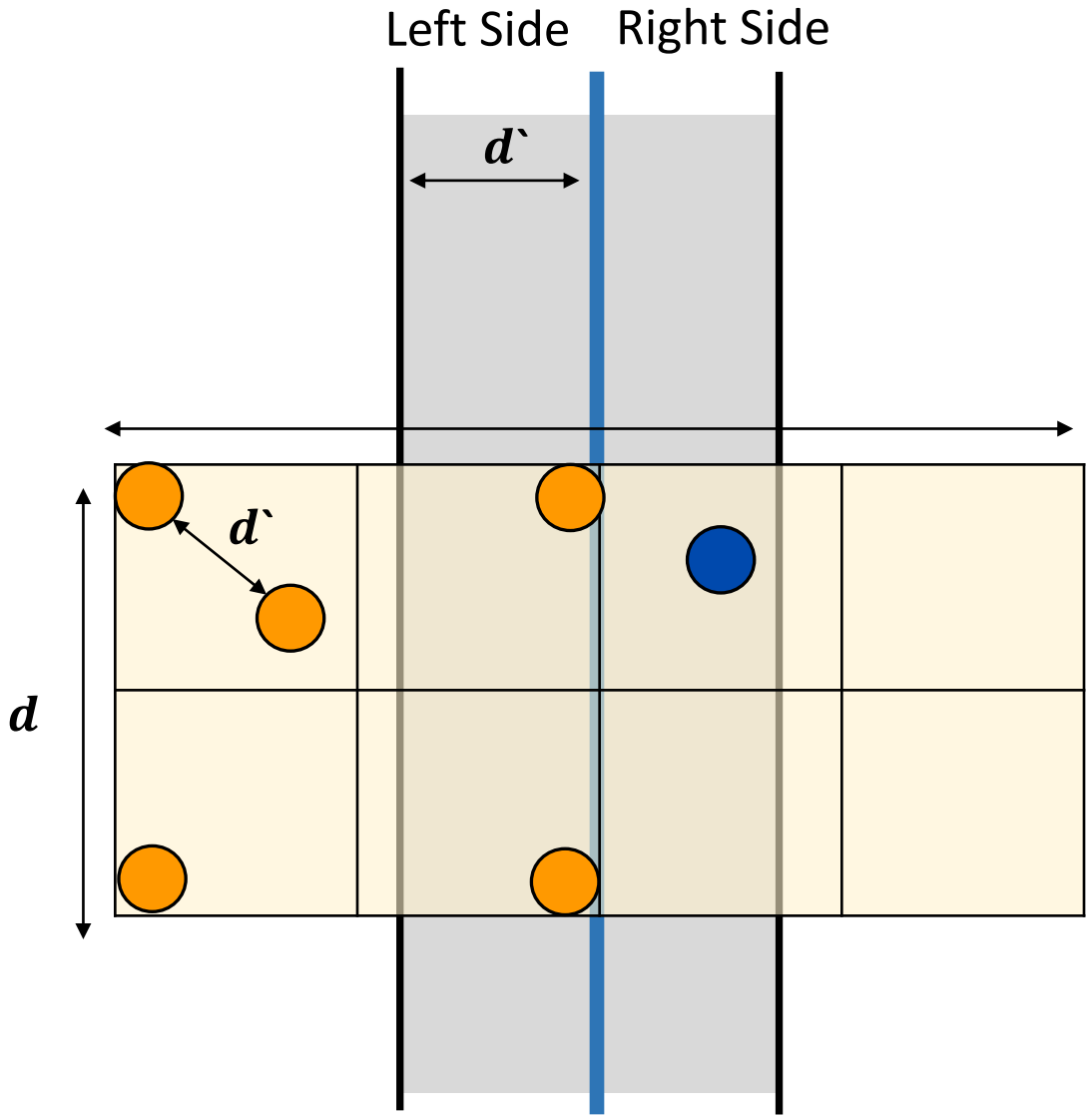

16
15
1
14
P_4
P_5
P_6
P_7
P_8
...
$P_{\sim n}$
Left Side
Right Side
$2\,d$
d
$d/2$
$d/\sqrt{2}$
$d/\sqrt{2} < d$

16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$

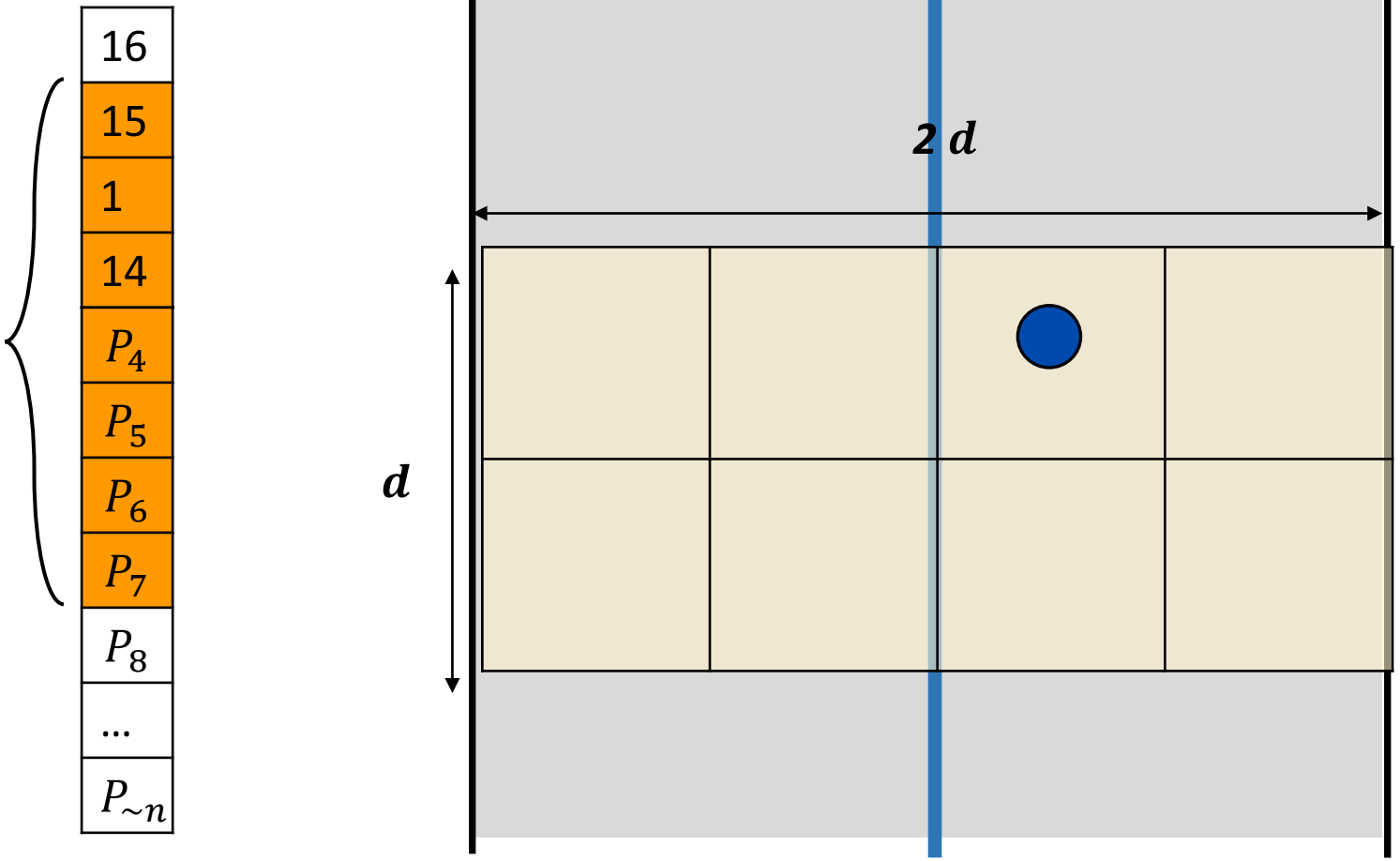


$$d` < d$$

16
15
1
14
P_4
P_5
P_6
P_7
P_8
…
$P_{\sim n}$



$$d` < d$$


16
15
1
14
P_4
P_5
P_6
P_7
P_8
...
$P_{\sim n}$
Left Side
Right Side
2 d
d

Closest Pair: Live Poll 1

What is the complexity of this algo?

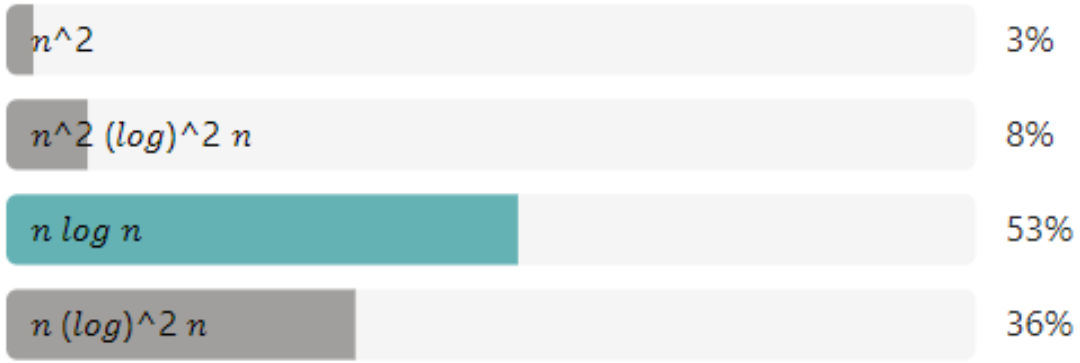
$A.\ n^2$
$B.\ n^2\ log^2 n$
$C.\ n\ log\ n$
$D.\ n\ log^2\ n$

Scan the QR code to vote or go to https://forms.office.com/r/zKJjPb1tCK

Closet Pair: Live Poll 1

Only people in my organization can respond, Record name

1. What is the complexity of this algo?



36 responses

1/1

Scan the QR code to vote or go to https://forms.office.com/r/zKJjPb1tCK

Finding Closest Point Algorithm

Sort points based on X-coordinates → $O(n \log n)$

CLOSEST-PAIR($p_1, p_2, \ldots, p_n$)

Compute vertical line L such that half the points are on each side of the line. → $O(n)$

$\delta_1 \leftarrow$ CLOSEST-PAIR(points in left half). → $T(n/2)$

$\delta_2 \leftarrow$ CLOSEST-PAIR(points in right half). → $T(n/2)$

$\delta \leftarrow \min \{ \delta_1 , \delta_2 \}$. → $O(1)$

$A \leftarrow$ list of all points closer than δ to line L. → $O(n)$

Sort points in A by y-coordinate. → $O(n \log n)$

Scan points in A in y-order and compare distance between each point and next 7 neighbors. → $O(n)$

If any of these distances is less than δ, update δ. → $O(1)$

RETURN δ.

Finding Closest Point Algorithm

$$O(n \log n) + O(log\ n).(O(n) + O(1) + O(n) + O(n\ log\ n) + O(n) + O(1))$$

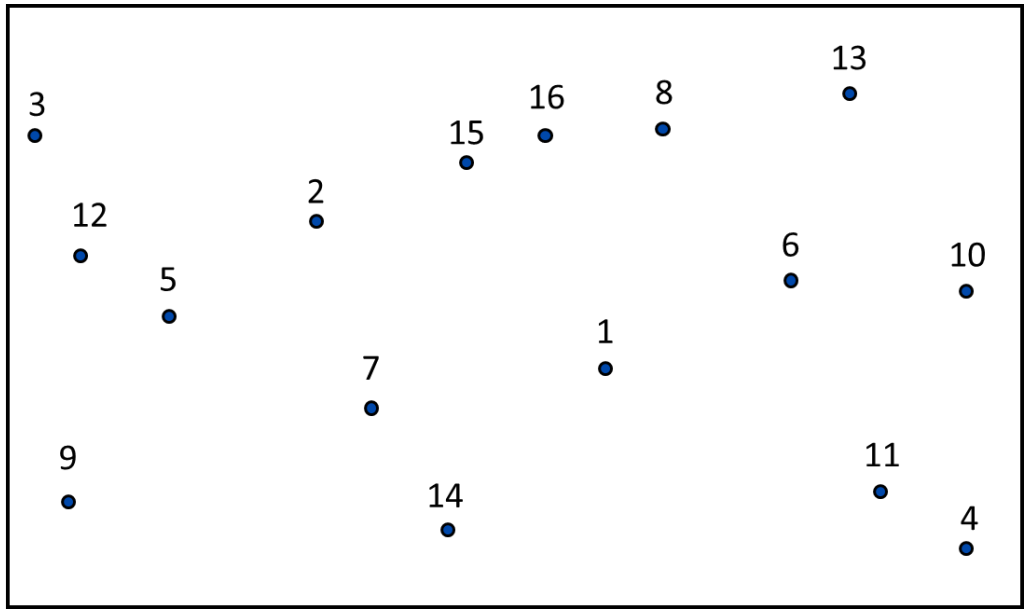
$$O\left(n\ log^2 n\right)$$

One Last Thing….

Can we bring it down to $n \log n$?

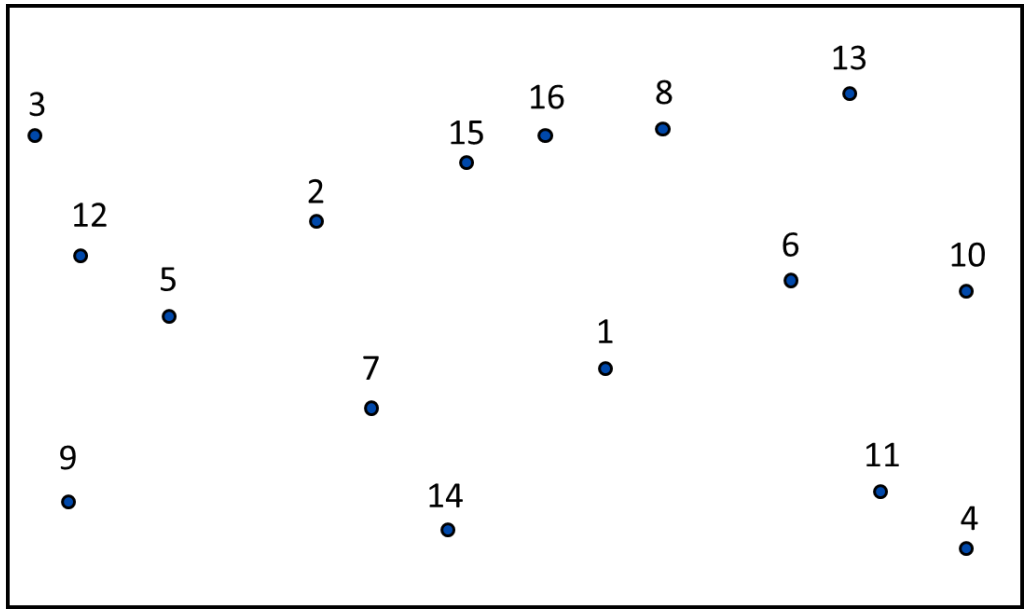
One Last Thing… Yup

Sorted X-axis



(3, 9, 12, 5, 2, 7, 15, 14, 16, 1, 8, 6, 13, 11, 10, 4)

Sorted Y-axis



(13, 8, 16, 3, 15, 2, 12, 6, 10, 5, 1, 7, 9, 11, 14, 4)

Sort points based on X-coordinates → $O(n \log n)$

CLOSEST-PAIR($p_1, p_2, \ldots, p_n$)

Compute vertical line L such that half the points are on each side of the line. → $O(n)$

$\delta_1 \leftarrow$ CLOSEST-PAIR(points in left half). → $T(n/2)$

$\delta_2 \leftarrow$ CLOSEST-PAIR(points in right half). → $T(n/2)$

$\delta \leftarrow \min \{ \delta_1 , \delta_2 \}$. → $O(1)$

$A \leftarrow$ list of all points closer than δ to line L. → $O(n)$

Sort points in A by y-coordinate. → $O(n \log n)$

Scan points in A in y-order and compare distance between each point and next 7 neighbors. → $O(n)$

If any of these distances is less than δ, update δ. → $O(1)$

RETURN δ.

Sort points based on X-coordinates → $O(n \log n)$

Sort points based on Y-coordinates → $O(n \log n)$

CLOSEST-PAIR($p_1, p_2, \ldots, p_n$)

Compute vertical line L such that half the points are on each side of the line. → $O(n)$

$\delta_1 \leftarrow$ CLOSEST-PAIR(points in left half). → $T(n/2)$

$\delta_2 \leftarrow$ CLOSEST-PAIR(points in right half). → $T(n/2)$

$\delta \leftarrow \min \{ \delta_1 , \delta_2 \}$. → $O(1)$

$A \leftarrow$ list of all points closer than δ to line L. → $O(n)$

Sort points in A by y-coordinate. → $O(n \log n)$

Scan points in A in y-order and compare distance between each point and next 7 neighbors. → $O(n)$

If any of these distances is less than δ, update δ. → $O(1)$

RETURN δ.

Sort points based on X-coordinates → $O(n \log n)$

Sort points based on Y-coordinates → $O(n \log n)$

CLOSEST-PAIR($p_1, p_2, \ldots, p_n$)

Compute vertical line L such that half the points are on each side of the line. → $O(n)$

$\delta_1 \leftarrow$ CLOSEST-PAIR(points in left half). → $T(n/2)$

$\delta_2 \leftarrow$ CLOSEST-PAIR(points in right half). → $T(n/2)$

$\delta \leftarrow \min \{ \delta_1 , \delta_2 \}$. → $O(1)$

$A \leftarrow$ list of all points closer than δ to line L. → $O(n)$

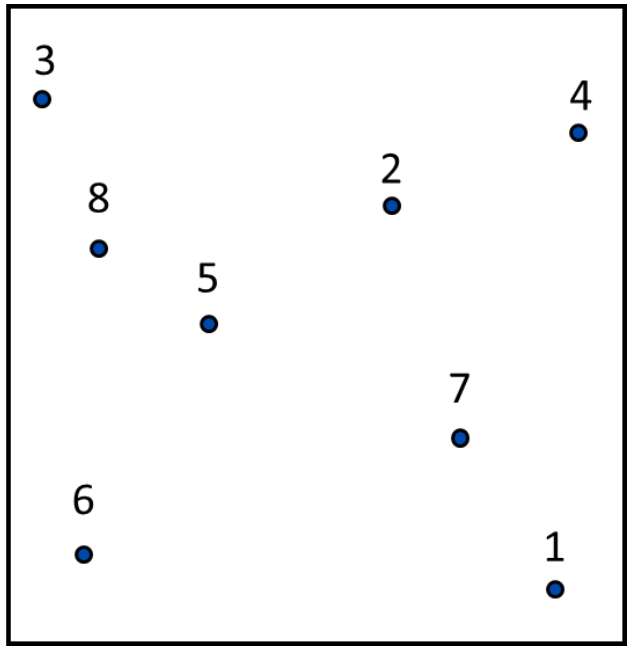
Sort points in A by y-coordinate. → $O(n)$

Scan points in A in y-order and compare distance between each point and next 7 neighbors. → $O(n)$

If any of these distances is less than δ, update δ. → $O(1)$

RETURN δ.

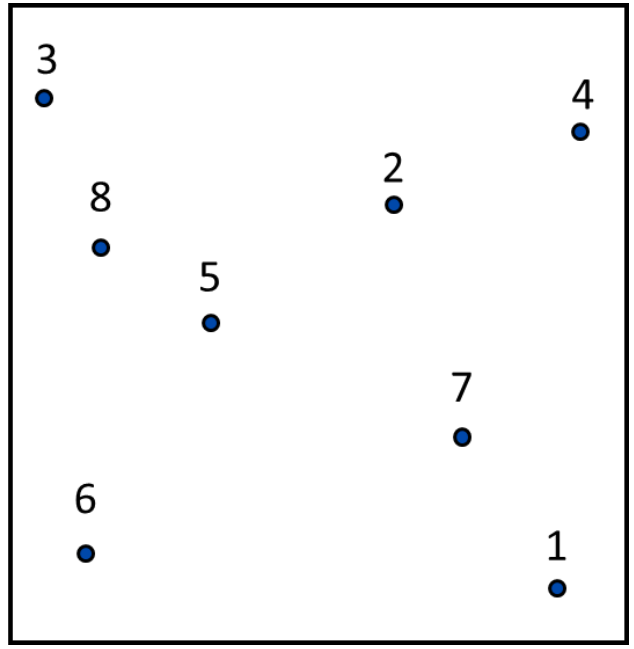
Sorted X-axis



(3 , 6 , 8 , 5 , 2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5),(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

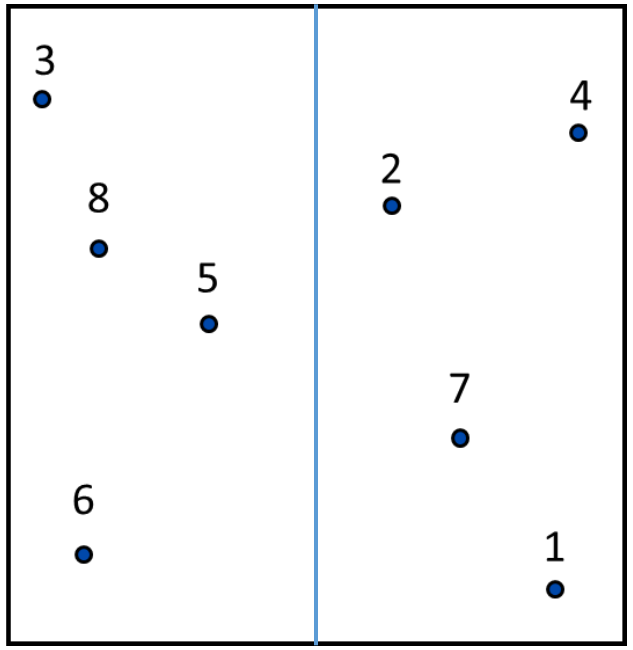
Sorted Y-axis



(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

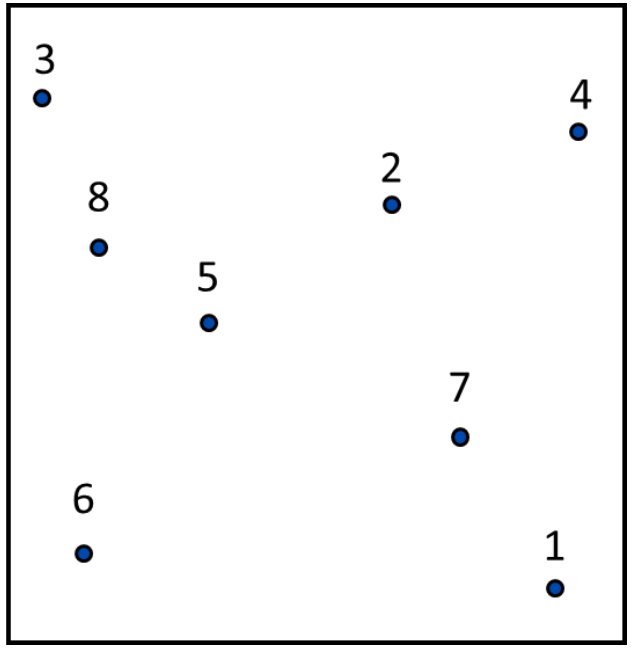
Sorted X-axis



(3 , 6 , 8 , 5 , 2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5),(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

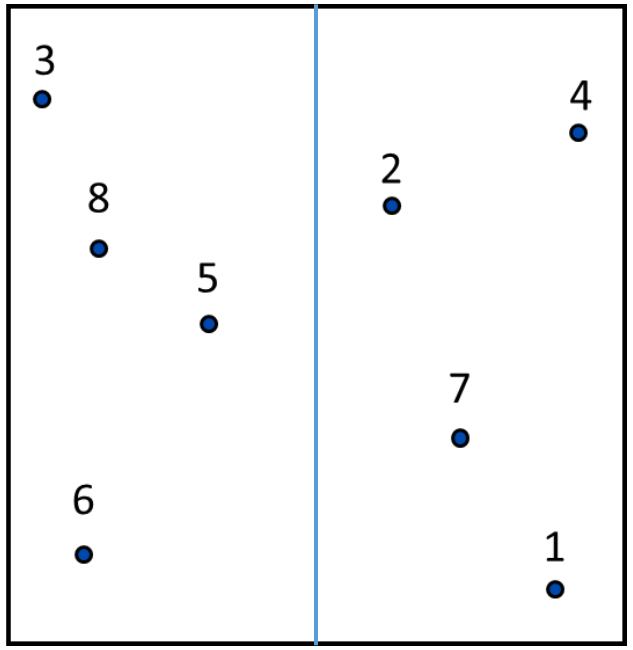
Sorted Y-axis



(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

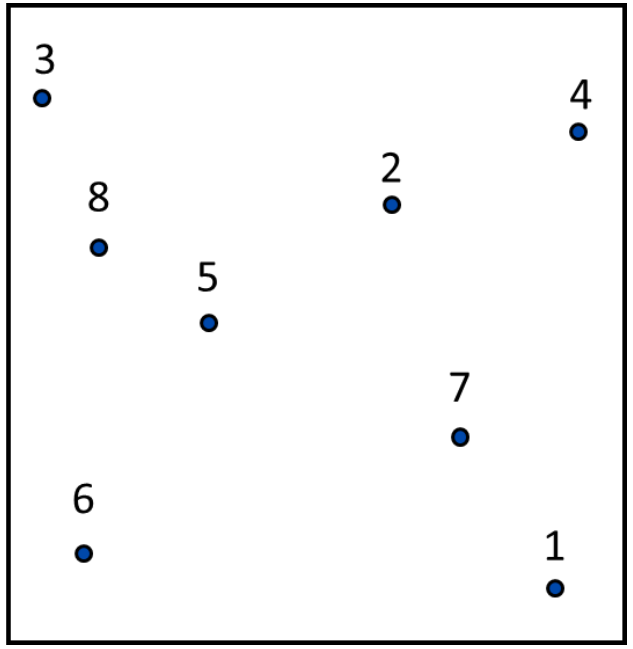
Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

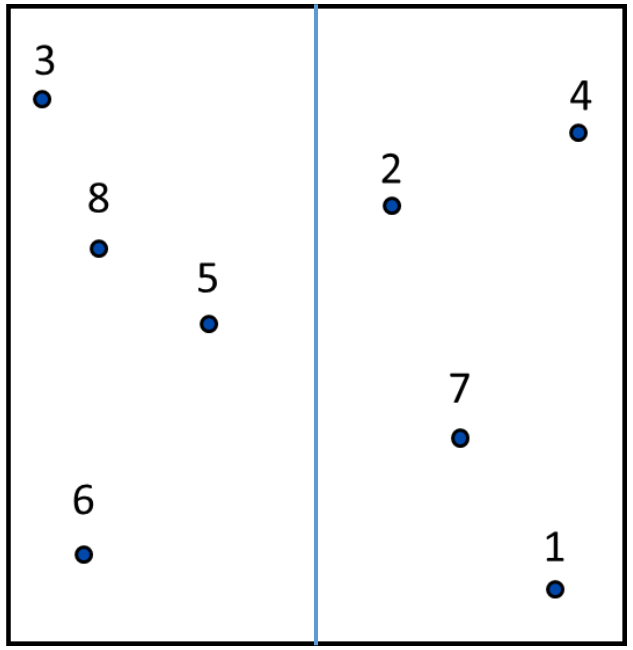
Sorted Y-axis



(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

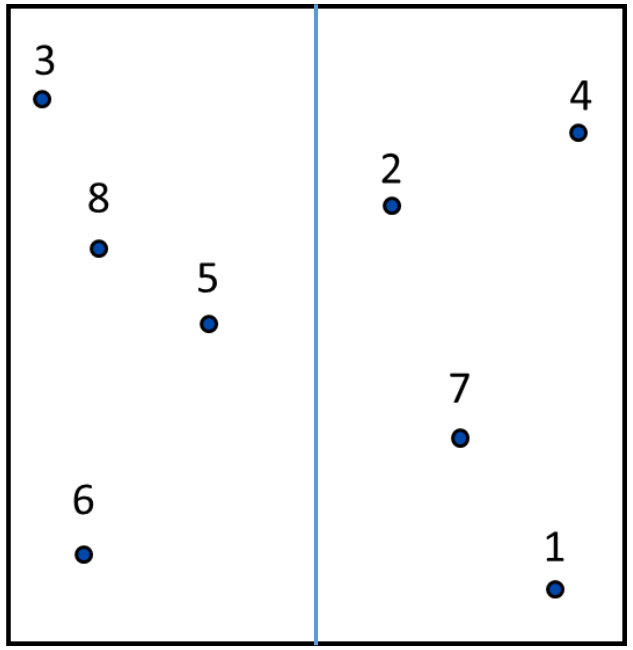
(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate =***< 4)

else ()

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

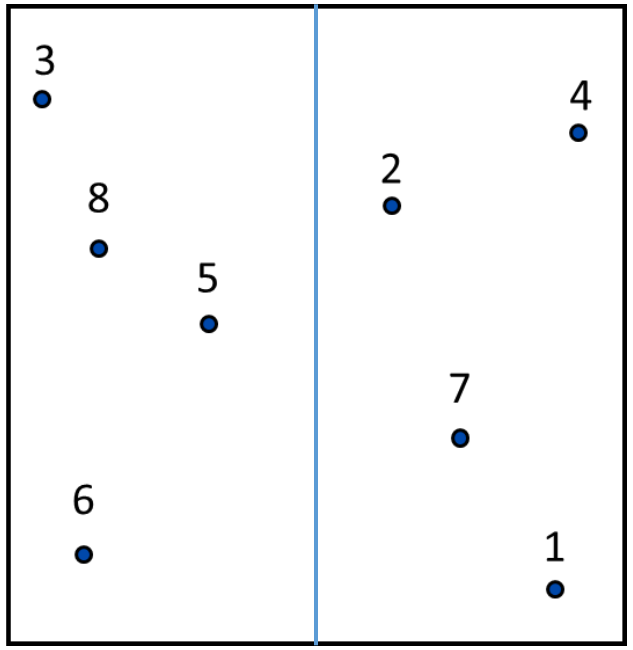
If (***X_coordinate =***< 4)

else ()

1

(8,**1**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate*** =< 4)

6

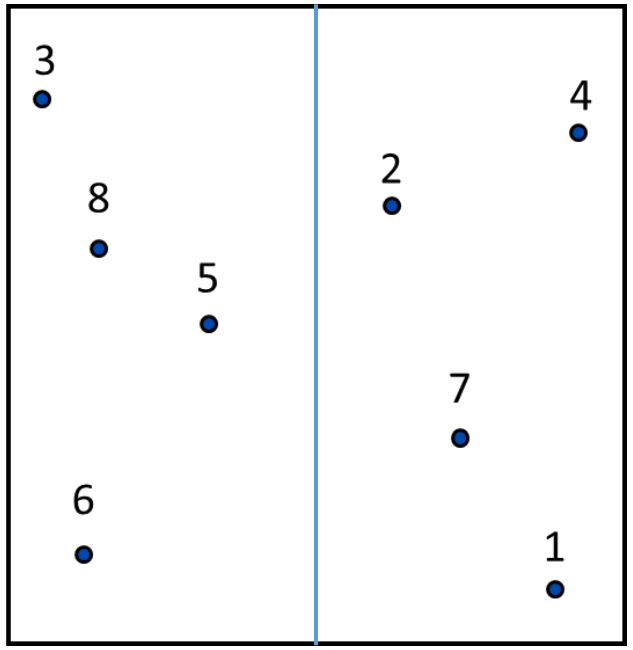
(2,**2**)

else ()

1

(8,**1**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate*** =< 4)

6

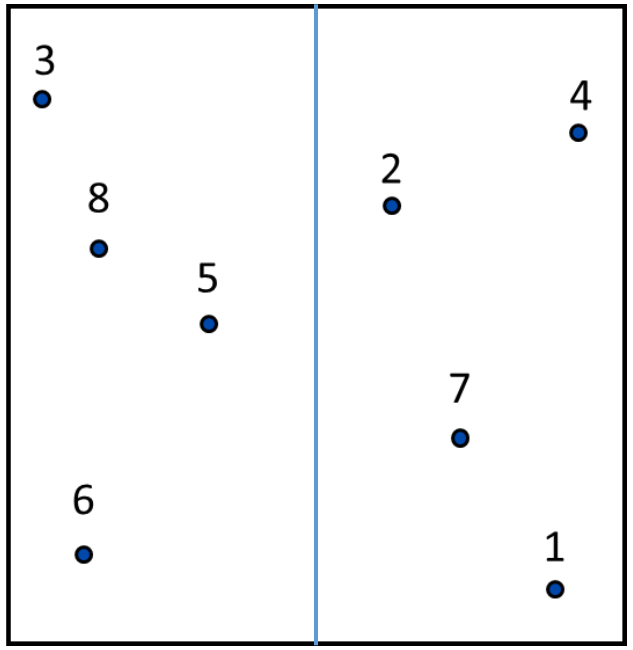
(2,**2**)

else ()

1 7

(8,**1**) (7,**4**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate*** =< 4)

6 5

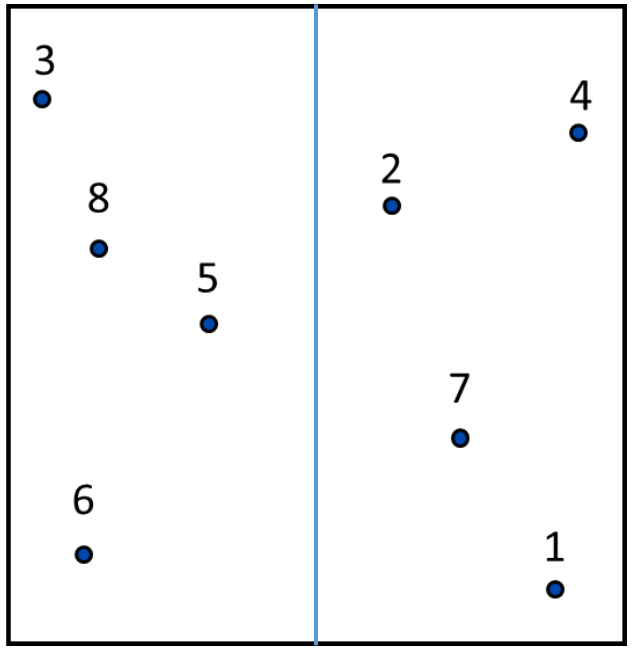
(2,**2**) (4,**5**)

else ()

1 7

(8,**1**) (7,**4**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate*** =< 4)

6 5 8

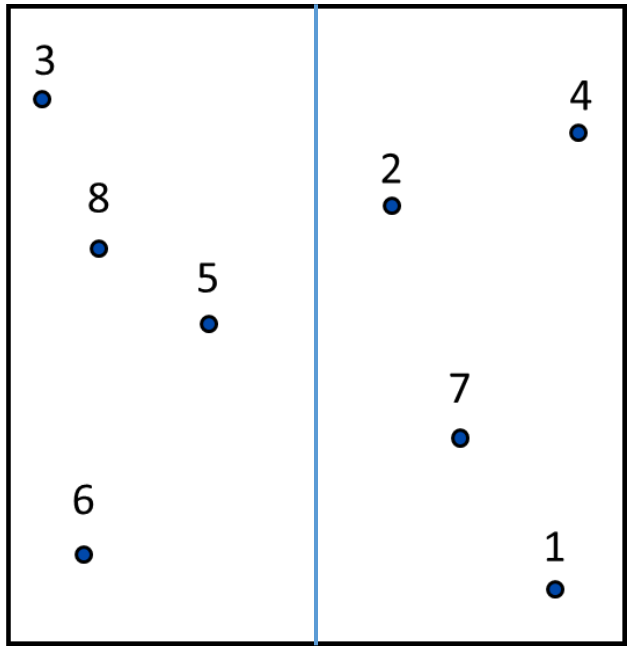
(2,**2**) (4,**5**) (3,**6**)

else ()

1 7

(8,**1**) (7,**4**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate =<*** 4)

6 5 8

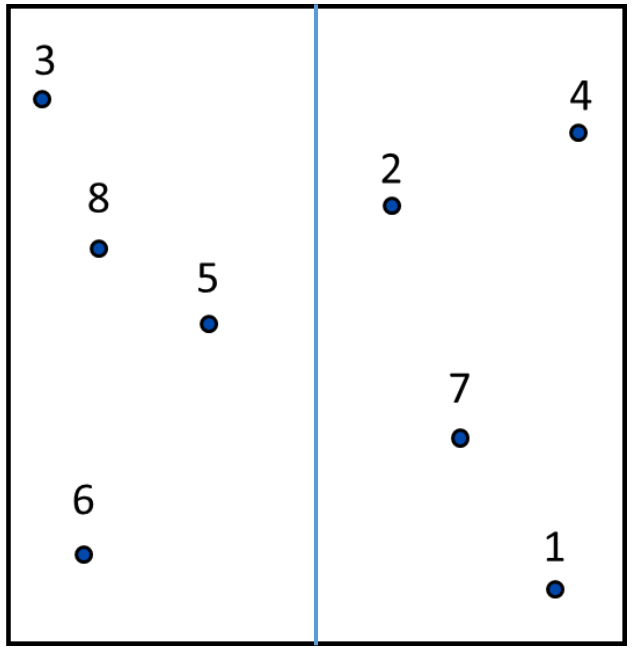
(2,**2**) (4,**5**) (3,**6**)

else ()

1 7 2

(8,**1**) (7,**4**) (5,**6**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate*** =< 4)

6 5 8

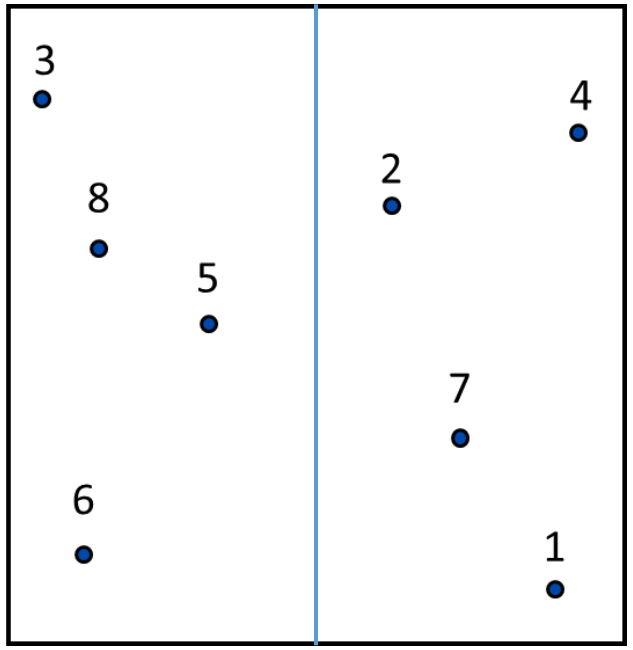
(2,**2**) (4,**5**) (3,**6**)

else ()

1 7 2 4

(8,**1**) (7,**4**) (5,**6**) (8,**7**)

Sorted X-axis



(3 , 6 , 8 , 5) (2 , 7 , 1 , 4)

{(**1**,8),(**2**,2),(**3**,4),(**4**,5)} {(**5**,6),(**7**,4),(**8**,1),(**8**,7)}

(1 , 6 , 7 , 5 , 8 , 2 , 4 , 3)

{(8,**1**),(2,**2**),(7,**4**),(4,**5**),(3,**6**),(5,**6**),(8,**7**),(1,**8**)}

If (***X_coordinate*** =< 4)

6 5 8 3

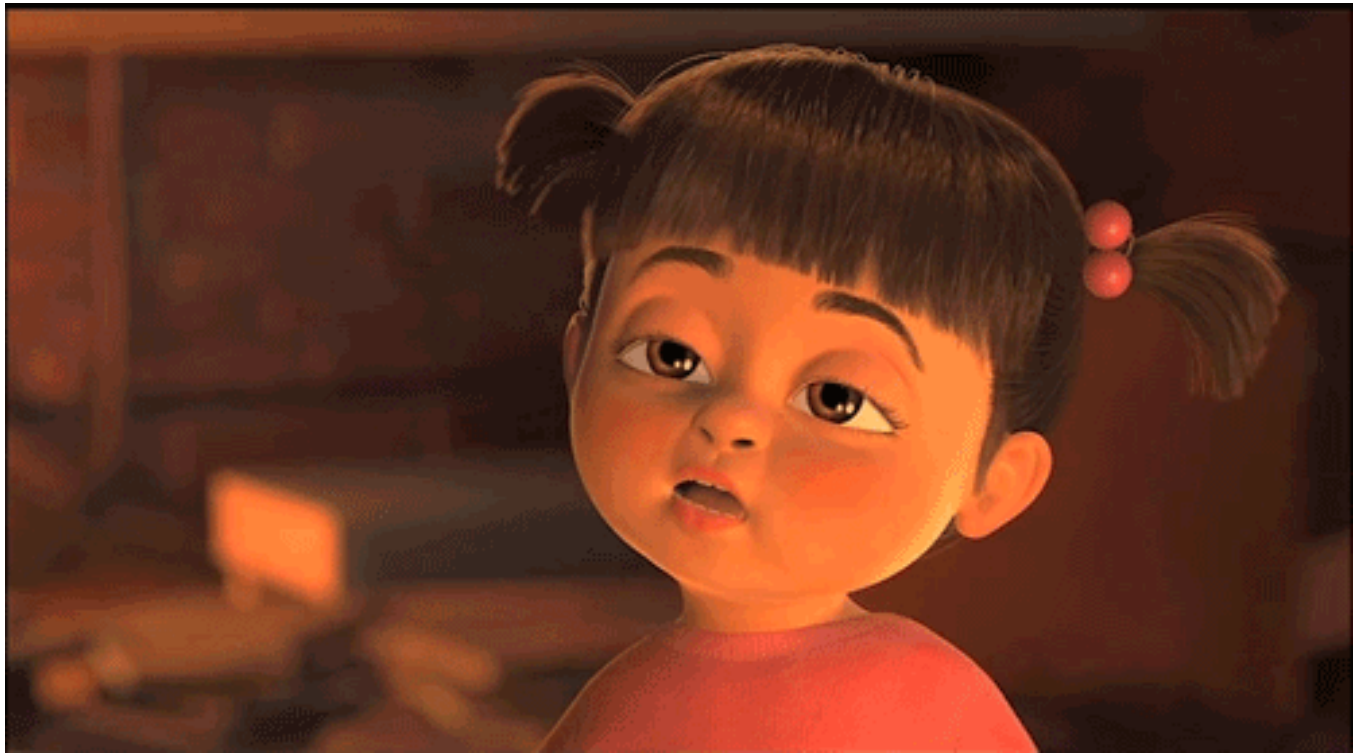
(2,**2**) (4,**5**) (3,**6**) (1,**8**)

else ()

1 7 2 4

(8,**1**) (7,**4**) (5,**6**) (8,**7**)

Thanks a lot



If you are taking a Nap, **wake up**.....Lecture Over