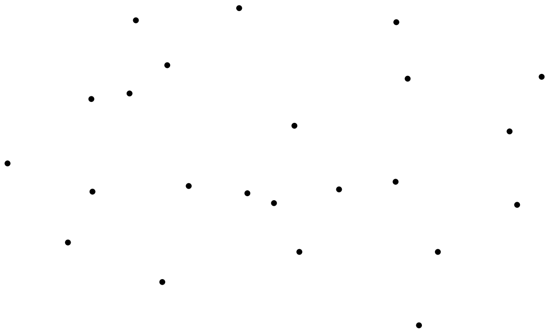
Closest Pair

Input: n points in plane: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

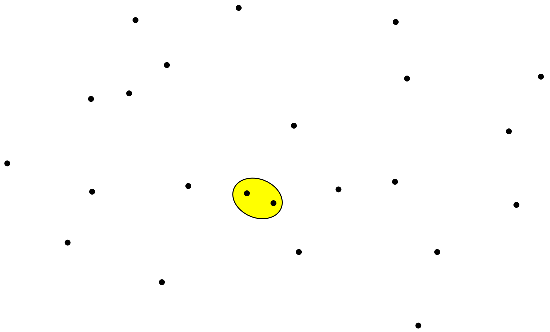
Output: the pair of points that are closest



Closest Pair

Input: n points in plane: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

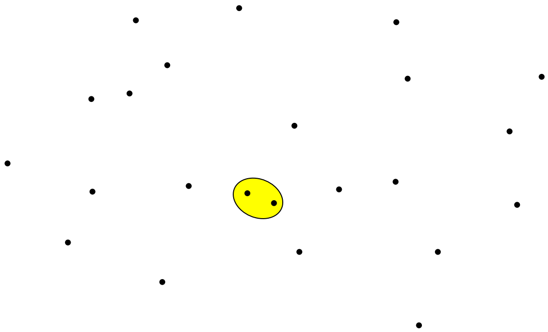
Output: the pair of points that are closest



Closest Pair

Input: n points in plane: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

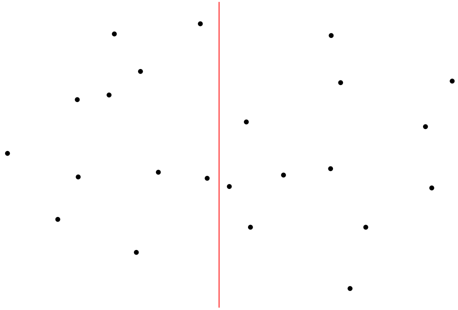
Output: the pair of points that are closest



- Trivial algorithm: $O(n^2)$ running time

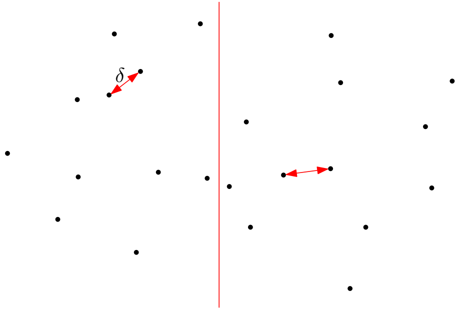
Divide-and-Conquer Algorithm for Closest Pair

- **Divide**: Divide the points into two halves via a vertical line



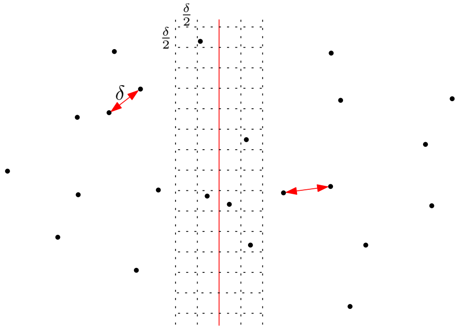
Divide-and-Conquer Algorithm for Closest Pair

- **Divide**: Divide the points into two halves via a vertical line
- **Conquer**: Solve two sub-instances recursively

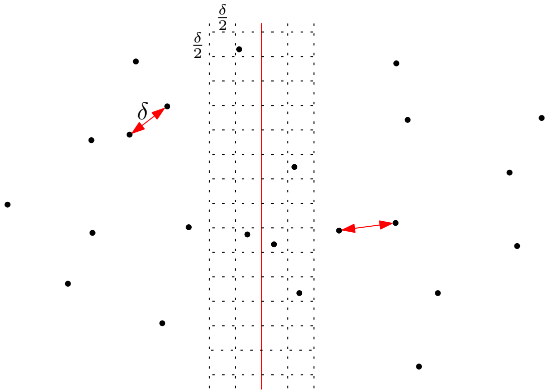


Divide-and-Conquer Algorithm for Closest Pair

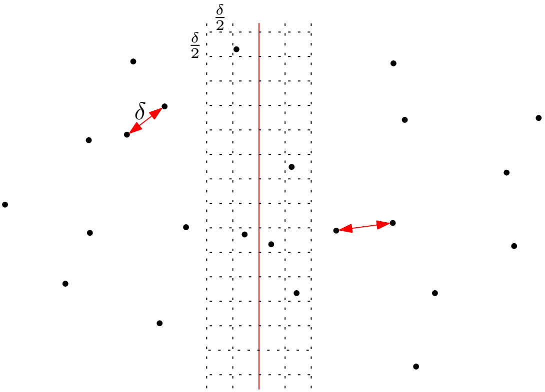
- **Divide**: Divide the points into two halves via a vertical line
- **Conquer**: Solve two sub-instances recursively
- **Combine**: Check if there is a closer pair between left-half and right-half



Divide-and-Conquer Algorithm for Closest Pair

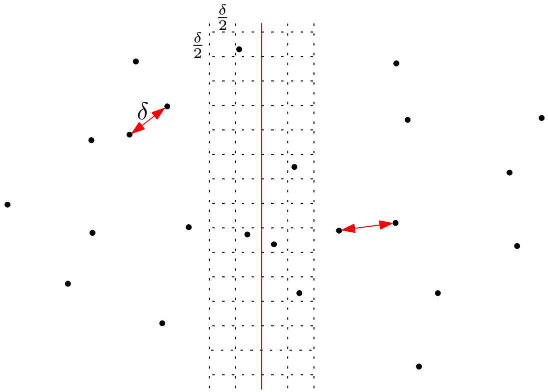


Divide-and-Conquer Algorithm for Closest Pair



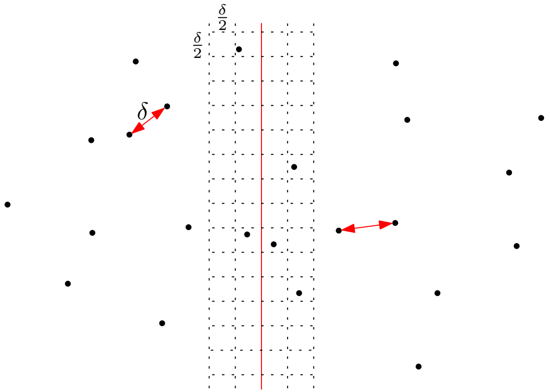
- Each box contains at most one pair

Divide-and-Conquer Algorithm for Closest Pair



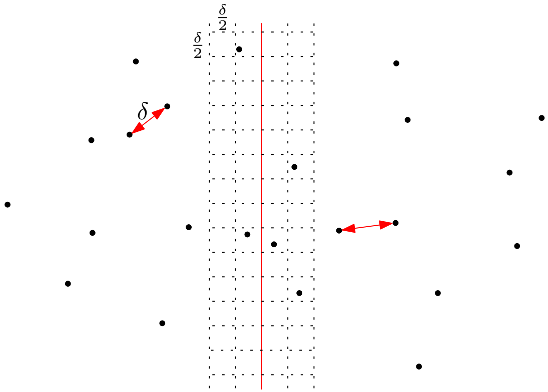
- Each box contains at most one pair
- For each point, only need to consider $O(1)$ boxes nearby

Divide-and-Conquer Algorithm for Closest Pair



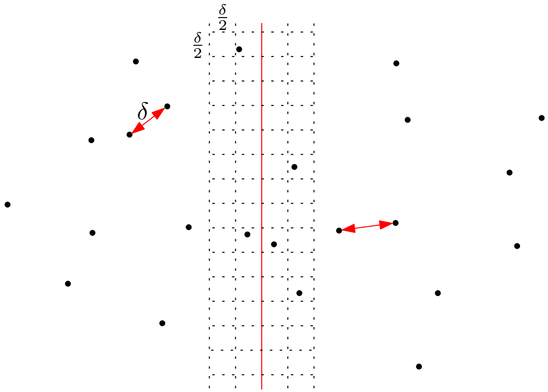
- Each box contains at most one pair
- For each point, only need to consider $O(1)$ boxes nearby
- time for combine $= O(n)$ (many technicalities omitted)

Divide-and-Conquer Algorithm for Closest Pair



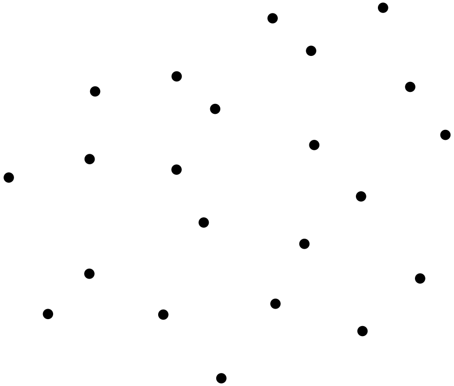
- Each box contains at most one pair
- For each point, only need to consider $O(1)$ boxes nearby
- time for combine $= O(n)$ (many technicalities omitted)
- Recurrence: $T(n) = 2T(n/2) + O(n)$

Divide-and-Conquer Algorithm for Closest Pair

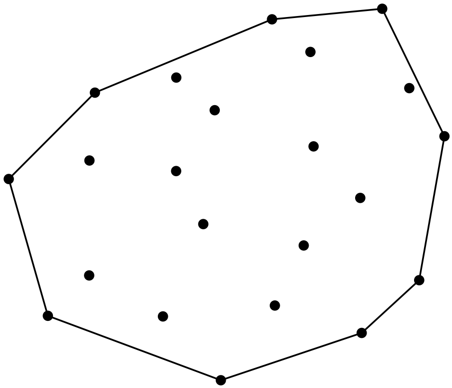


- Each box contains at most one pair
- For each point, only need to consider $O(1)$ boxes nearby
- time for combine $= O(n)$ (many technicalities omitted)
- Recurrence: $T(n) = 2T(n/2) + O(n)$
- Running time: $O(n \lg n)$

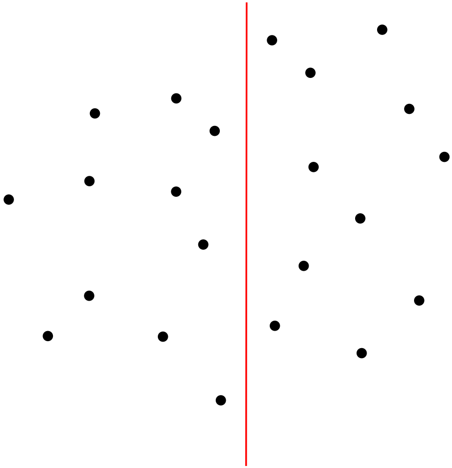
$O(n \lg n)$-Time Algorithm for Convex Hull



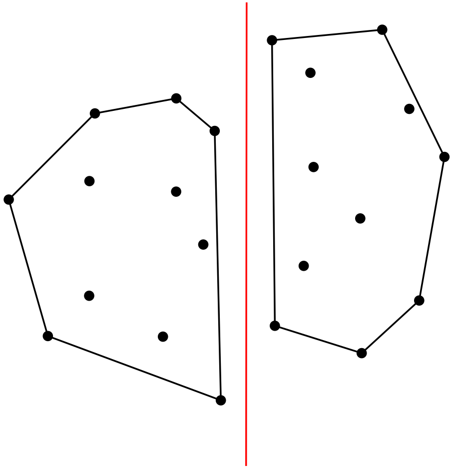
$O(n \lg n)$-Time Algorithm for Convex Hull



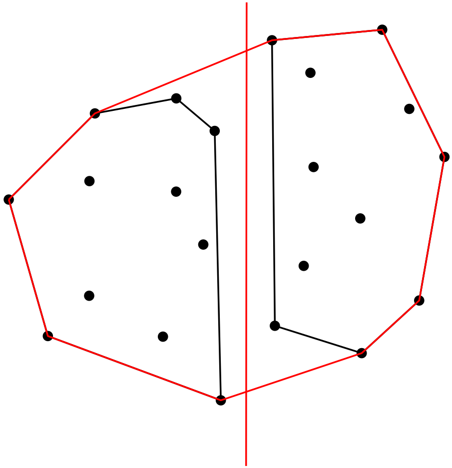
$O(n \lg n)$-Time Algorithm for Convex Hull



$O(n \lg n)$-Time Algorithm for Convex Hull



$O(n \lg n)$-Time Algorithm for Convex Hull



Strassen's Algorithm for Matrix Multiplication

Matrix Multiplication

Input: two $n \times n$ matrices A and B

Output: $C = AB$

Strassen's Algorithm for Matrix Multiplication

Matrix Multiplication

Input: two $n \times n$ matrices A and B

Output: $C = AB$

Naive Algorithm: matrix-multiplication(A, B, n)

```
for i ← 1 to n do
    for j ← 1 to n do
        C[i, j] ← 0
        for k ← 1 to n do
            C[i, j] ← C[i, j] + A[i, k] × B[k, j]
return C
```

Strassen's Algorithm for Matrix Multiplication

Matrix Multiplication

Input: two $n \times n$ matrices A and B

Output: $C = AB$

Naive Algorithm: matrix-multiplication(A, B, n)

```
for i ← 1 to n do
    for j ← 1 to n do
        C[i, j] ← 0
        for k ← 1 to n do
            C[i, j] ← C[i, j] + A[i, k] × B[k, j]
return C
```

- running time $= O(n^3)$

Try to Use Divide-and-Conquer

$$A = \begin{array}{|c|c|}\hline A_{11} & \overbrace{A_{12}}^{n/2} \\ \hline A_{21} & A_{22} \\ \hline \end{array}\left.\vphantom{A_{12}}\right\} n/2 \qquad B = \begin{array}{|c|c|}\hline B_{11} & \overbrace{B_{12}}^{n/2} \\ \hline B_{21} & B_{22} \\ \hline \end{array}\left.\vphantom{B_{12}}\right\} n/2$$

- $C = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$
- matrix_multiplication(A, B) recursively calls matrix_multiplication(A_{11}, B_{11}), matrix_multiplication(A_{12}, B_{21}), $\cdots$

Try to Use Divide-and-Conquer

$$A = \begin{array}{|c|c|}\hline A_{11} & \overbrace{A_{12}}^{n/2} \\ \hline A_{21} & A_{22} \\ \hline \end{array} \quad B = \begin{array}{|c|c|}\hline B_{11} & \overbrace{B_{12}}^{n/2} \\ \hline B_{21} & B_{22} \\ \hline \end{array}$$

- $C = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$
- matrix_multiplication(A, B) recursively calls matrix_multiplication(A_{11}, B_{11}), matrix_multiplication(A_{12}, B_{21}), $\cdots$
- Recurrence for running time: $T(n) = 8T(n/2) + O(n^2)$
- $T(n) = O(n^3)$

Strassen's Algorithm

- $T(n) = 8T(n/2) + O(n^2)$
- Strassen's Algorithm: improve the number of multiplications from 8 to 7!
- New recurrence: $T(n) = 7T(n/2) + O(n^2)$

Strassen's Algorithm

- $T(n) = 8T(n/2) + O(n^2)$
- Strassen's Algorithm: improve the number of multiplications from 8 to 7!
- New recurrence: $T(n) = 7T(n/2) + O(n^2)$
- Solving Recurrence $T(n) = O(n^{\log_2 7}) = O(n^{2.808})$