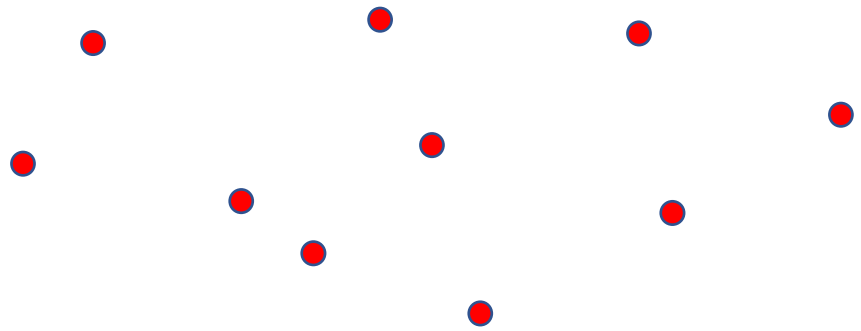
Closest Pair

P12. Closest pair of n points

I/P: Given $P=\{p_1, p_2,\ldots, p_n\}$ a set of n points. Each $p_i=(x_i,y_i)$.

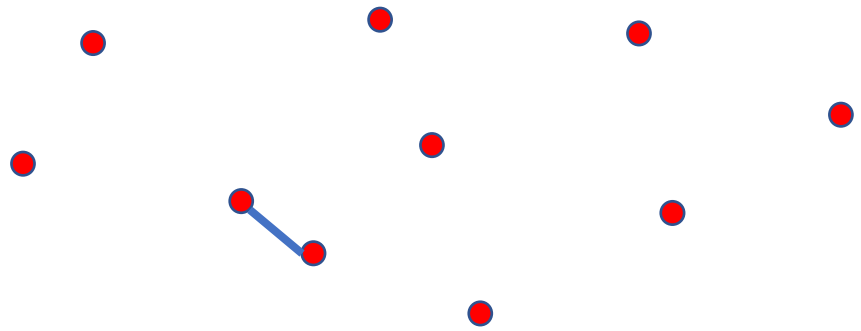
O/P: Find the closest pair of points.



P12. Closest pair of n points

I/P: Given $P=\{p_1, p_2,\ldots, p_n\}$ a set of n points. Each $p_i=(x_i,y_i)$.

O/P: Find the closest pair of points.



Closest pair of n points

I/P: Given $P=\{p_1, p_2,\ldots, p_n\}$ a set of n points. Each $p_i=(x_i,y_i)$.

O/P: Find the closest pair of points.

Naïve algorithm:

1. For each point p in P, check all points in P \ {p}.
2. Compute the minimum distances with respect to p, i.e., min_p
3. Repeat step 1 and 2 for all points in P,
4. Finally return min of all min_p's.

Let’s do the problem in 1-Dimension

y-coordinate of all points are same.

Let’s do the problem in 1-Dimension

y-coordinate of all points are same.

1.Sort each point with respect to x-coordinate.

Let’s do the problem in 1-Dimension

y-coordinate of all points are same.

1.Sort each point with respect to x-coordinate.

2. For each point p_i , check two neighbours p_{i-1} and p_{i+1}.

3. Do it for all points and return the minimum.

Let’s do the problem in 1-Dimension

y-coordinate of all points are same.

1.Sort each point with respect to x-coordinate. (doing this to know two adjacent points)

2. For each point p_i , check two neighbours p_{i-1} and p_{i+1}.

3. Do it for all points and return the minimum.

Let’s do the problem in 1-Dimension

y-coordinate of all points are same.

1.Sort each point with respect to x-coordinate. O(nlogn)

2. For each point p_i , check two neighbours p_{i-1} and p_{i+1}.
3. Do it for all points and return the minimum.

O(n)

Can we do via divide and conquer?

Here input is not sorted….. Means the point are not sorted with respect to x-coordinate…. ??

How to apply D&C??

Can we do via divide and conquer?

Here input is not sorted….. Means the point are not sorted with respect to x-coordinate…. ??

How to apply D&C??

Intuition is to divide points set in equal halves ??– But how??

Can we do via divide and conquer?

Here input is not sorted….. Means the point are not sorted with respect to x-coordinate…. ??

How to apply D&C??

Intuition is to divide points set in equal halves ??– But how??

We need median to partition in two halves….??

Lets apply the Select(P,n/2) algorithm to partition point set in two halves.

P split in two halves P_L and P_R

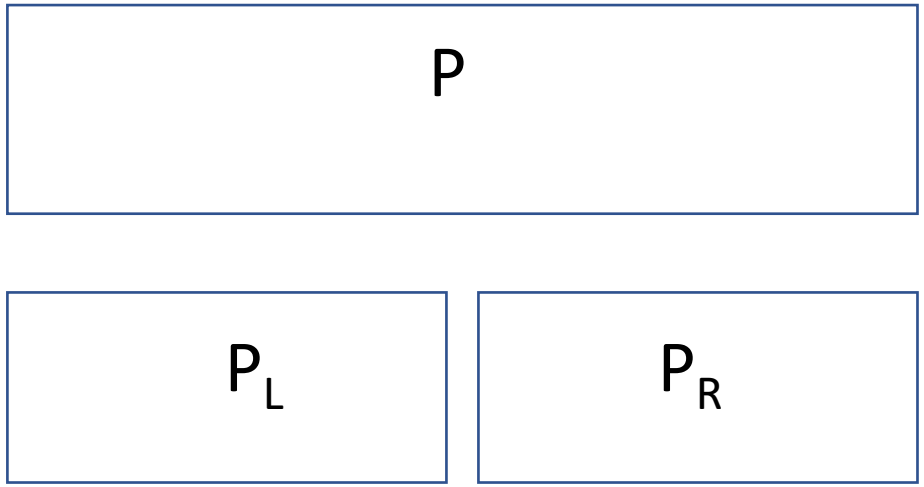
Recurse on both halves.

Lets apply the unknown algorithm to partition point set in two halves.

P split in two halves P_L and P_R.

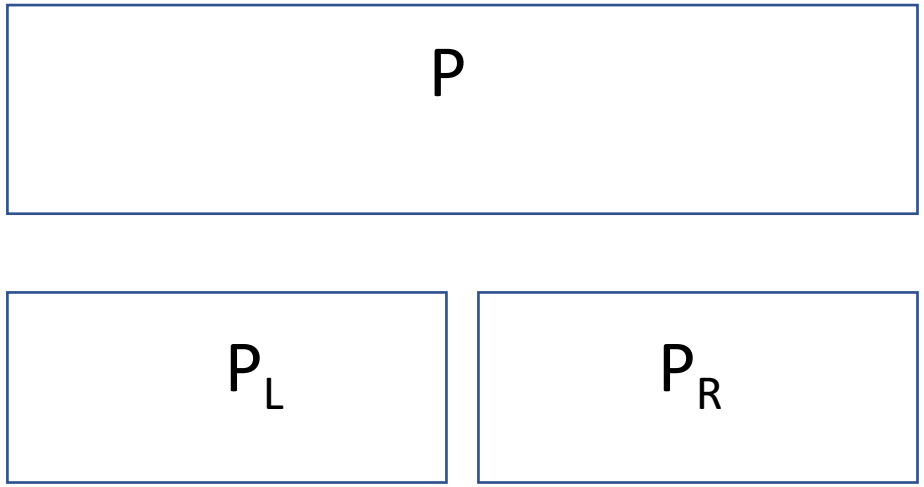
Recurse on both halves.

P_L contains the median point (W.L.O.G)…



Suppose the closest pair in P_L is d_L

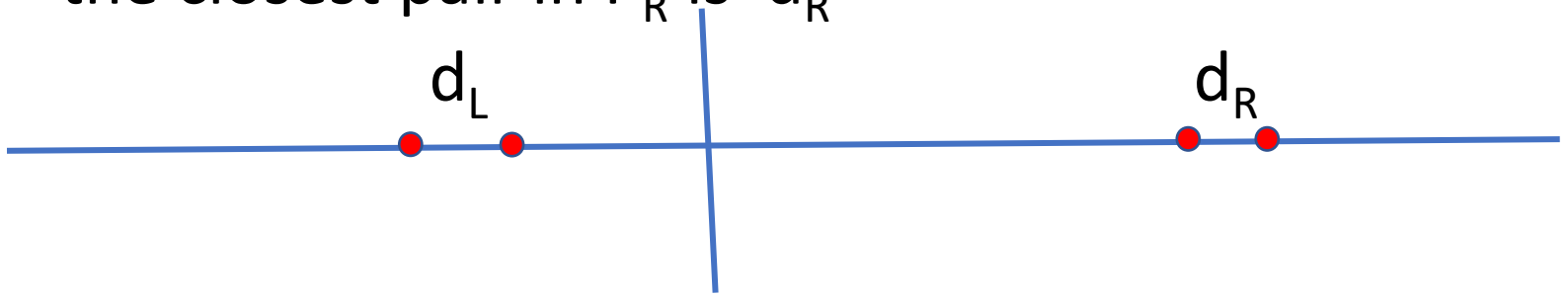
the closest pair in P_R is d_R

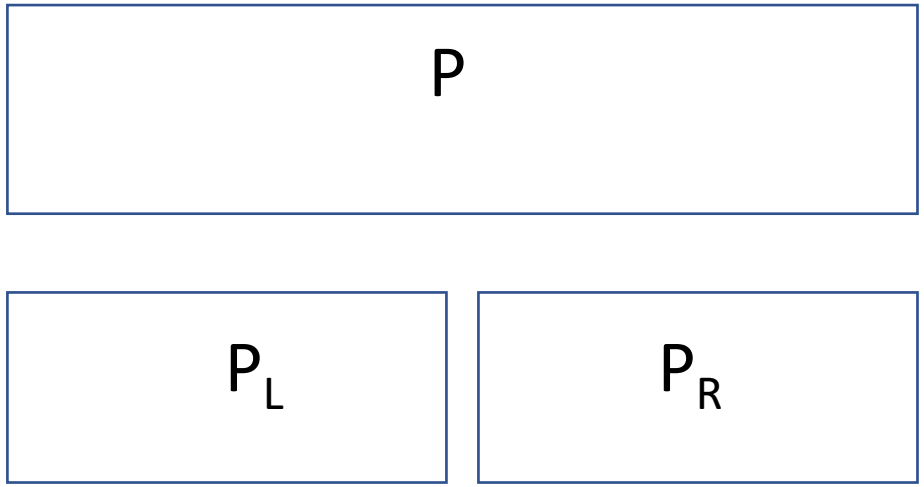


Suppose the closest pair in P_L is d_L

the closest pair in P_R is d_R

$s=\min(d_L, d_R)$

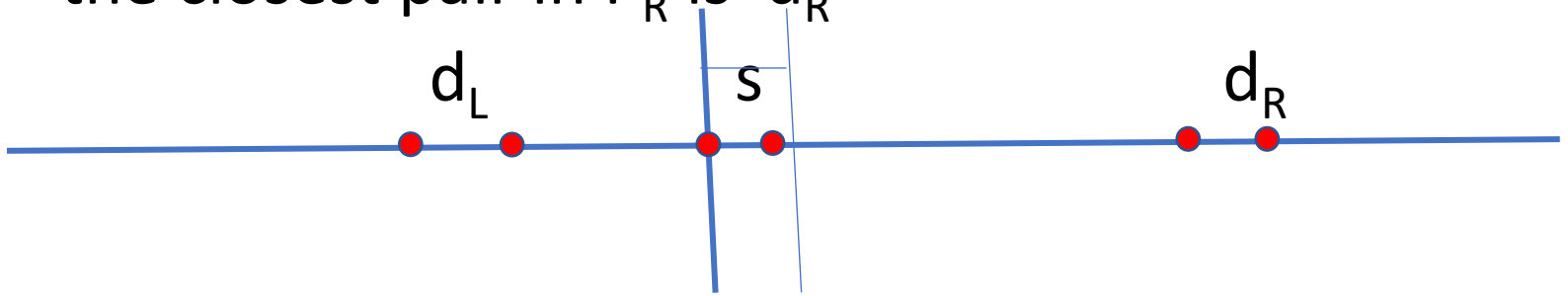


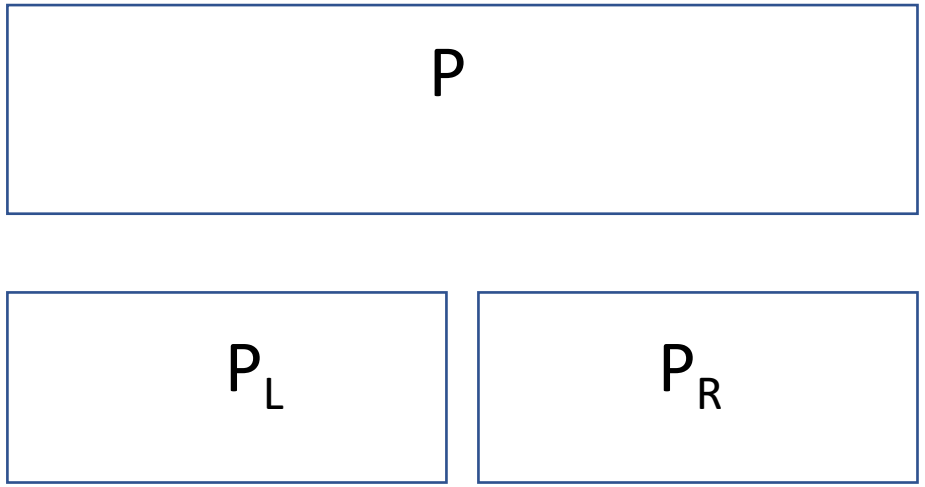


Suppose the closest pair in P_L is d_L

the closest pair in P_R is d_R

$s=\min(d_L, d_R)$



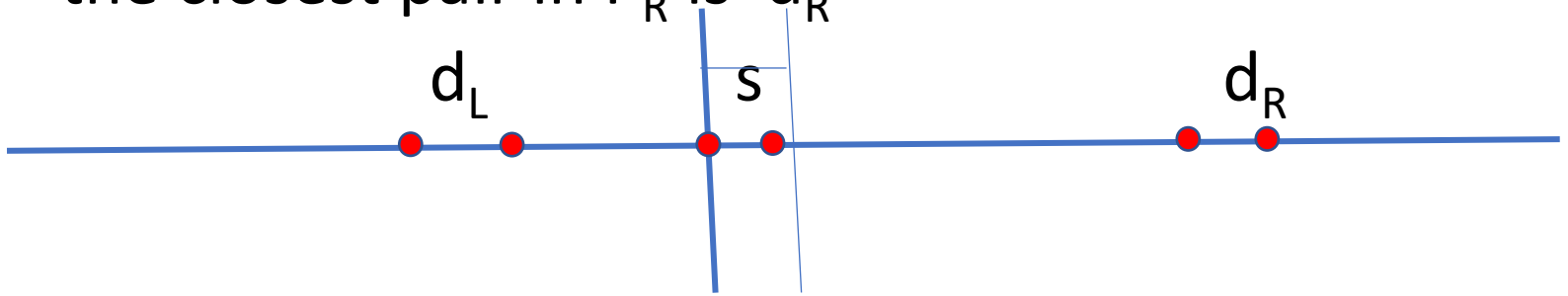


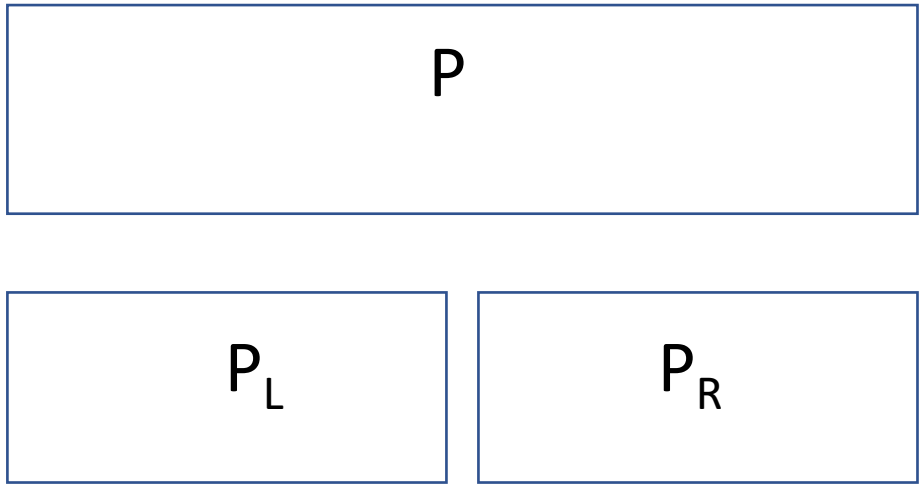
complexity??

Suppose the closest pair in P_L is d_L

the closest pair in P_R is d_R

$s=\min(d_L, d_R)$



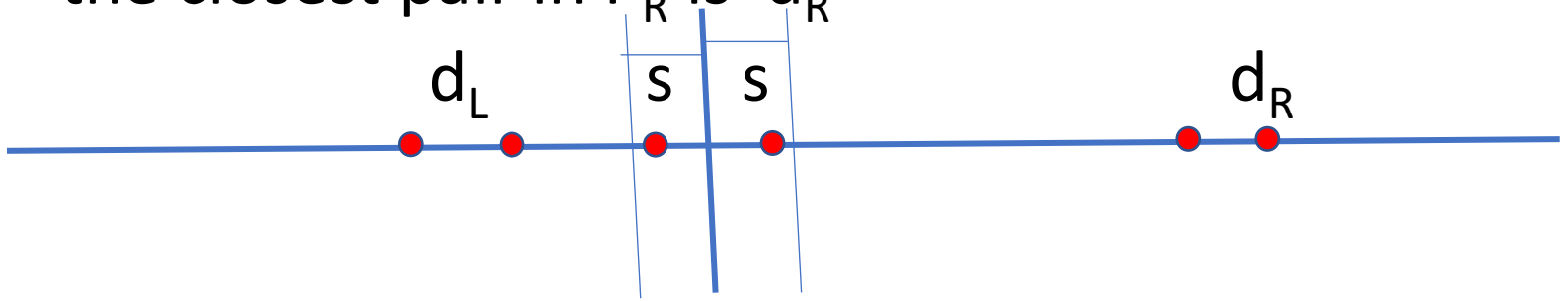


complexity??

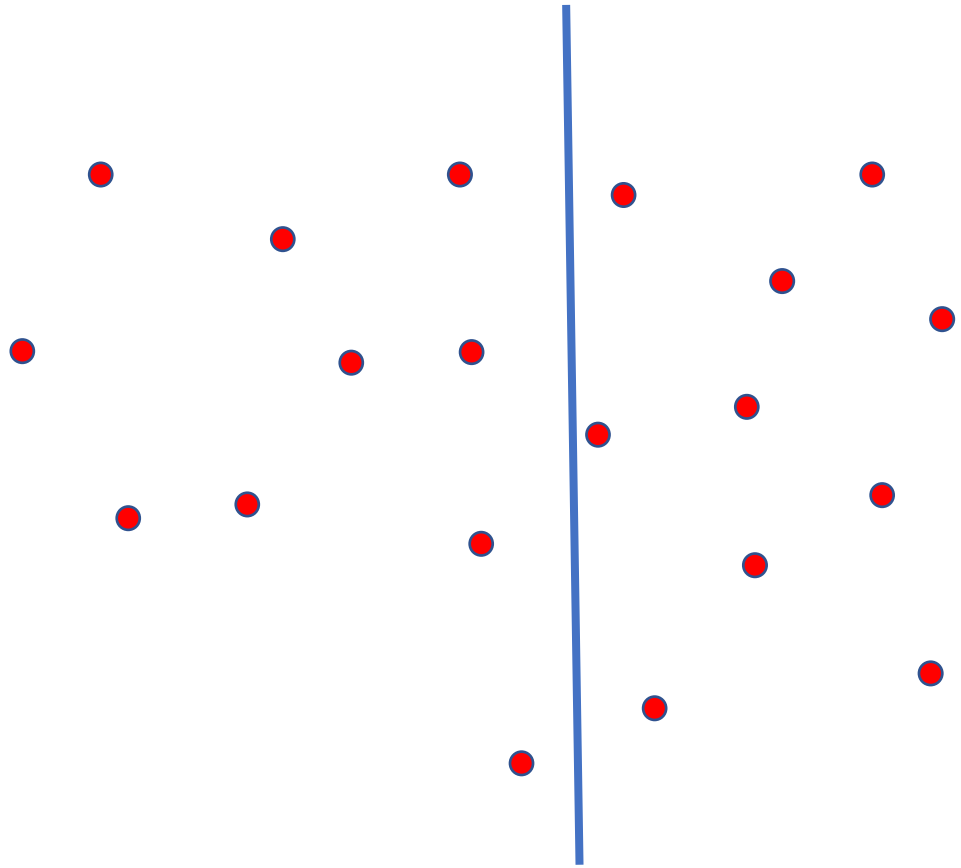
Suppose the closest pair in P_L is d_L

the closest pair in P_R is d_R

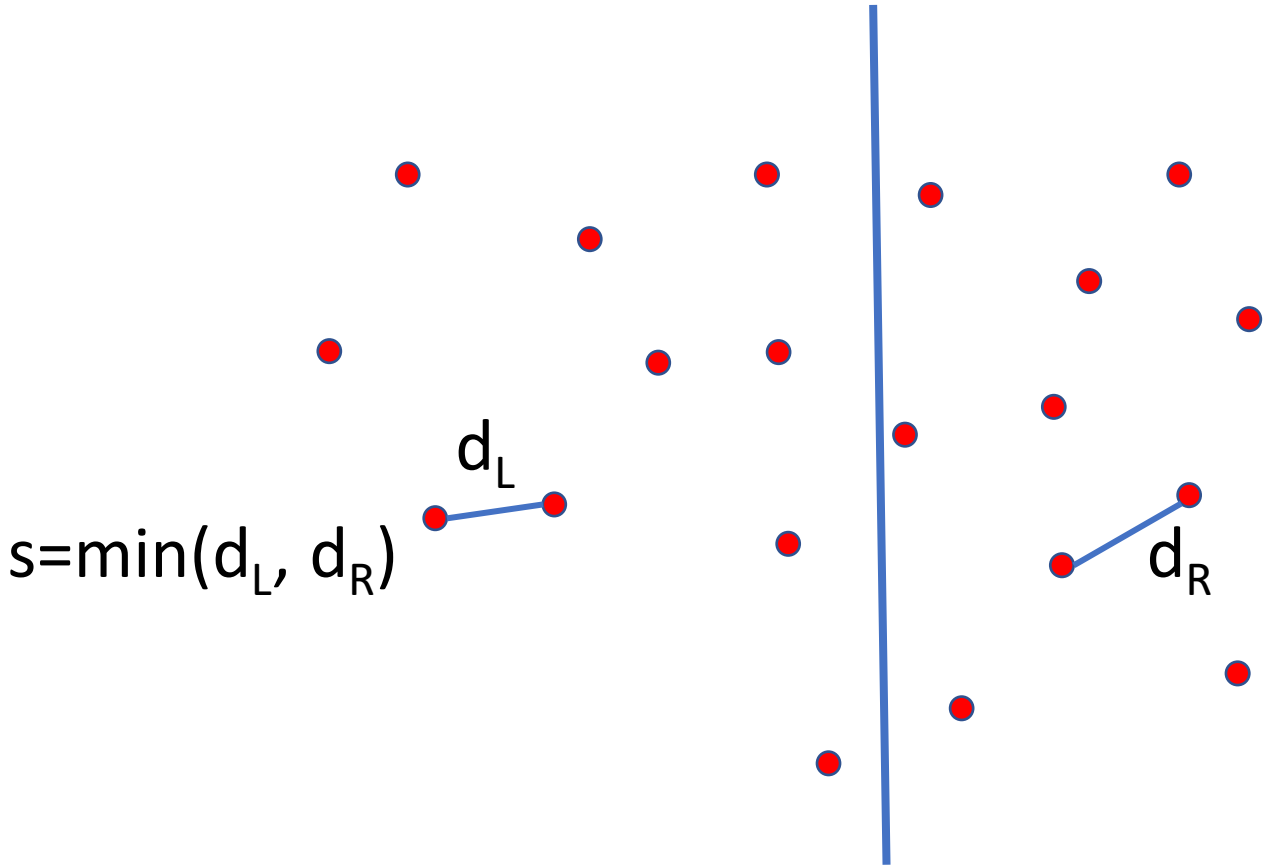
$s=\min(d_L, d_R)$



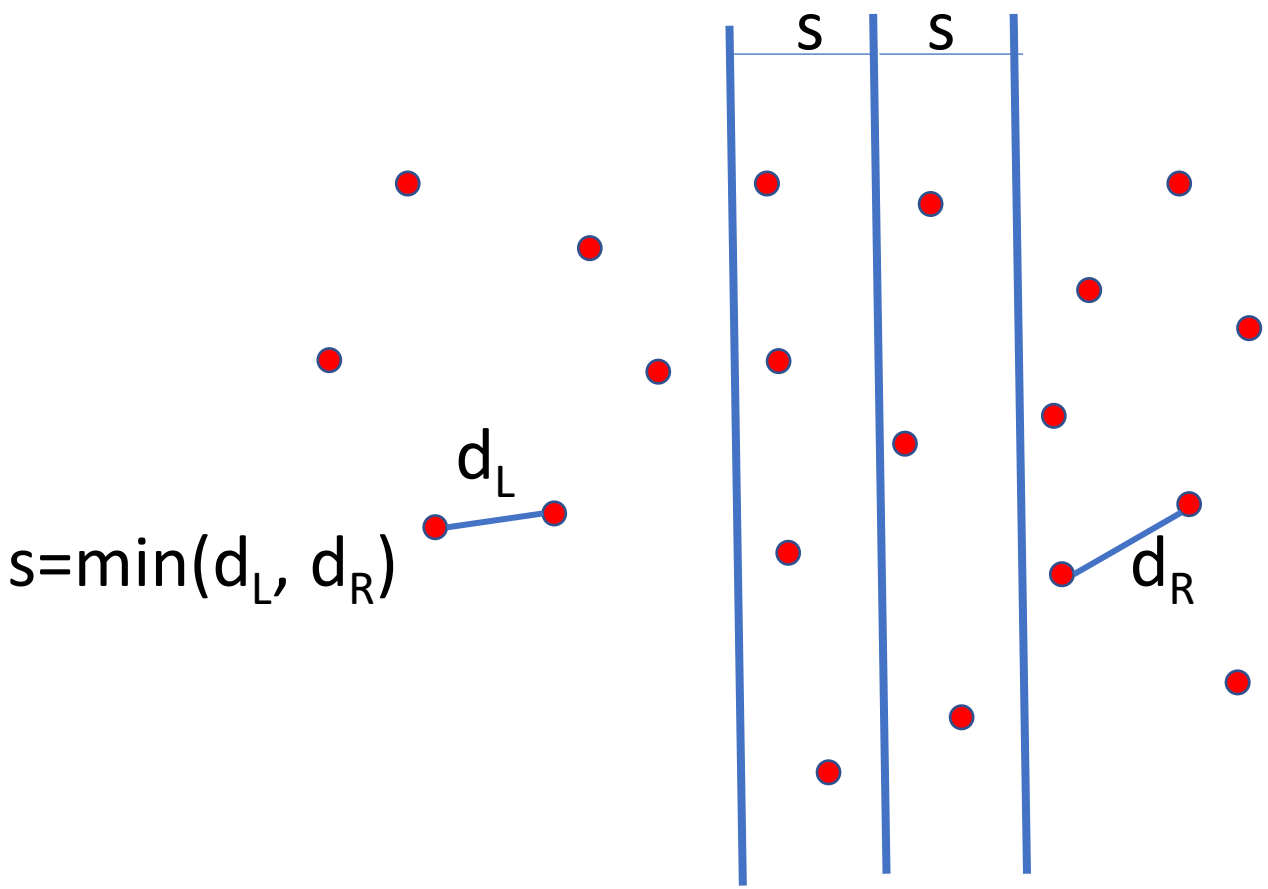
In 2D



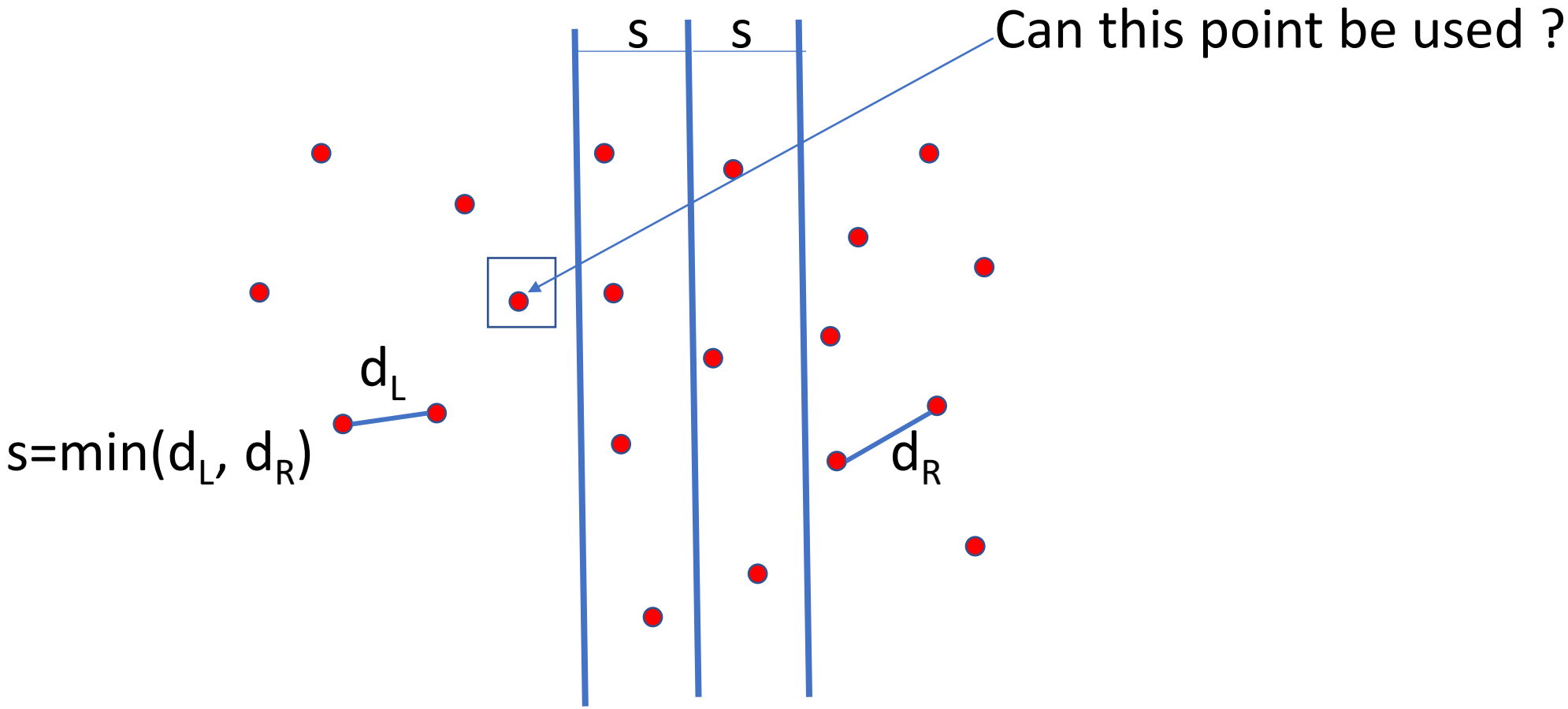
In 2D



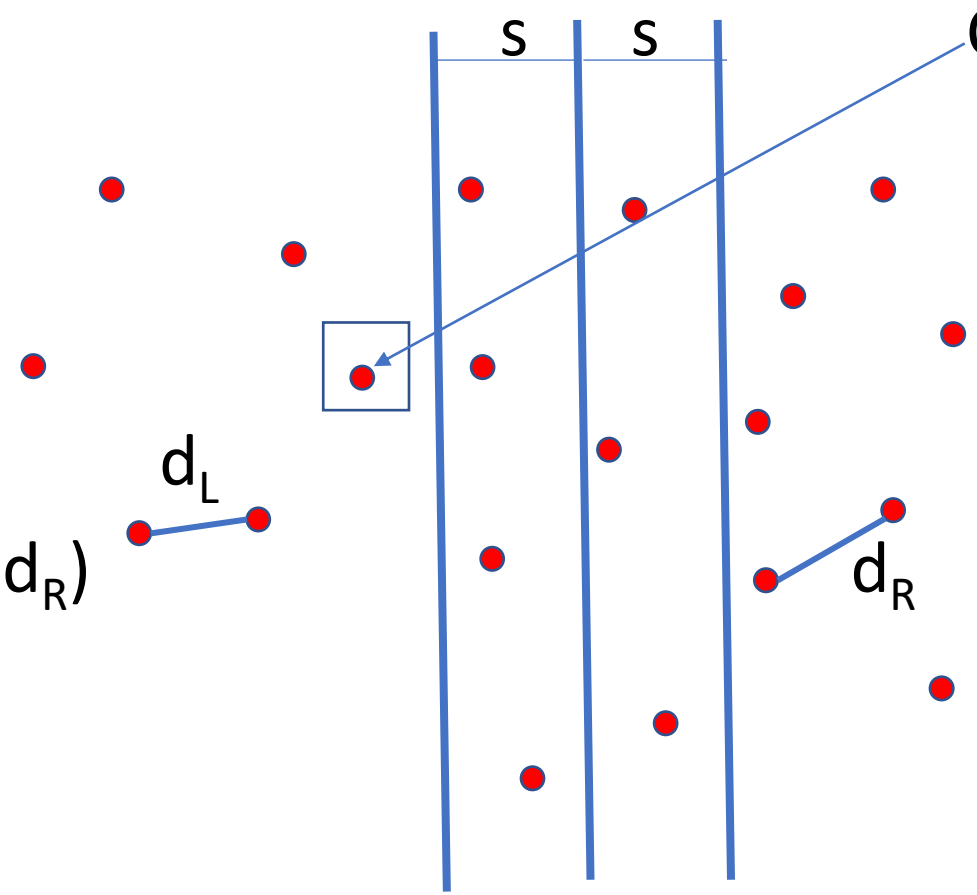
In 2D



In 2D



In 2D

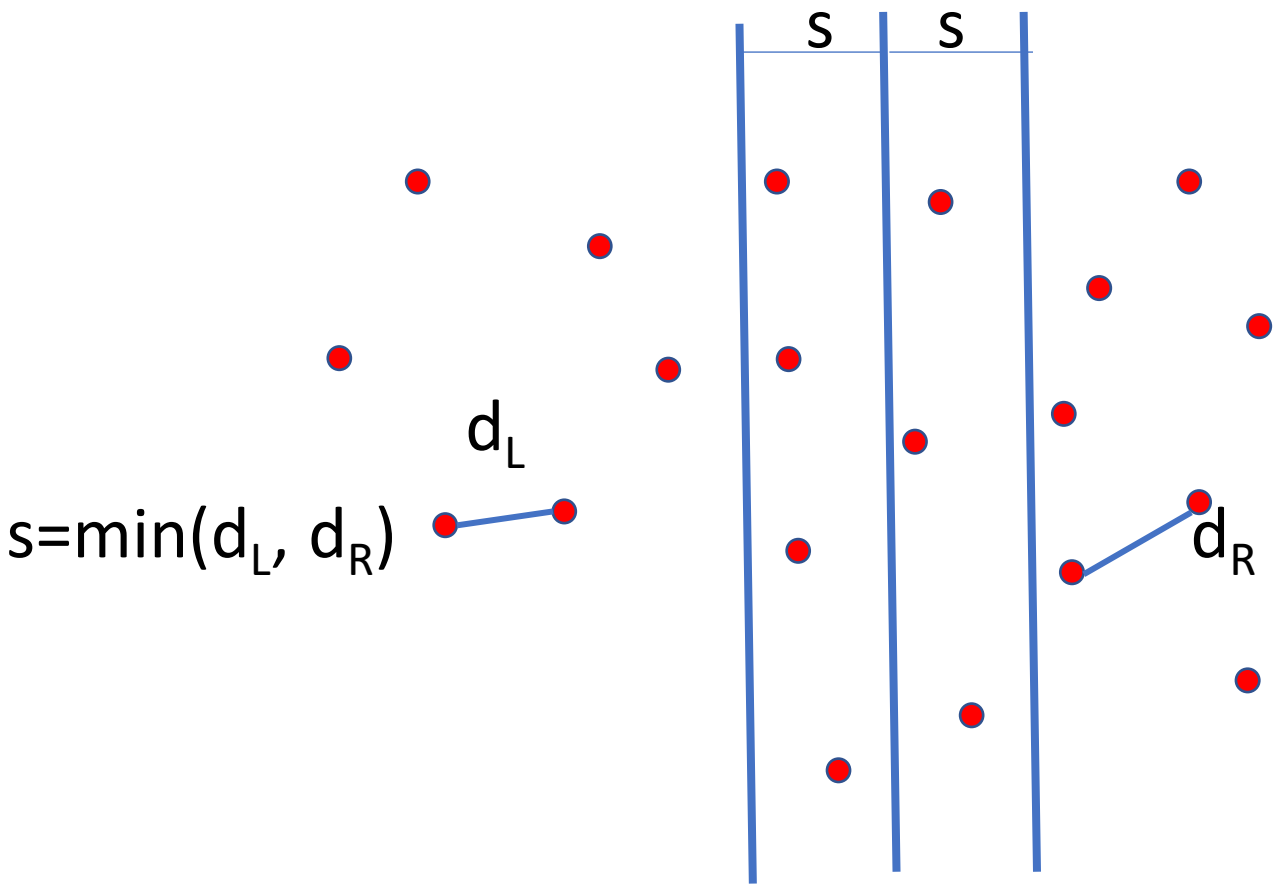


Can this point be used ?

No, so check points in strip.

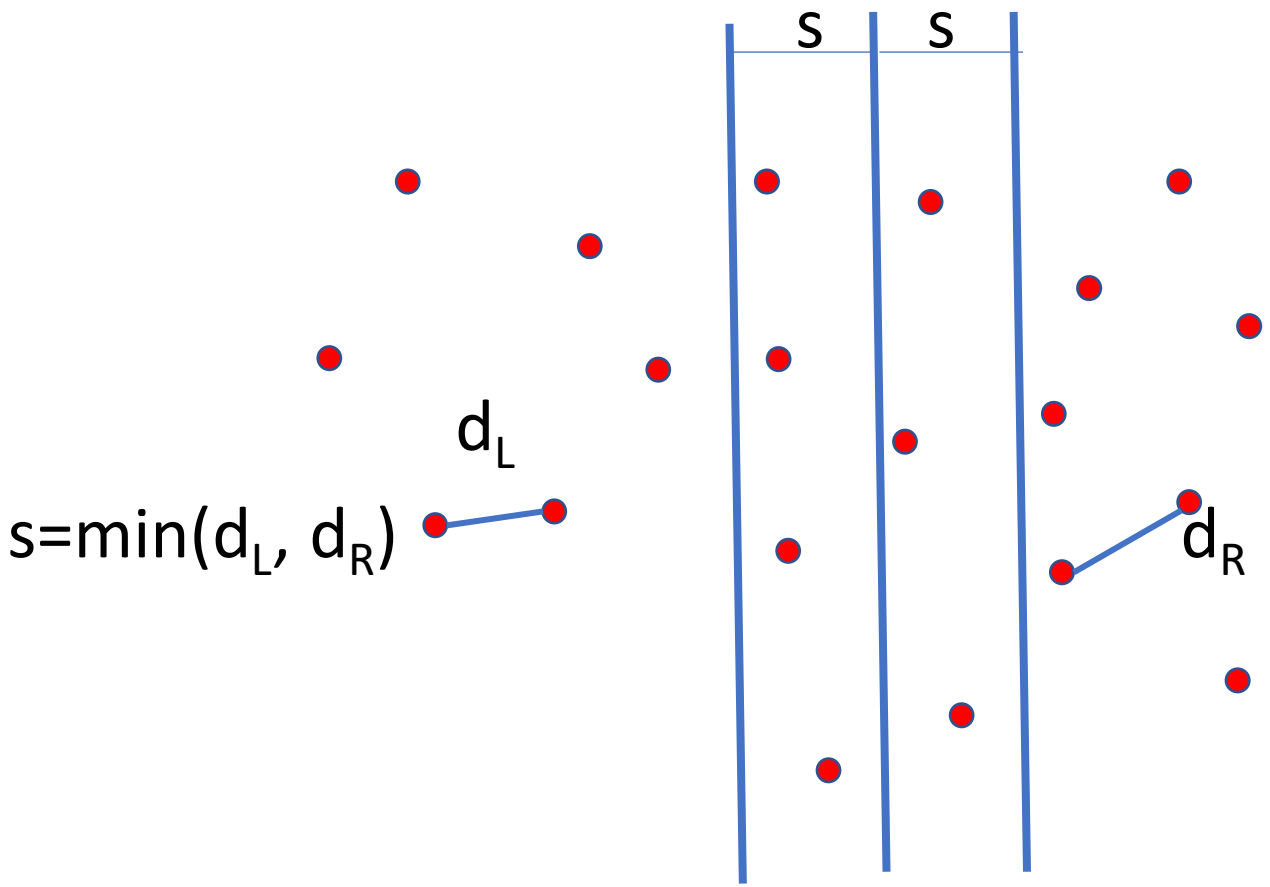
$s=\min(d_L, d_R)$

In 2D



So, we need to find the closest pair in strip only.

In 2D

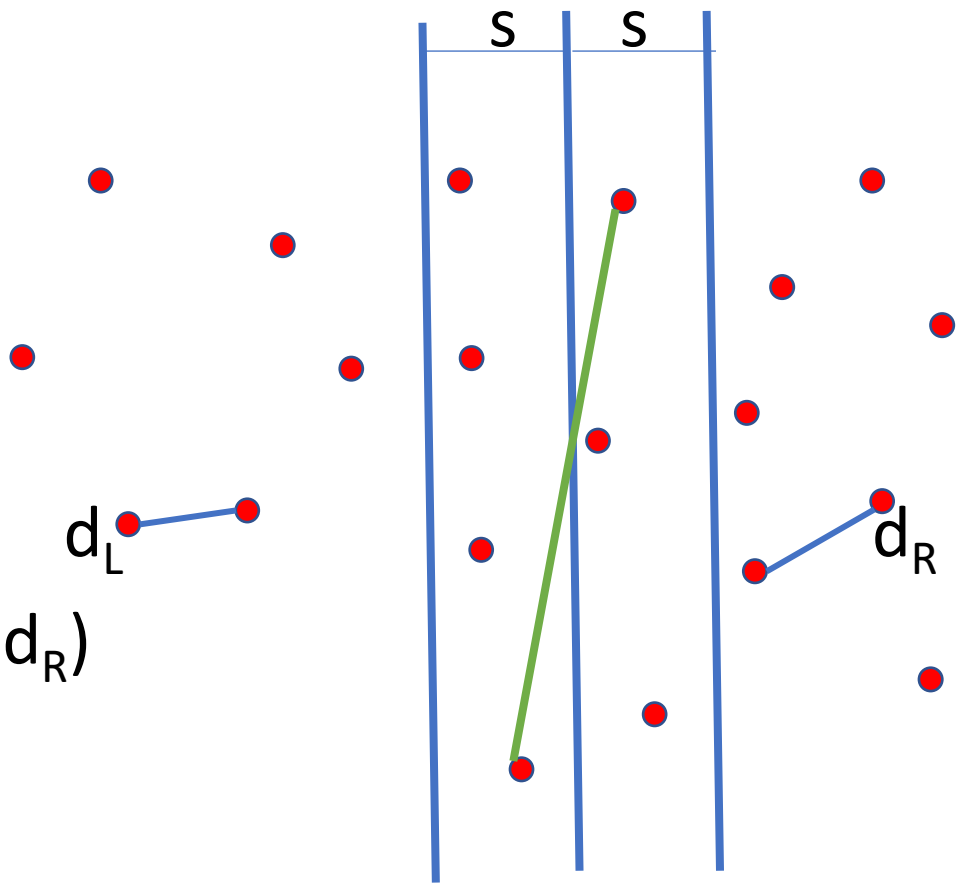


So, we need to find the closest pair in strip only.

naïve way: $O(n^2)$

$T(n)=T(n/2)+ O(n^2)$

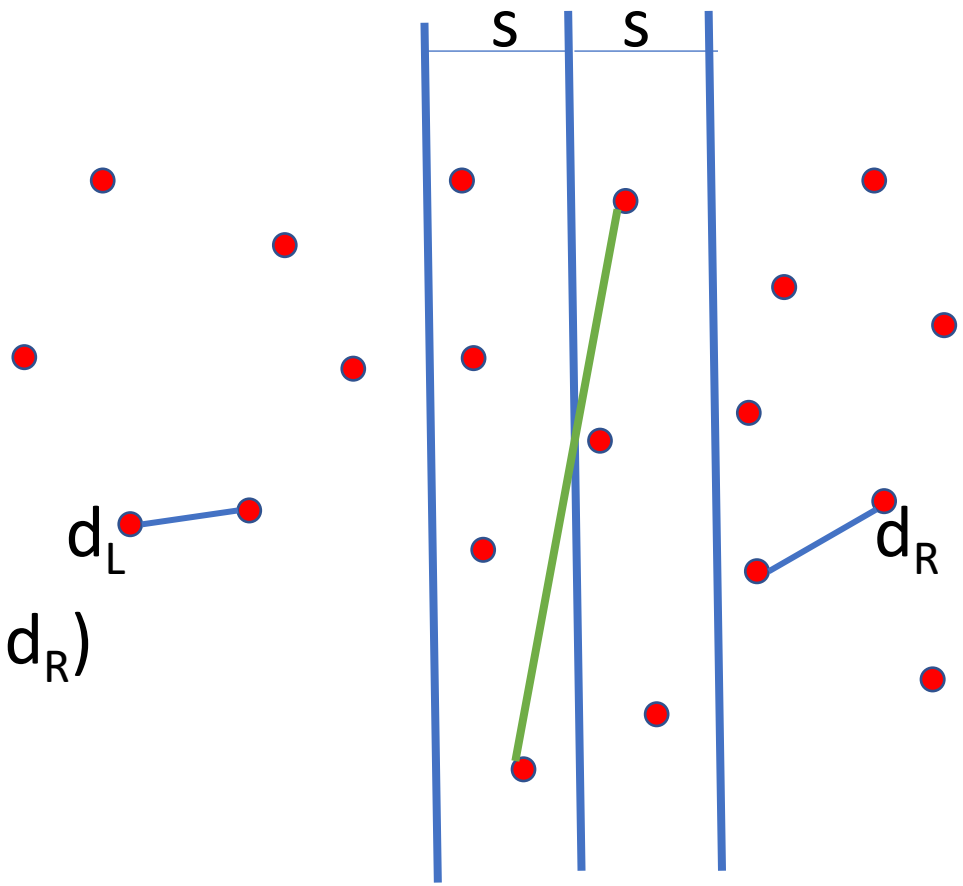
In 2D



$s=\min(d_L, d_R)$

Do we need to compare this pair?

In 2D

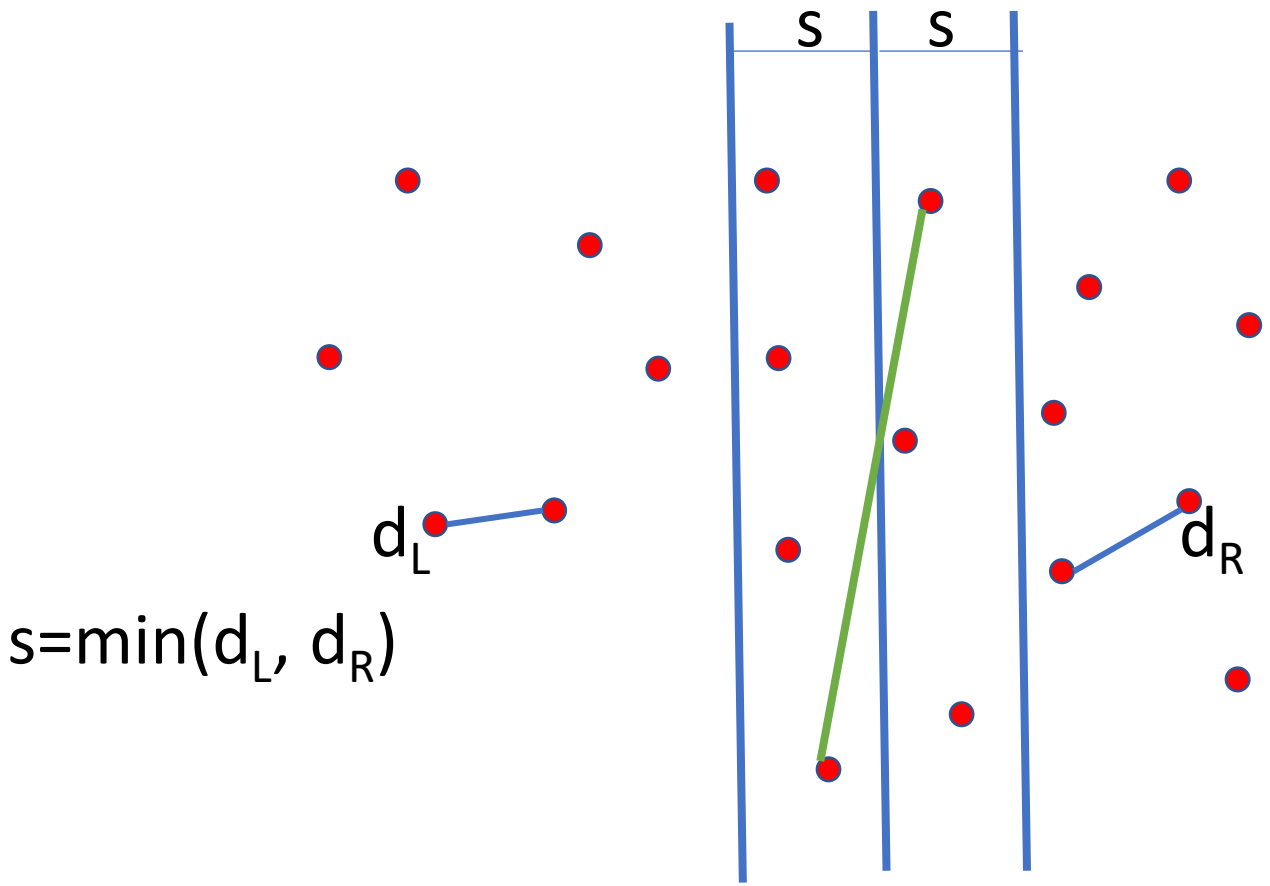


$s=\min(d_L, d_R)$

Do we need to compare this pair?

NO!!

In 2D

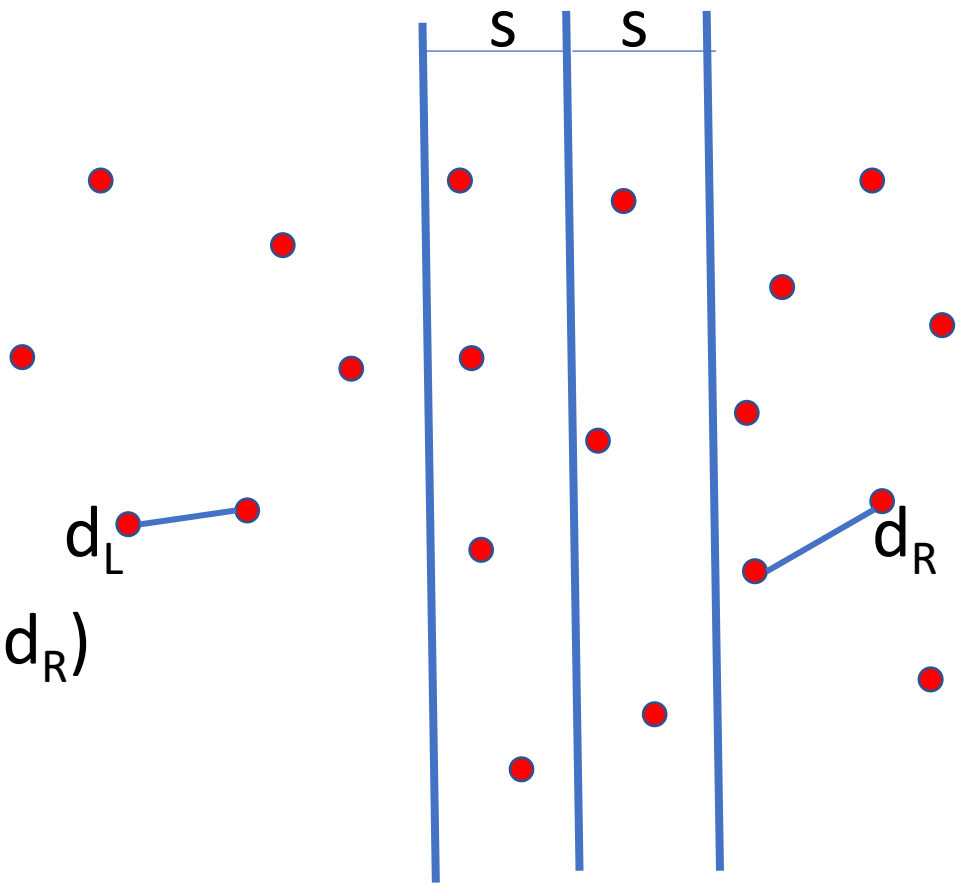


Do we need to compare this pair?

NO!!

Let's dissects the strip.

In 2D

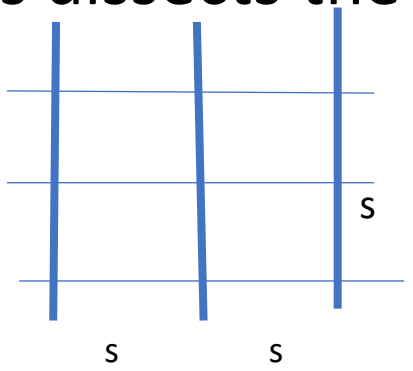


$s=\min(d_L, d_R)$

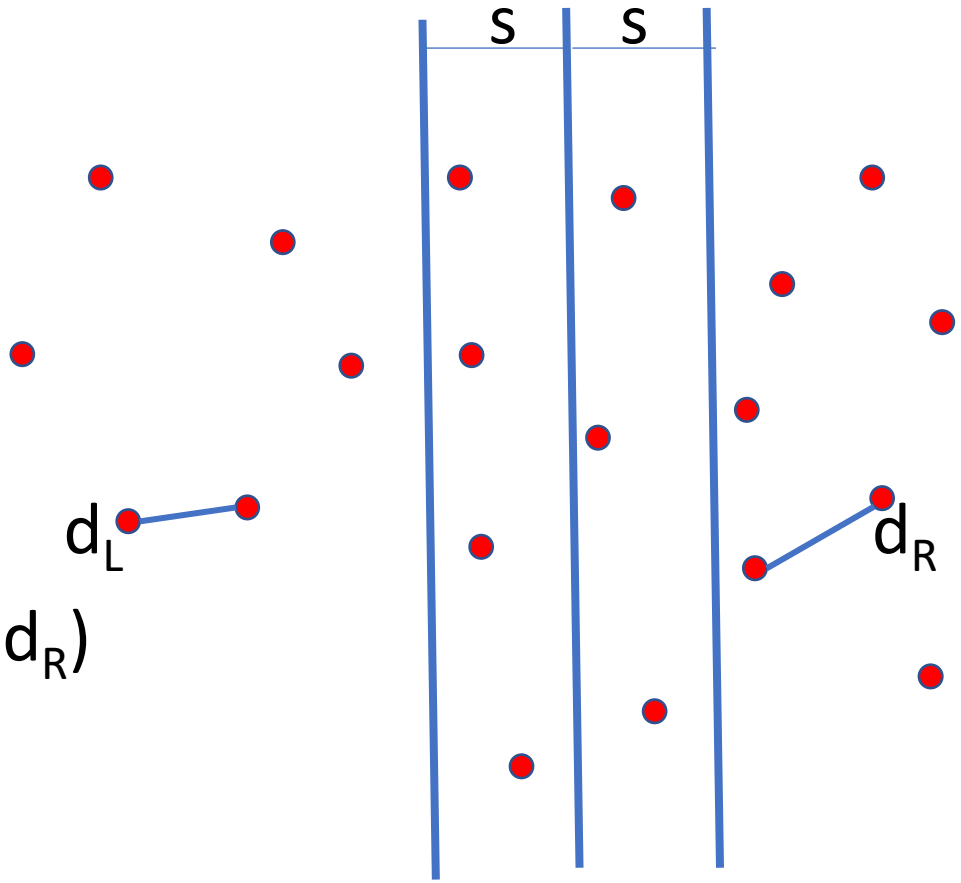
Do we need to compare this pair?

NO!!

Let's dissects the strip.



In 2D

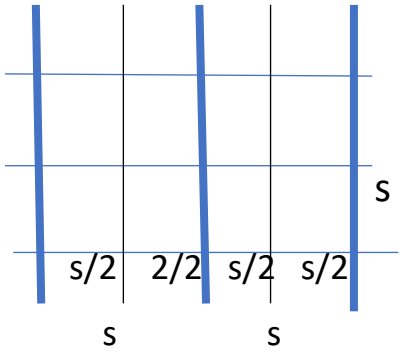


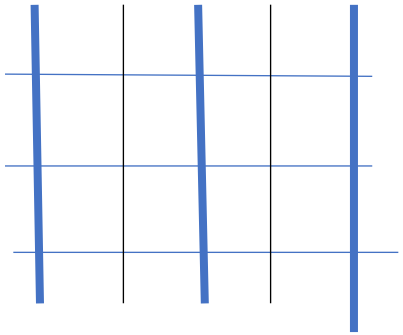
$s=\min(d_L, d_R)$

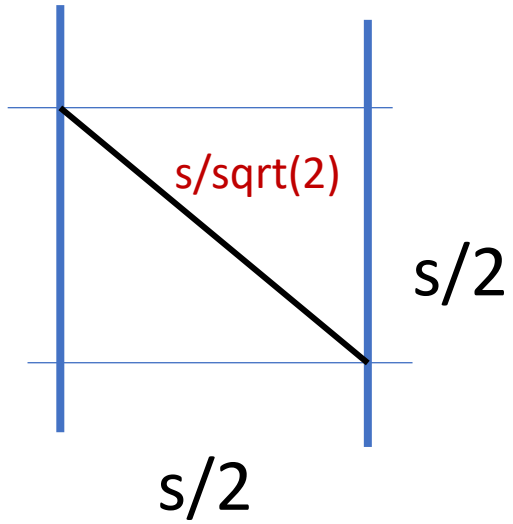
Do we need to compare this pair?

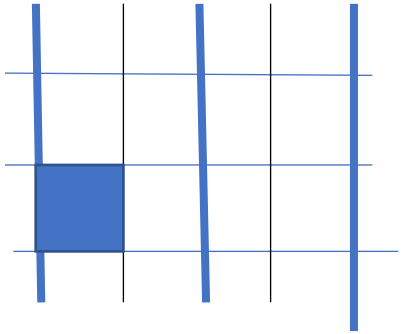
NO!!

Let's dissects the strip.

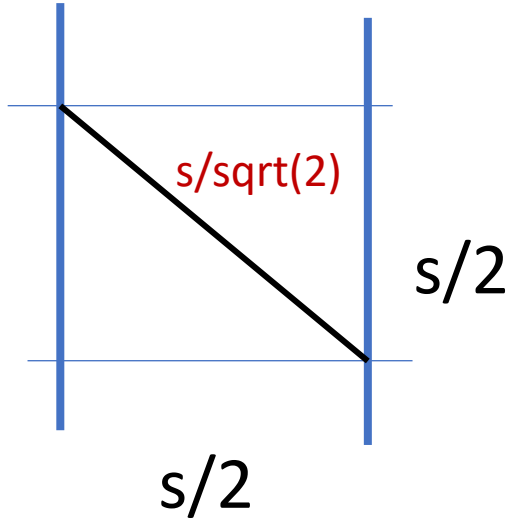


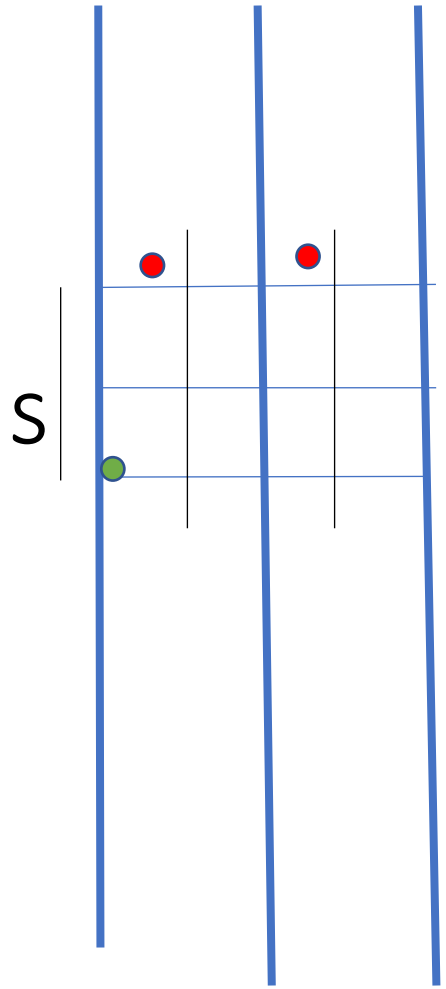



s/sqrt(2)
s/2
s/2



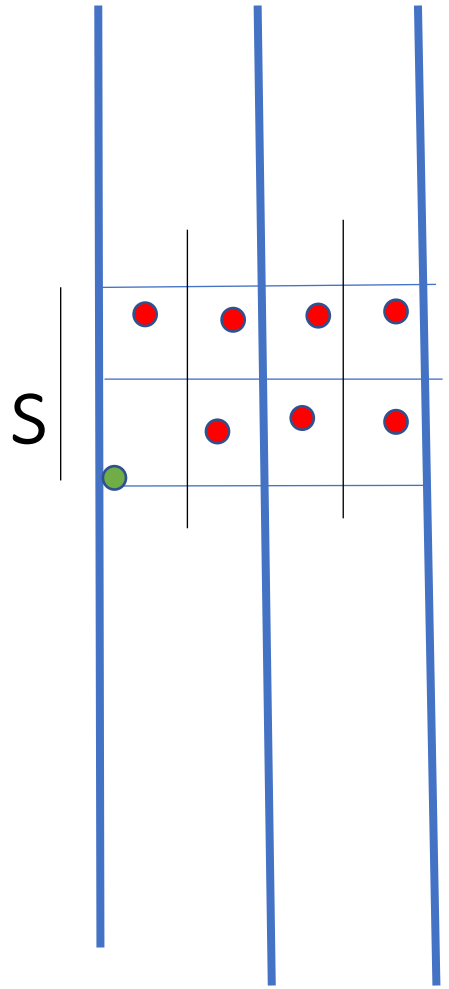
so, in a box, can have one point.





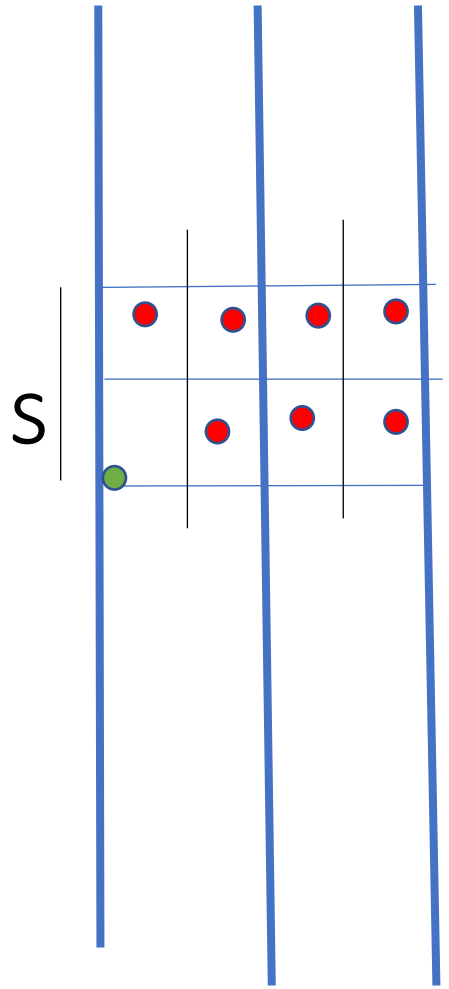
for green point, do we need to check these red points ?

NO? Why?



for green point, we need to check these red points ?

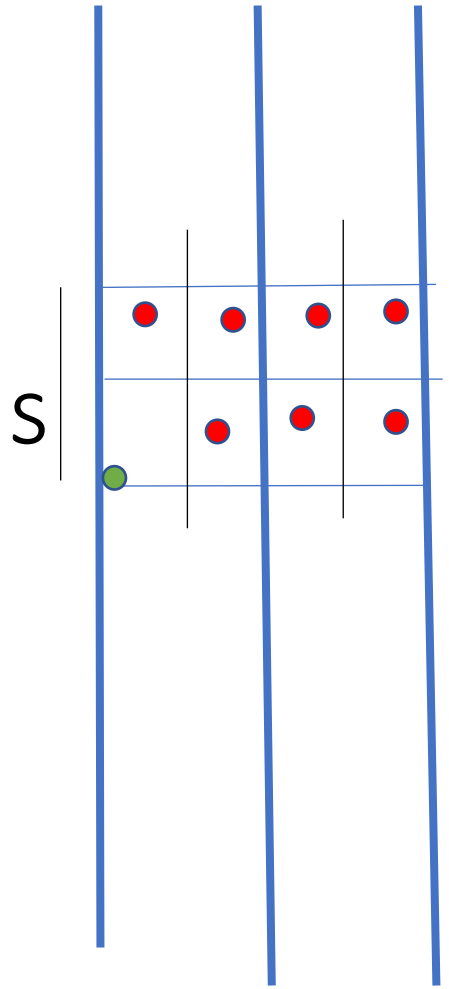
Yes? Why?



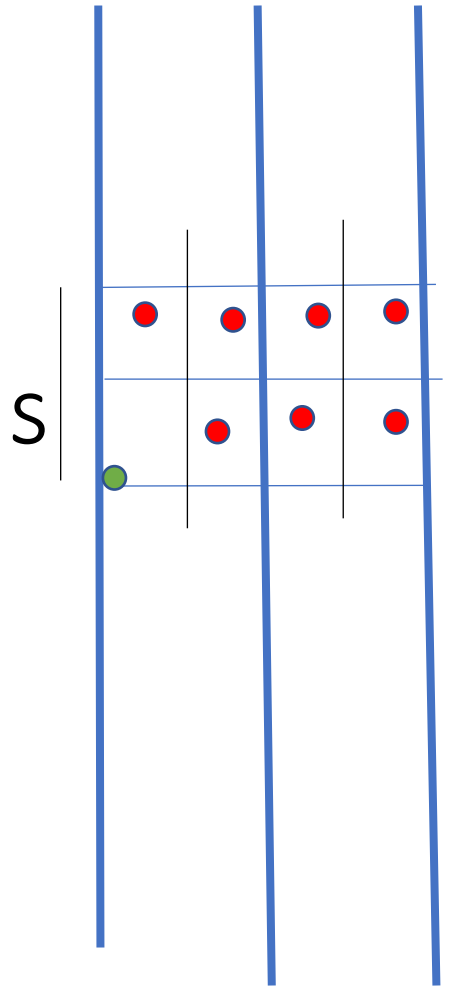
for green point, we need to check these red points ?

Yes? Why?

7 points



how to find seven points for each point?



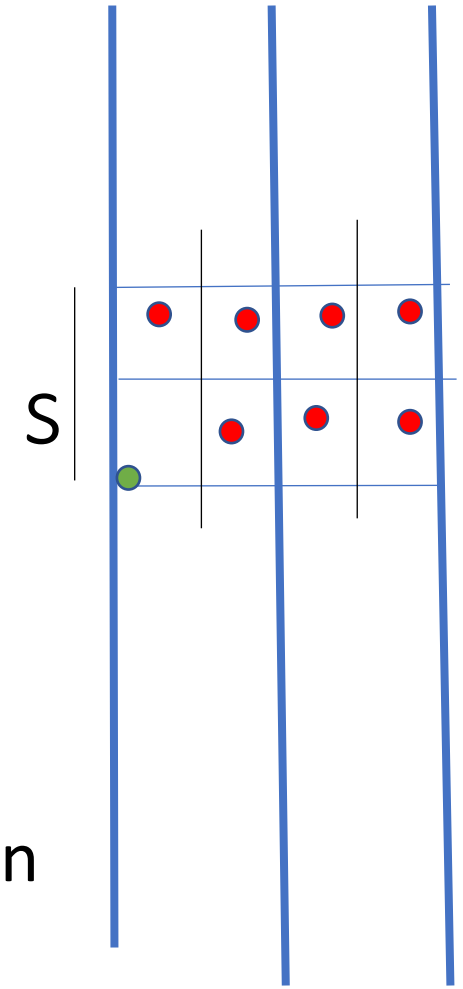
for green point, we need to check these red points ?

Yes? Why?

7 points

so, for each point, we need to check at most next 7 points.

To get next 7, need to sort



for green point, we need to check these red points ?

Yes? Why?

7 points

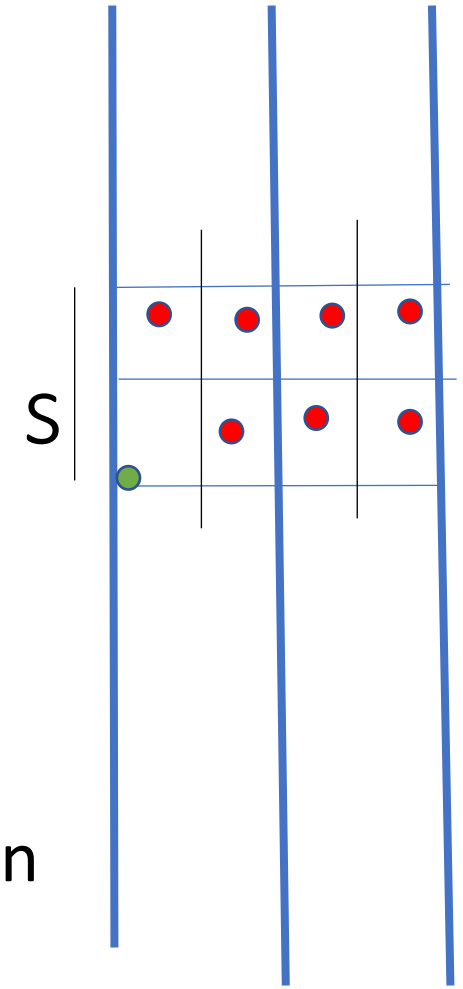
so, for each point, we need to check at most next 7 points.

To get next 7, need to sort

T(n)=2T(n/2)+nlogn

$O(n\log^2 n)$

$T(n)=2T(n/2)+n\log n$



for green point, we need to check these red points ?

Yes? Why?

7 points

so, for each point, we need to check at most next 7 points.

To get next 7, need to sort

2 dimensions, divide and conquer

- Split set of points into two halves by vertical line
- Recursively compute closest pair in left and right half
- Need to then compute closest pairs across separating line
- How can we do this efficiently?

Sorting points by x and y

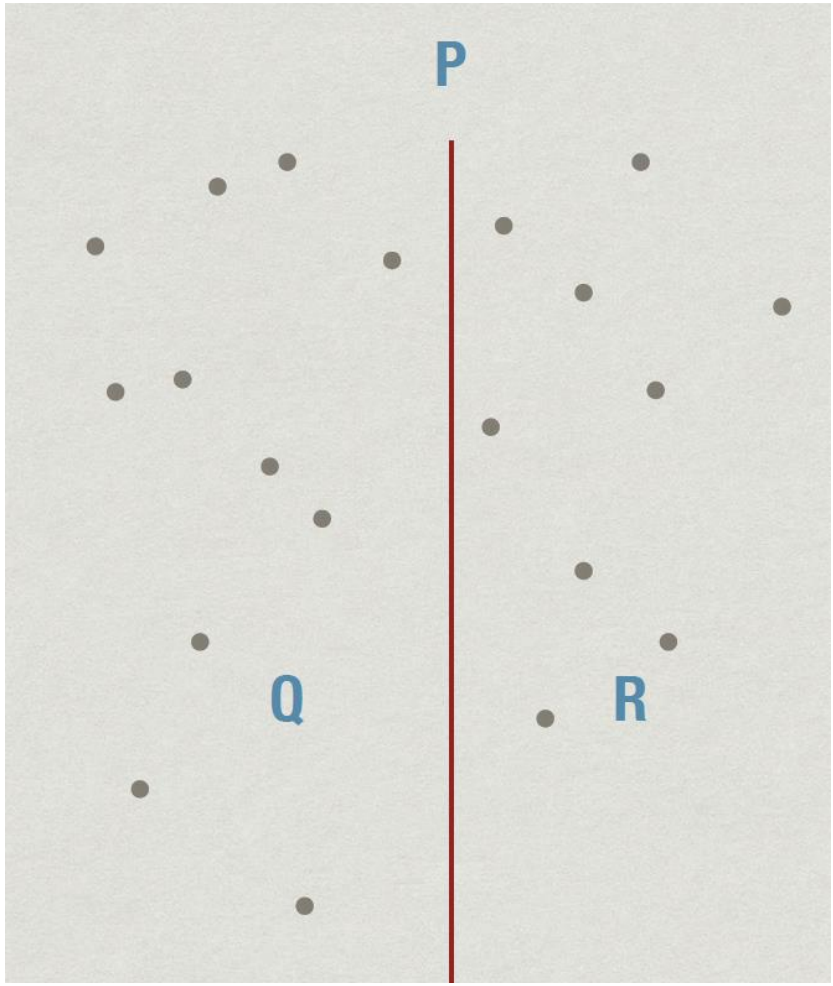
Given n points P = $\{p_1, p_2, \ldots, p_n\}$, compute

- P_x : P sorted by x coordinate
- P_y : P sorted by y coordinate

Divide P by vertical line into equal size sets Q and R.

Need to efficiently compute

Q_x, Q_y, R_x, R_y



Sorting points by x and y

- Need to efficiently compute

Q_x, Q_y, R_x, R_y

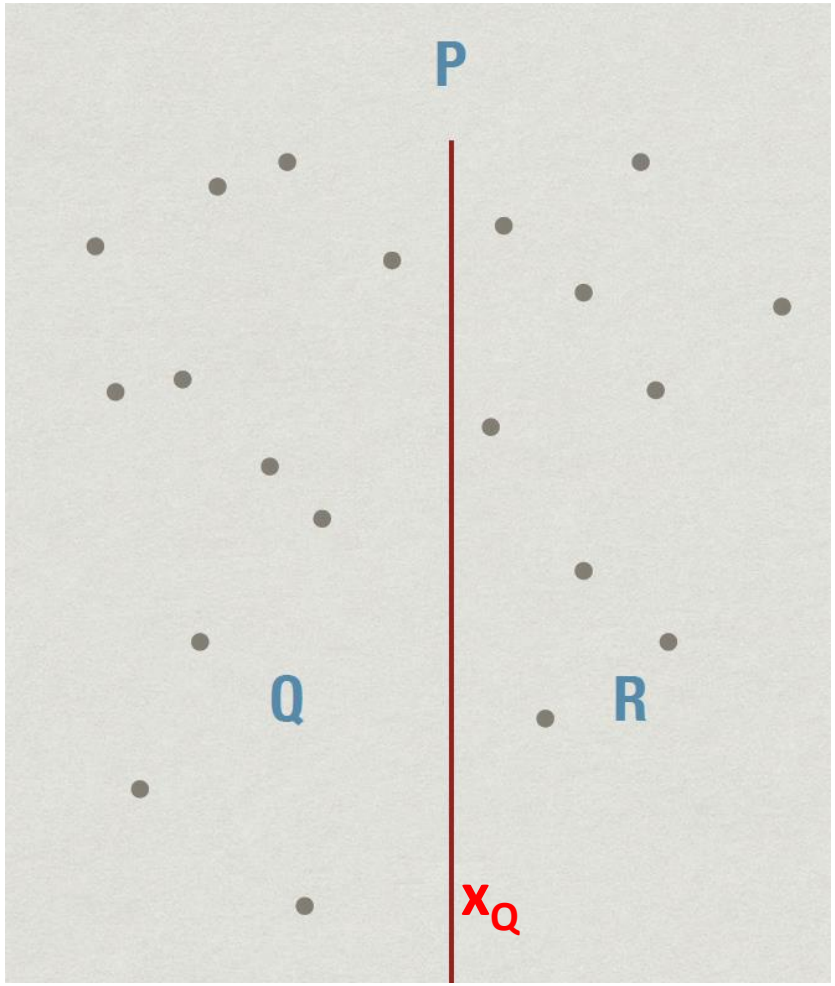
➢ Q_x is first half of P_x

➢ R_x is second half of P_x

- When splitting P_x,

note the largest x coordinate in Q, x_Q

- Separate P_y as Q_y, R_y by checking x coordinate with x_Q.
- All O(n)



2 dimensions, divide and conquer

- Basic recursive call is ClosestPair(P_x, P_y)

- Set up recursive calls ClosestPair(Q_x, Q_y)

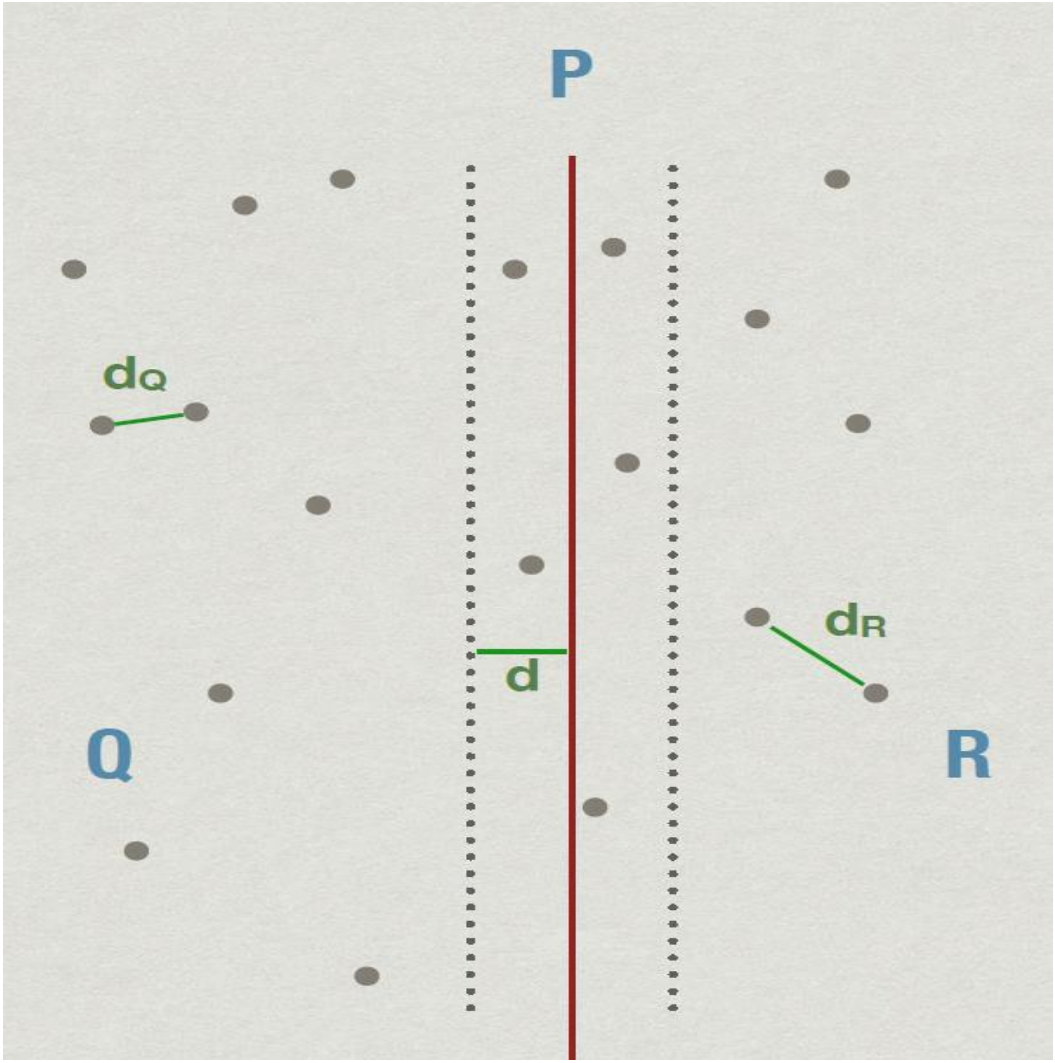
  and ClosestPair(R_x, R_y) for left and right half of P in time O(n)

- How to combine these recursive solutions?

Combining solutions

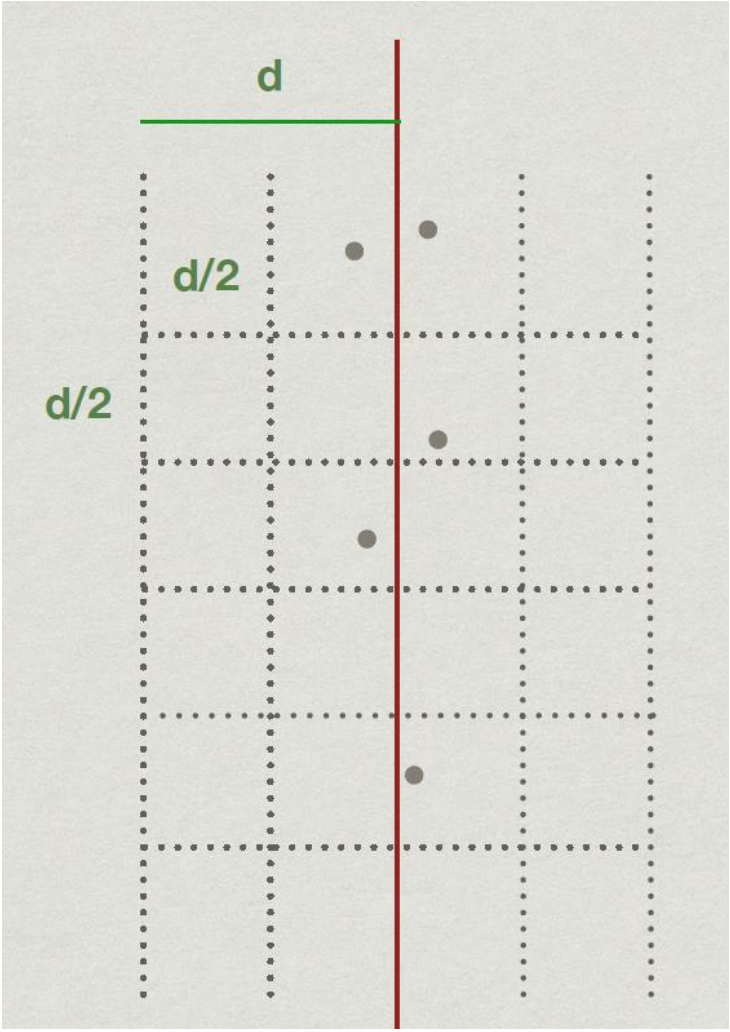
- Let d_Q be closest distance in Q and d_R be closest distance in R
- Let d be min(d_Q, d_R)
- Only need to consider points across the separator at most distance d from separator

➢ Any pair outside this strip cannot be closest pair overall



Combining solutions

- From Q_y, R_y, extract S_y,
  points in d-band sorted by y coordinate
- Scan S_y from bottom to top,
  comparing each point against next
  7 points in S_y
- Linear scan



Algorithm

ClosestPair(P_x,P_y)

if (|P_x| <= 3)

compute pairwise distances and

return the closest pair and distance

Construct (Q_x, Q_y, R_x, R_y)

(d_Q,q_1, q_2) = ClosestPair(Q_x, Q_y)

(d_R ,r_1, r_2) = ClosestPair(R_x, R_y)

Construct S_y and scan to find (d_S,s_1, s_2)

Return (d_Q,q_1, q_2) , (d_R ,r_1, r_2) , (d_S,s_1, s_2) depending on which among (d_Q,d_R, d_S) is minimum

Analysis

- Computing (P_x, P_y) from P takes O(n log n) (before recursion call)
- Recursive algorithm
  - Setting up (Q_x, Q_y, R_x, R_y) from (P_x, P_y) is O(n)
  - Setting up S_y from Q_y, R_y is O(n)
  - Scanning S_y is O(n)

$$T(n) = 2T(n/2) + O(n) = O(n \log n)$$