Closest-Pair Problem: Divide and Conquer

- Brute force approach requires comparing every point with every other point
- Given n points, we must perform 1 + 2 + 3 + … + n-2 + n-1 comparisons.

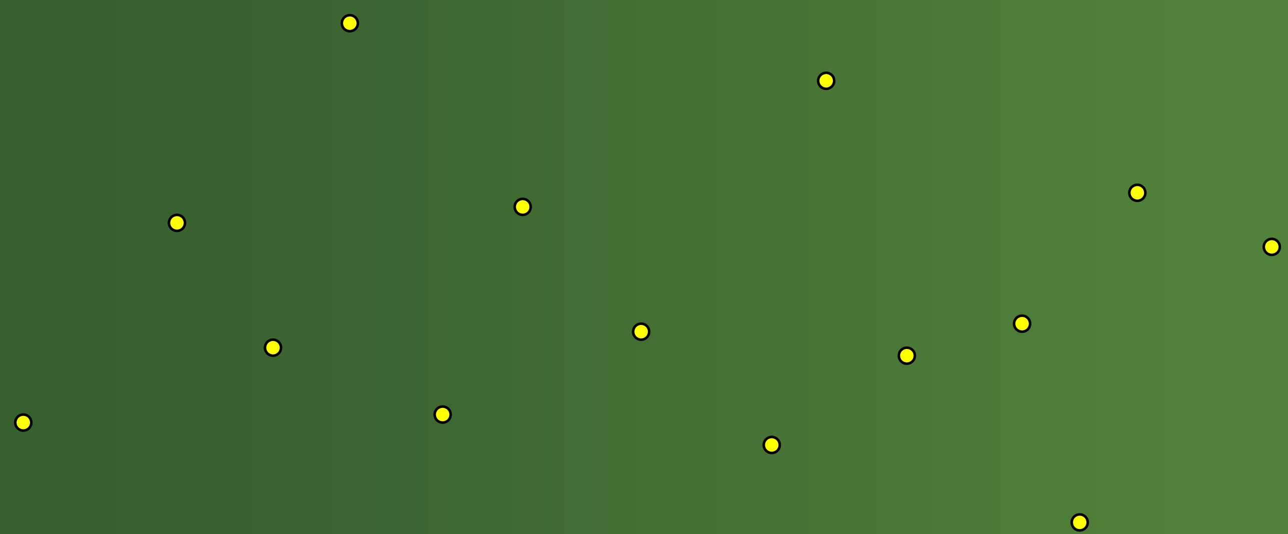
$$\sum_{k=1}^{n-1} k = \frac{(n-1)\cdot n}{2}$$

$$d(A,B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- Brute force → O(n^2)
- The Divide and Conquer algorithm yields → O(n log n)
- Reminder: if n = 1,000,000 then
  - n^2 = 1,000,000,000,000 whereas
  - n log n = 20,000,000

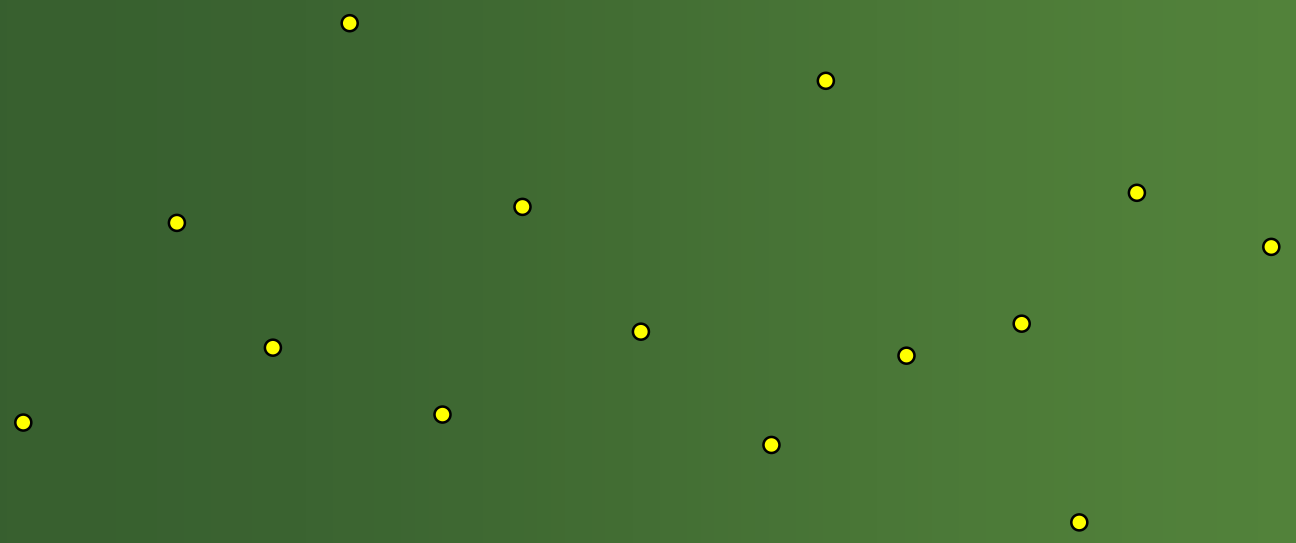
Closest-Pair Algorithm

Given: A set of points in 2-D



Closest-Pair Algorithm

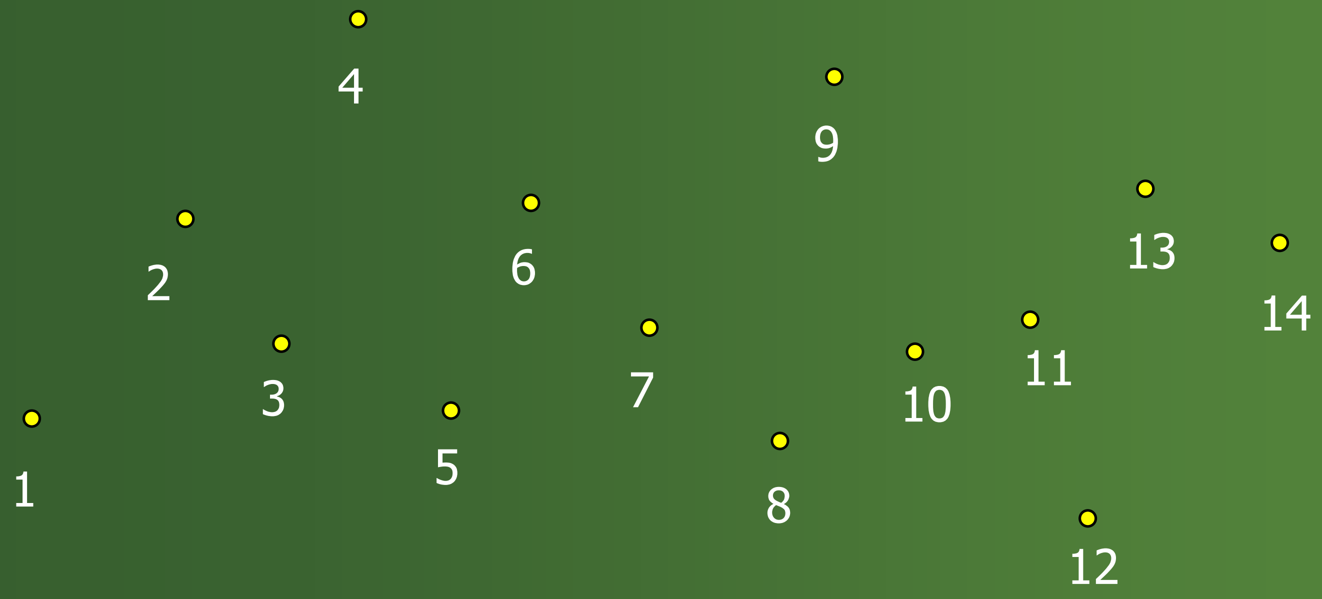
Step 1: Sort the points in one D



Closest-Pair Algorithm

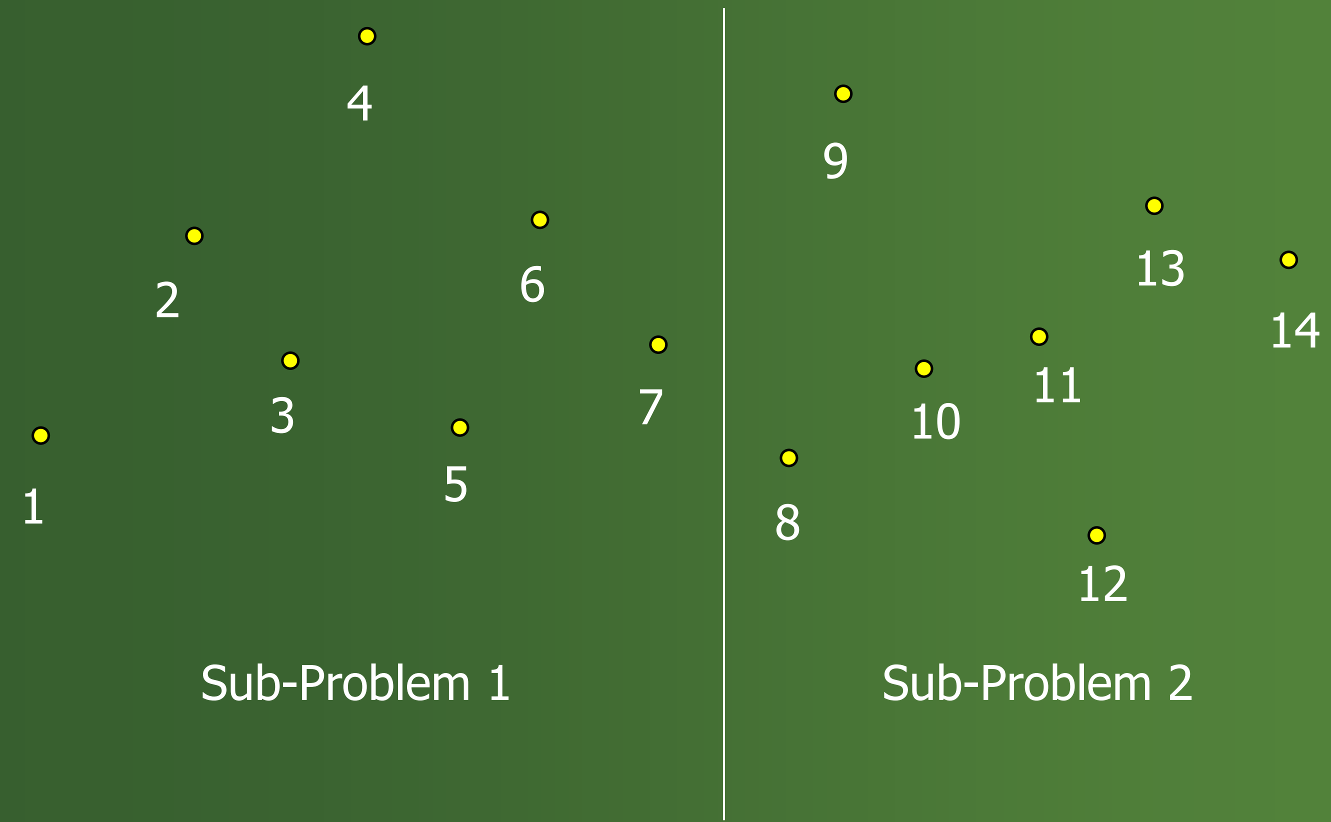
Lets sort based on the X-axis

O(n log n) using quicksort or mergesort



Closest-Pair Algorithm

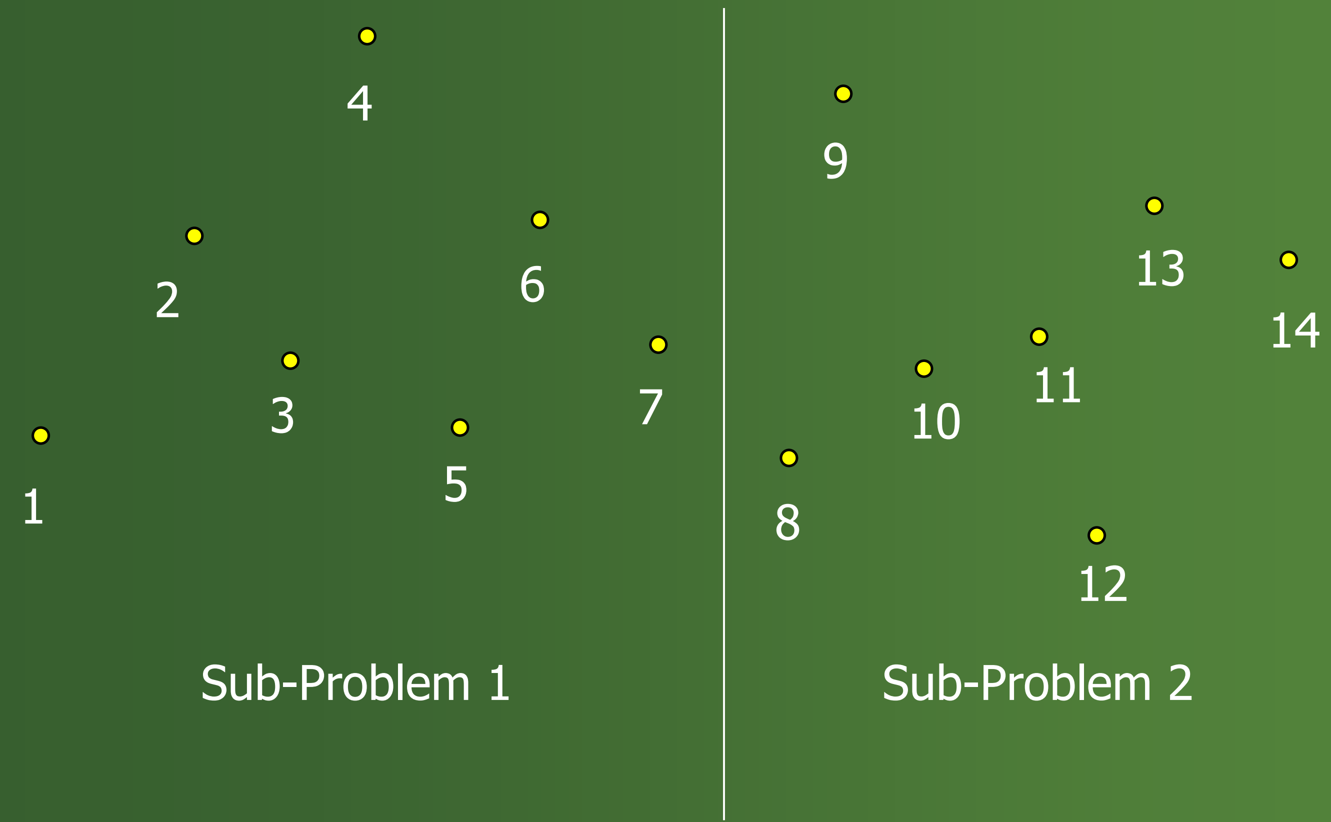
Step 2: Split the points, i.e.,
Draw a line at the mid-point between 7 and 8



Closest-Pair Algorithm

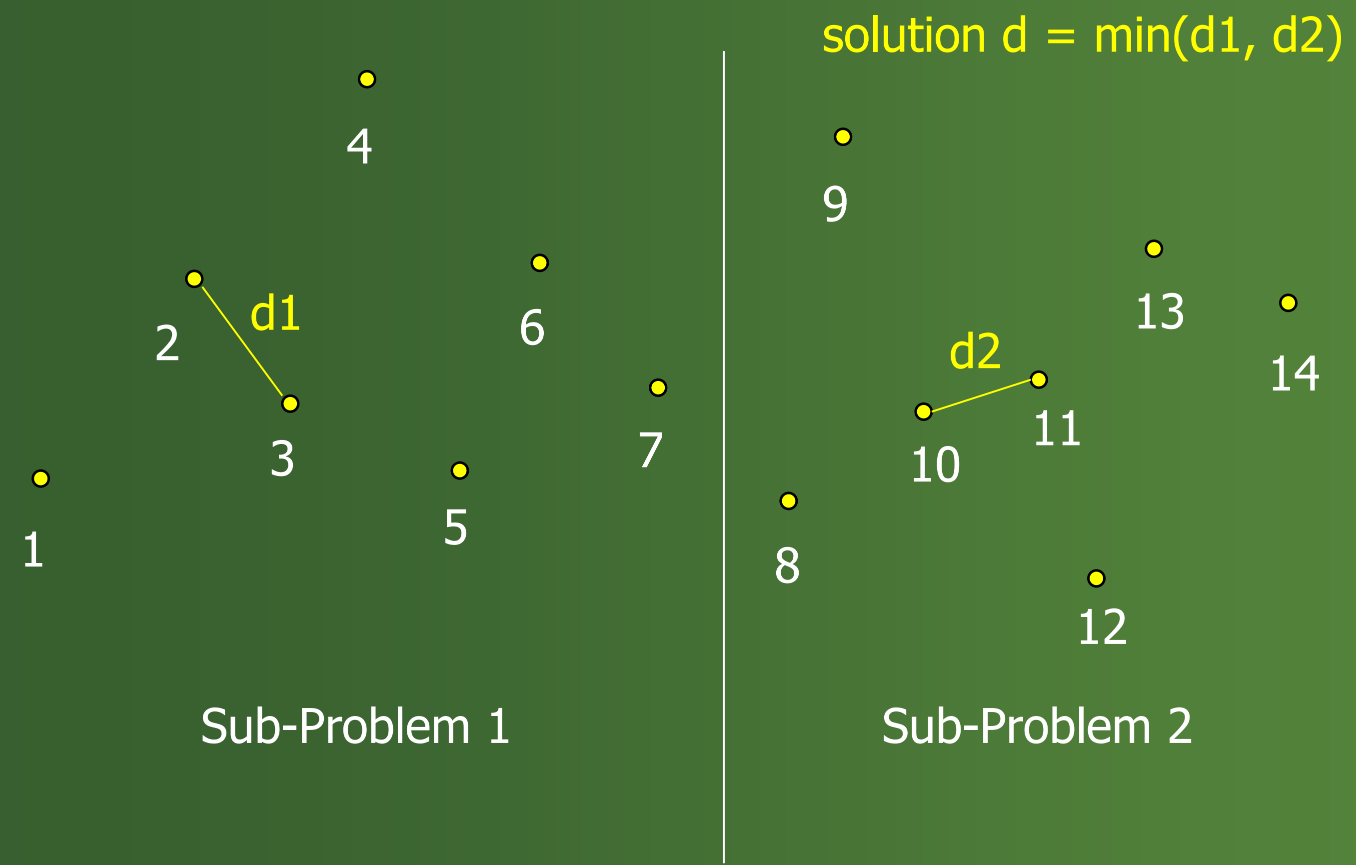
Advantage: Normally, we'd have to compare each of the 14 points with every other point.

(n-1)n/2 = 13*14/2 = **91 comparisons**



Closest-Pair Algorithm

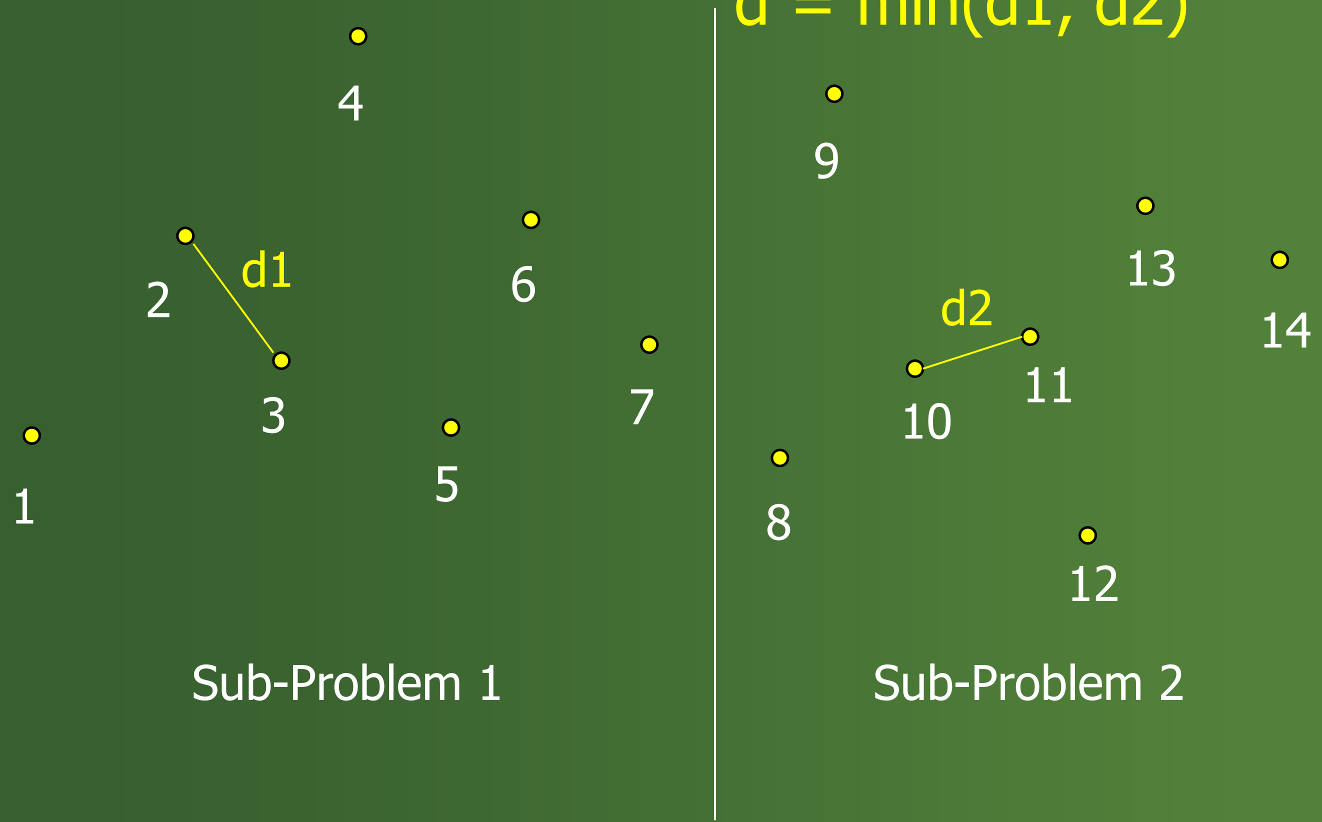
Advantage: Now, we have two sub-problems of half the size. Thus, we have to do 6*7/2 comparisons twice, which is 42 comparisons



Closest-Pair Algorithm

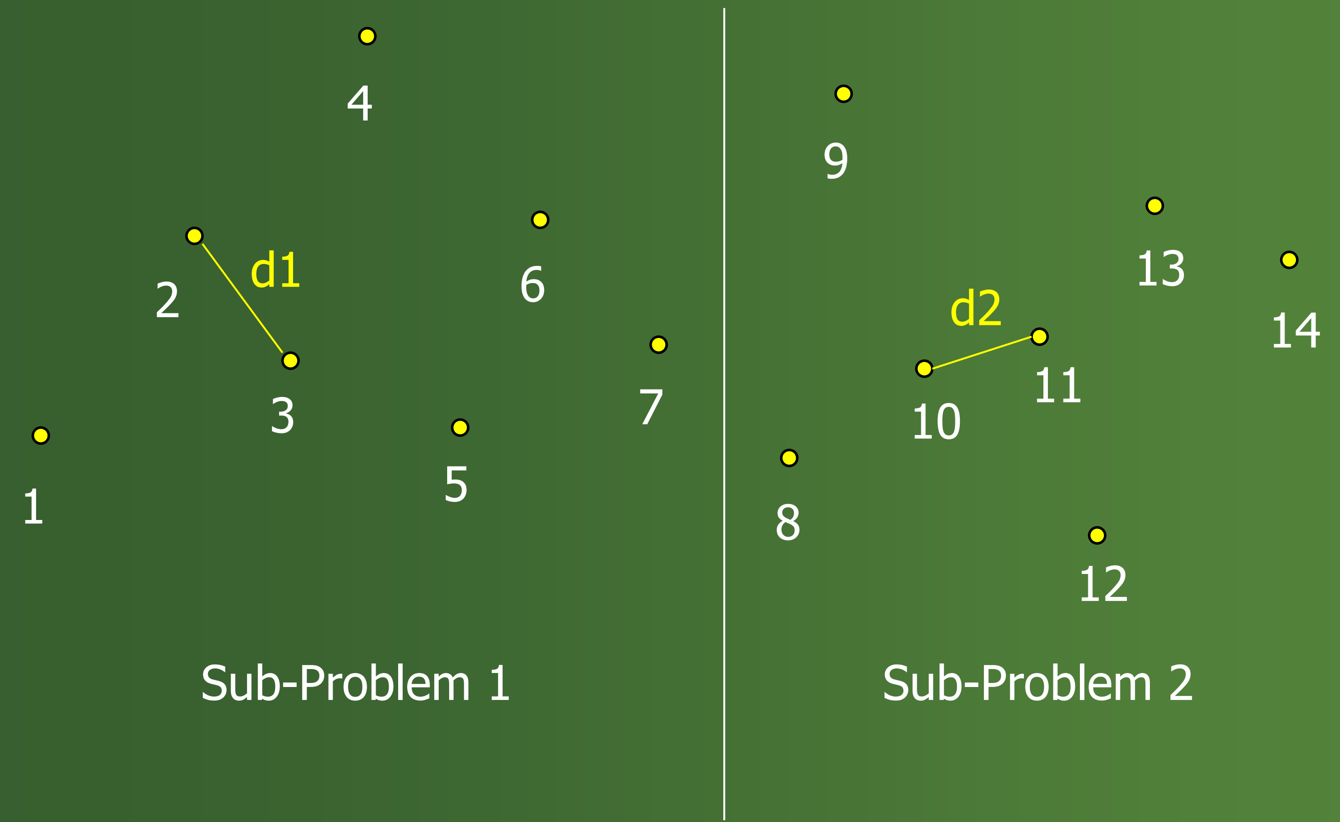
Advantage: With just one split we cut the number of comparisons in half. Obviously, we gain an even greater advantage if we split the sub-problems.

d = min(d1, d2)



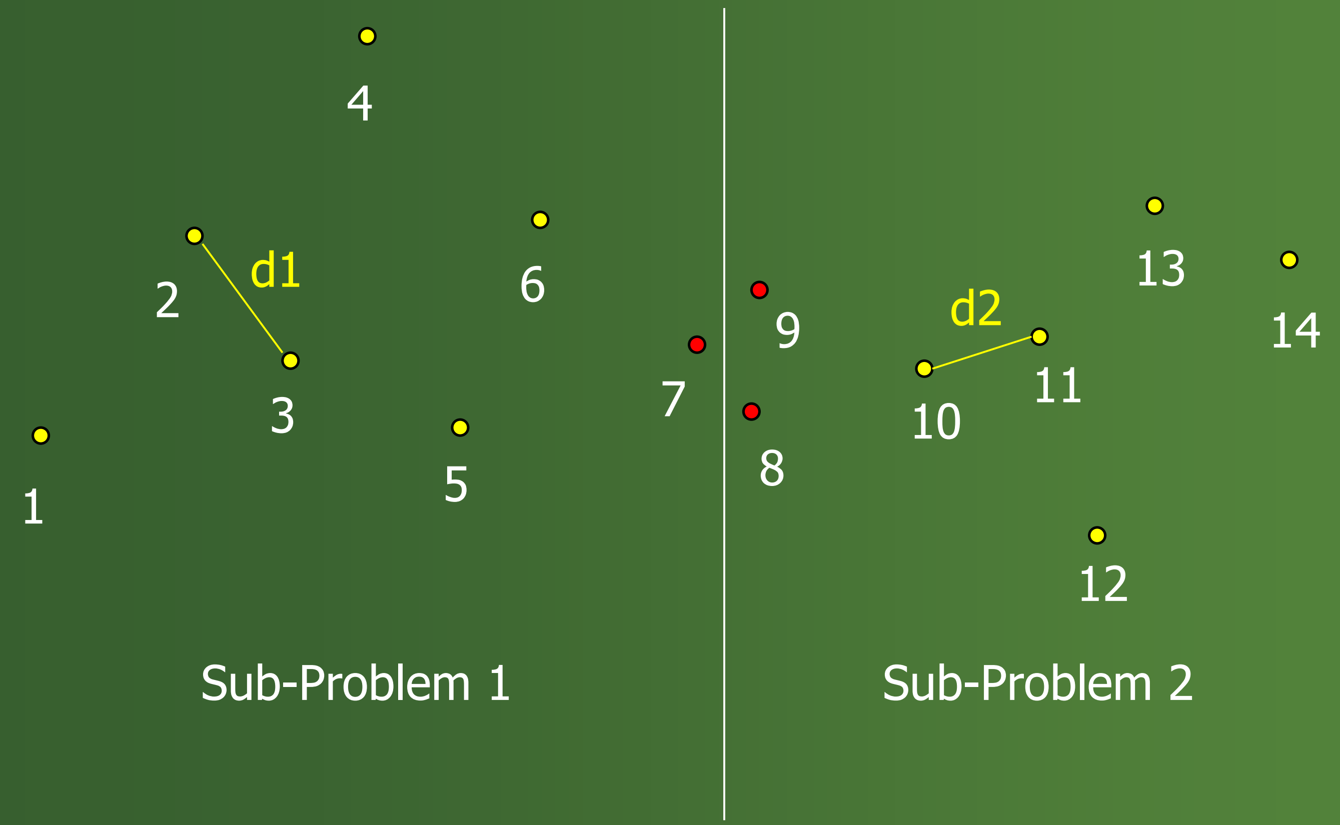
Closest-Pair Algorithm

Problem: However, what if the closest two points are each from different sub-problems?



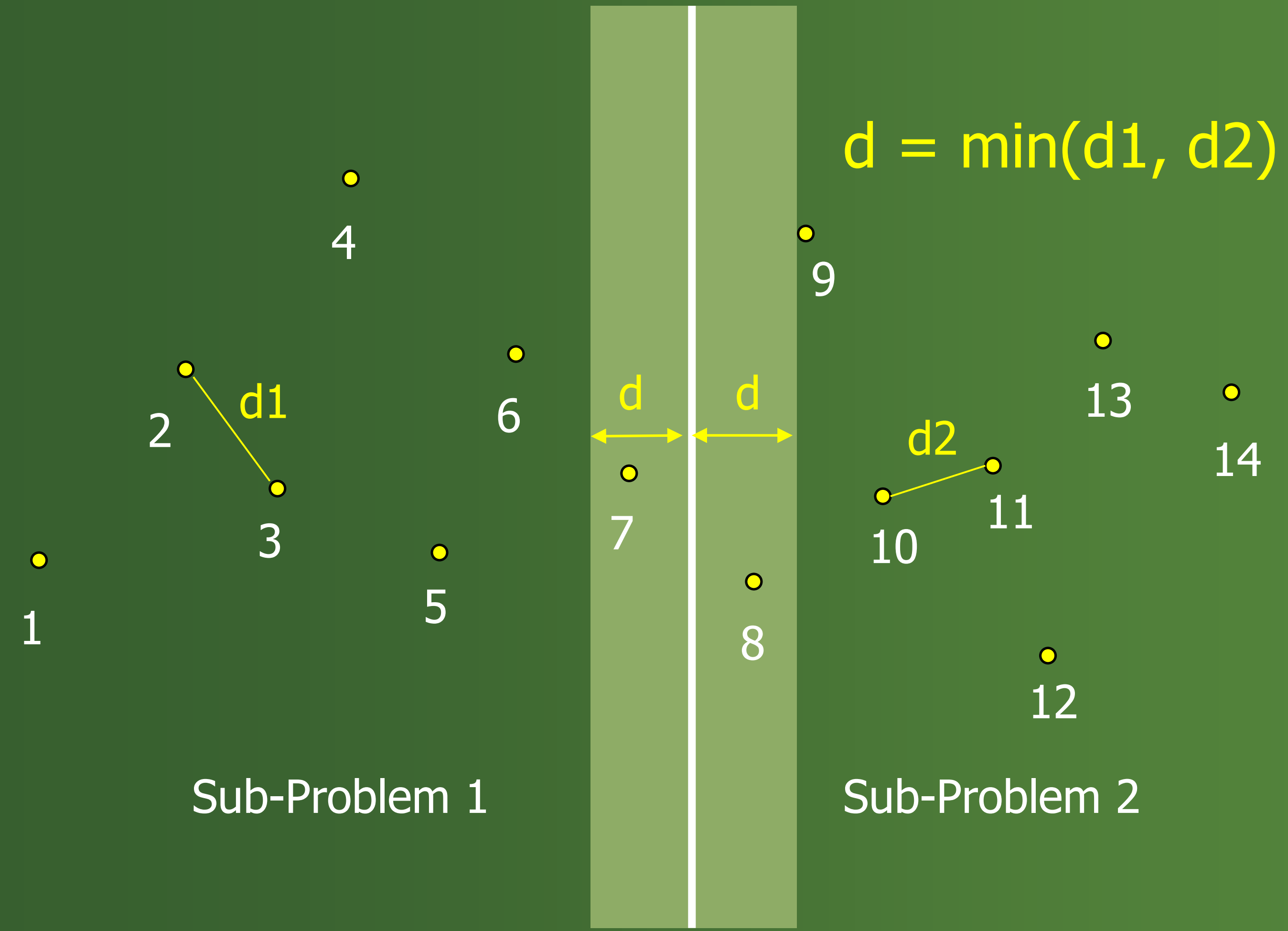
Closest-Pair Algorithm

Here is an example where we have to compare points from sub-problem 1 to the points in sub-problem 2.



Closest-Pair Algorithm

However, we only have to compare points inside the following “strip.”



Closest Pair of Points

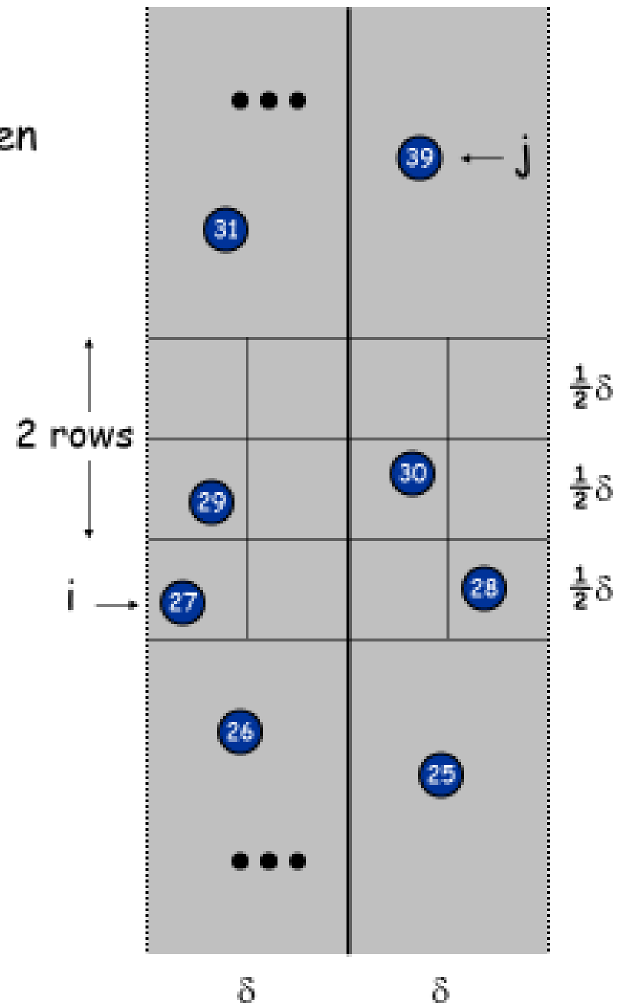
Def. Let s_i be the point in the 2δ-strip, with the i^{th} smallest y-coordinate.

Claim. If $|i - j| \geq 12$, then the distance between s_i and s_j is at least δ.

Pf.

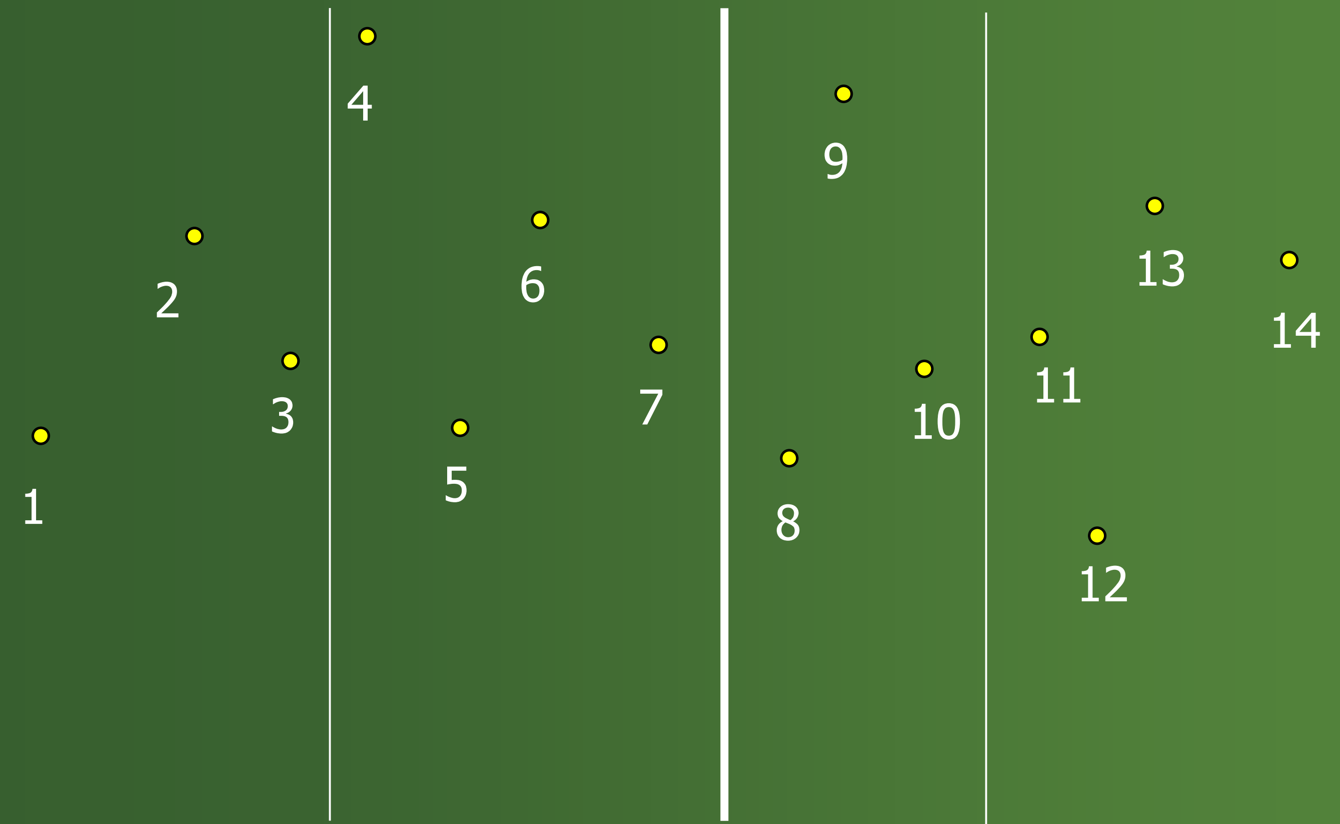
- No two points lie in same $\frac{1}{2}\delta$-by-$\frac{1}{2}\delta$ box.
- Two points at least 2 rows apart have distance $\geq 2(\frac{1}{2}\delta)$. ▪

Fact. Still true if we replace 12 with 7.



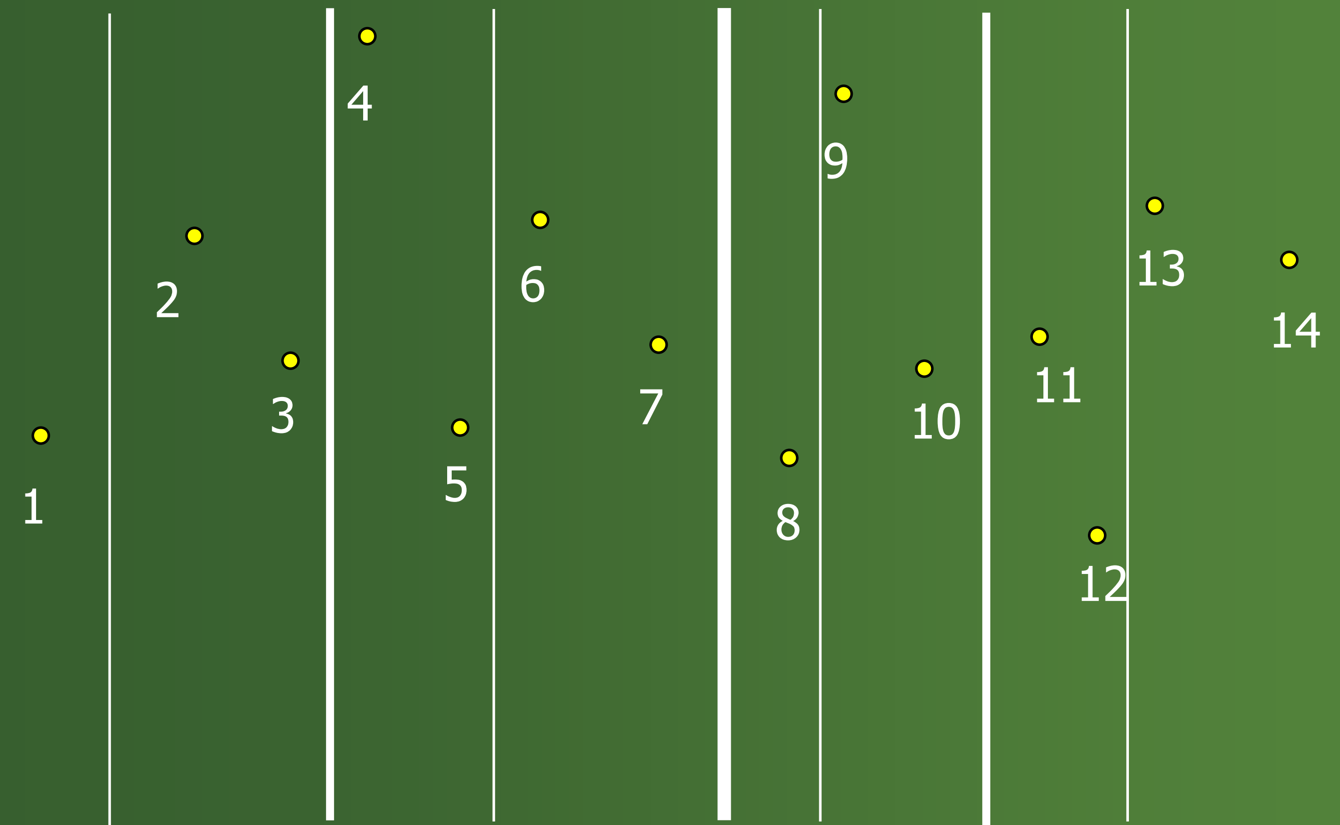
Closest-Pair Algorithm

Step 3: But, we can continue the advantage by splitting the sub-problems.



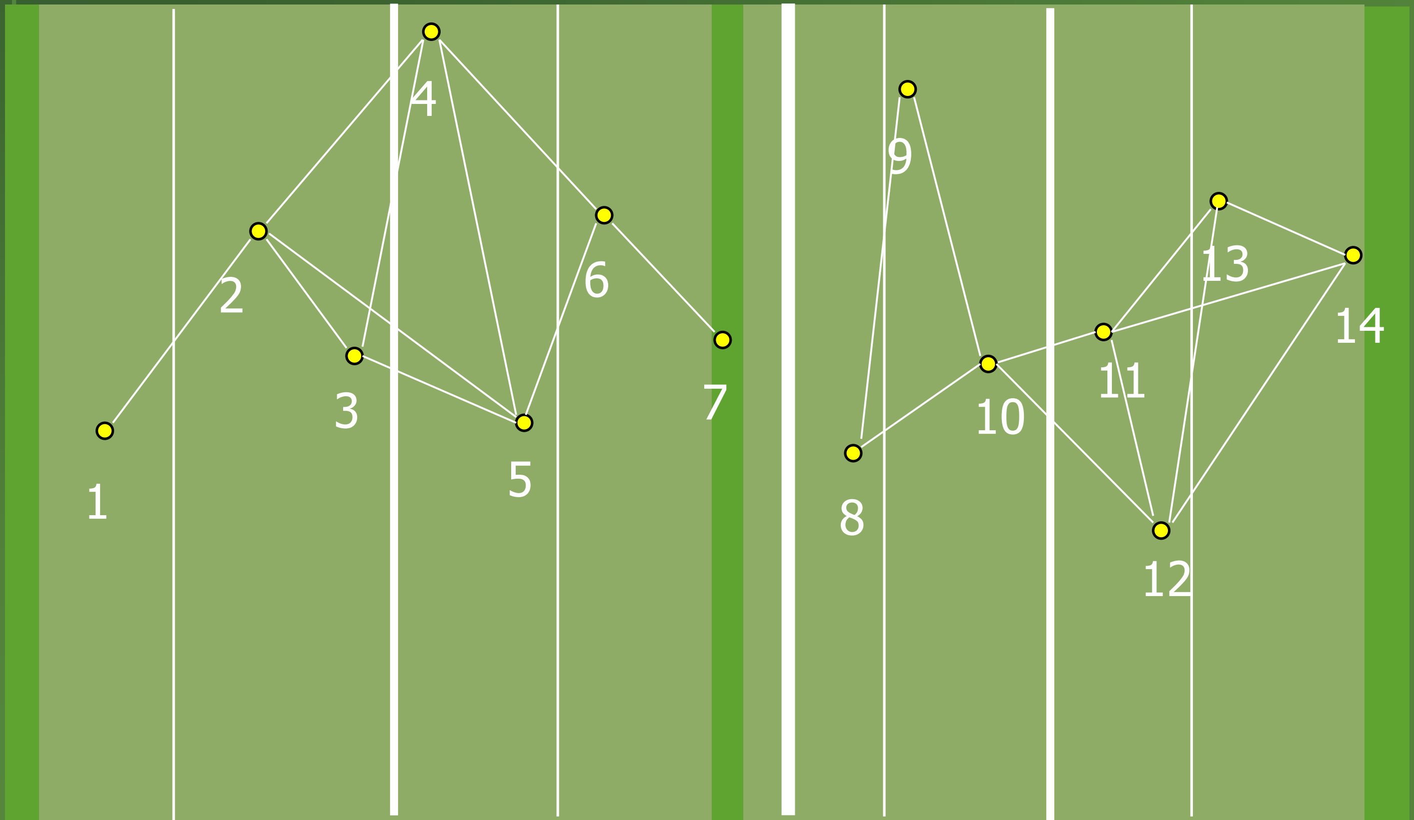
Closest-Pair Algorithm

Step 3: In fact we can continue to split until each sub-problem is trivial, i.e., takes one comparison.



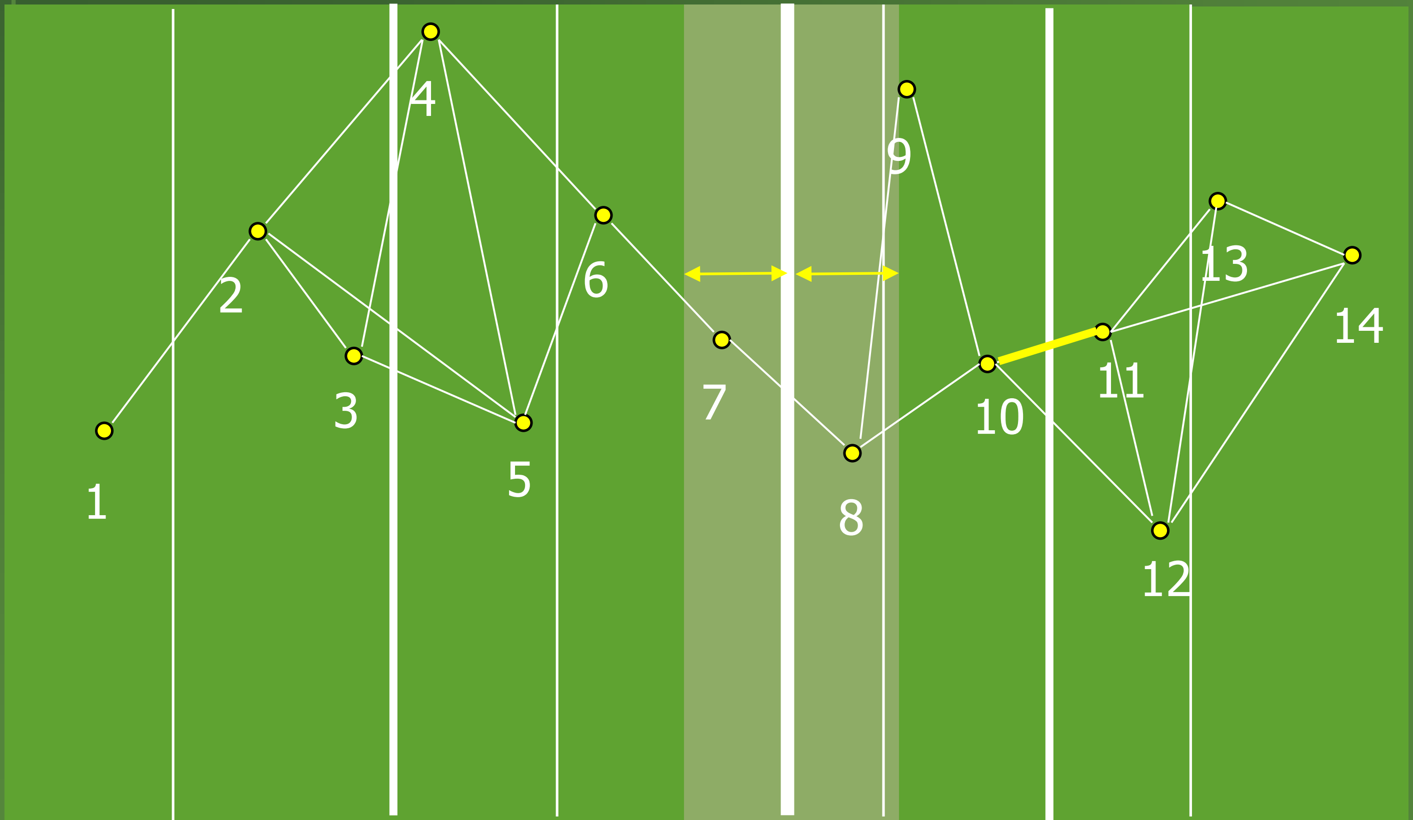
Closest-Pair Algorithm

Finally: The solution to each sub-problem is combined until the final solution is obtained



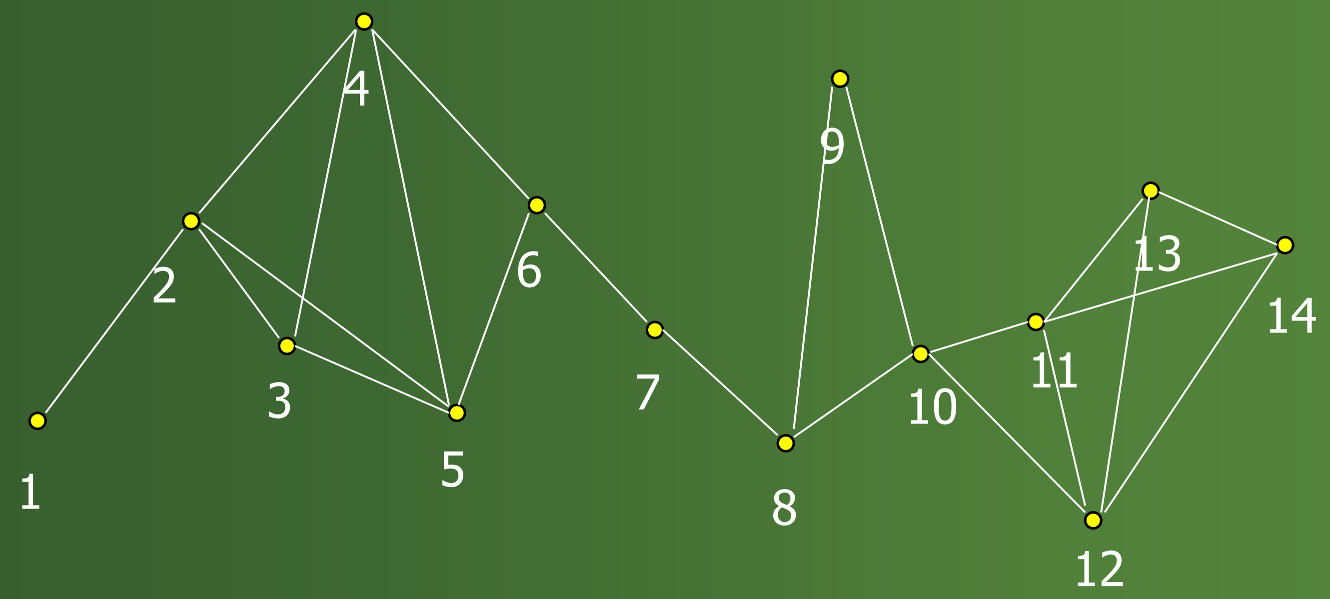
Closest-Pair Algorithm

Finally: On the last step the ‘strip’ will likely be very small. Thus, combining the two largest sub-problems won’t require much work.



Closest-Pair Algorithm

- In this example, it takes 22 comparisons to find the closets-pair.
- The brute force algorithm would have taken 91 comparisons.
- But, the real advantage occurs when there are millions of points.



Closest-Pair Problem: Divide and Conquer

- Here is another animation:
- http://www.cs.mcgill.ca/~cs251/ClosestPair/ClosestPairApplet/ClosestPairApplet.html