Matrix Multiplication

- Strassen's method gives O($n^{\log 7}$) time algorithm.
- It depends on multiplying two 2×2 matrices by 7 multiplications and 18 additions.
- Consider the matrices

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

- Let the result matrix be represented by

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

7 Scalar Products

- First compute the following products.

$$
\begin{aligned}
m_1 &= (a_{12} - a_{22})(b_{21} + b_{22}) \\
m_2 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\
m_3 &= (a_{11} - a_{21})(b_{11} + b_{12}) \\
m_4 &= (a_{11} + a_{12})b_{22} \\
m_5 &= a_{11}(b_{12} - b_{22}) \\
m_6 &= a_{22}(b_{21} - b_{11}) \\
m_7 &= (a_{21} + a_{22})b_{11}
\end{aligned}
$$

- Find the resultant elements as follows:

$$
\begin{aligned}
c_{11} &= m_1 + m_2 - m_4 + m_6 \\
c_{12} &= m_4 + m_5 \\
c_{21} &= m_6 + m_7 \\
c_{22} &= m_2 - m_3 + m_5 - m_7
\end{aligned}
$$

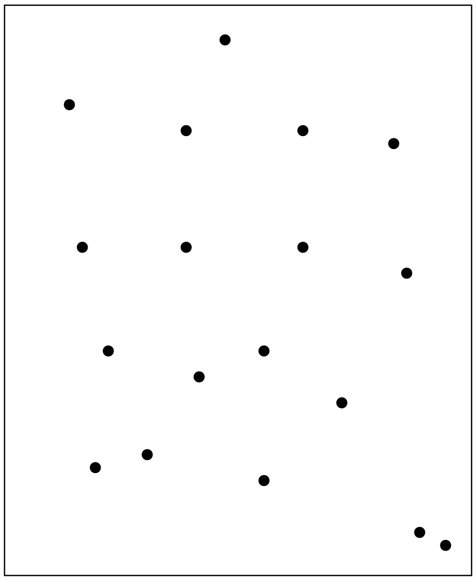
Strassen's Algorithm

- We can partition the each of the two $n \times n$ matrices into 4 $(n/2) \times (n/2)$ matrices each.
- Then by block matrix product rules we can find the resulting matrices.
- Let $T(n)$ be the number of arithmetic operations needed to multiply two $n \times n$ matrices.
- Then

$$T(n) = 7T(n/2) + 18(n/2)^2 \text{ for } n \geq 2$$

- Solving the recurrence $T(n) = O(7^{\log n})$ or $T(n) = O(n^{\log 7})$.

Closest Pair of Points

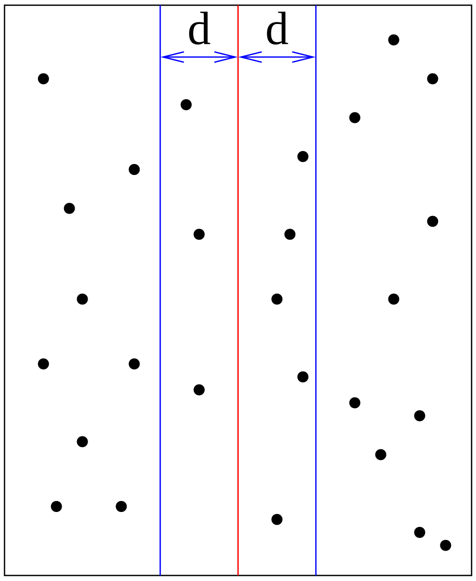


- Simplest attempt to solution will be
  - project the points on a line
  - sort the point-projections along the line
  - find the smallest distance.
- But projection will change the distances.

Closest Pair

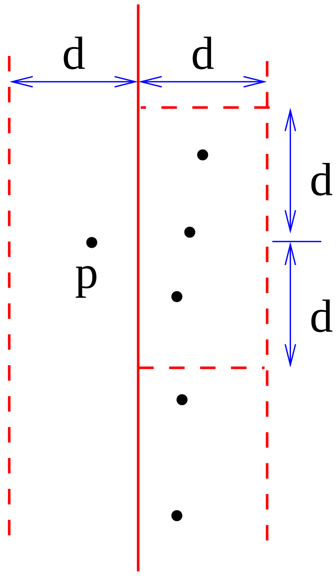
- Split the point set along a vertical line into two halves.
- Run `closestPair` algorithm recursively on each half
- Let d be the smallest of the distances from 2 recursive calls.
- Search along the boundary between two halves to find any pair of points that are closer than d.

Closest Pair



- While searching on the boundary we can ignore any point that is further than d from the partitioning line.
- That is we search only on the strip $2d$ strip on the boundary.

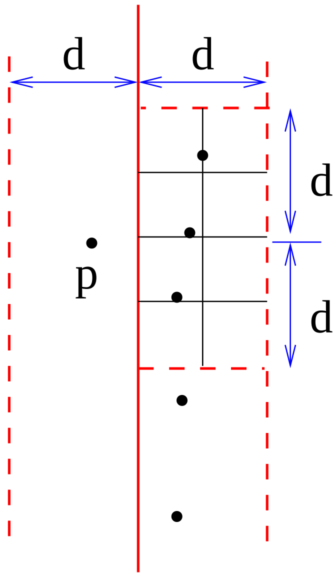
Closest Pair



- Consider a point p on the left half of the strip shown above.
- Corresponding to p we need to examine only 6 points on the right half of the strip.
- The points to be examined must lie within the rectangle as indicated.

- And this rectangle will have a maximum of 6 points.
- But how we must prove this sparsity condition.

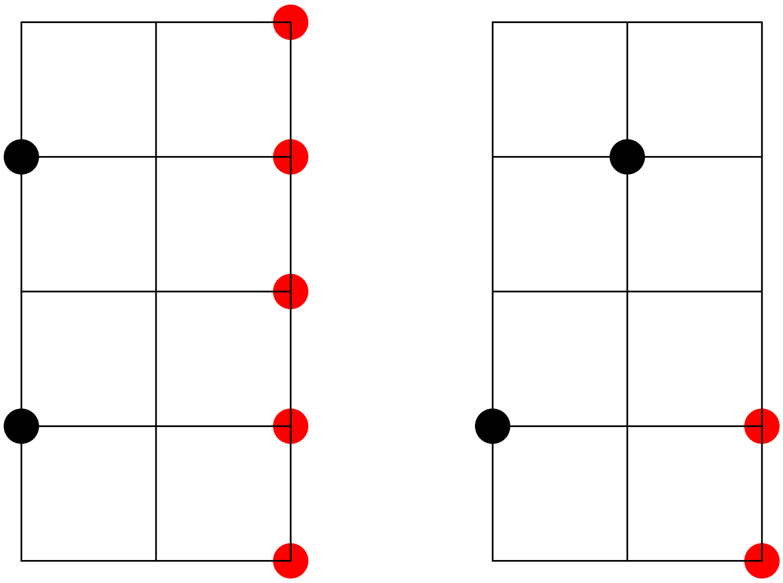
Proof of Sparsity condition



- Divide the rectangle into 8 squares of size $(d/2) \times (d/2)$.
- The length of the diagonal of the square is $d/\sqrt{2} < d$.
- Hence, no more than one point can exist in each of 8 squares.

- But the points of two adjacent middle squares either to the left or to the right must coincide. (Why?)

Proof of Sparsity condition



- Otherwise, the points in each such square must be placed in far corners, which will violate distance restriction with respect to points in other adjacent squares as figure indicates.
- Hence a maximum of 6 points need to be considered for each point p on the left.

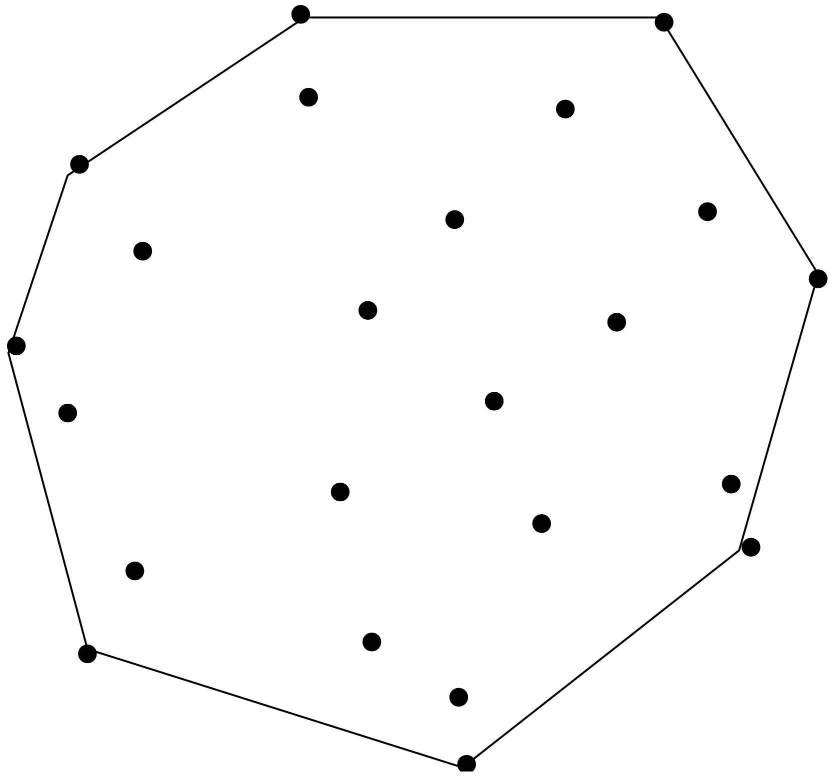
Closest Pair of Points

- We can presort the points in left half strip and do the walking over the sorted list.
- Since there can be at most n points in a half strip, the searching time for closest along the middle strip can require at most O(n) time.
- So the recurrence equation for closest pair

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

- Applying master theorem $T(n) = O(n \log n)$

Convex Hull



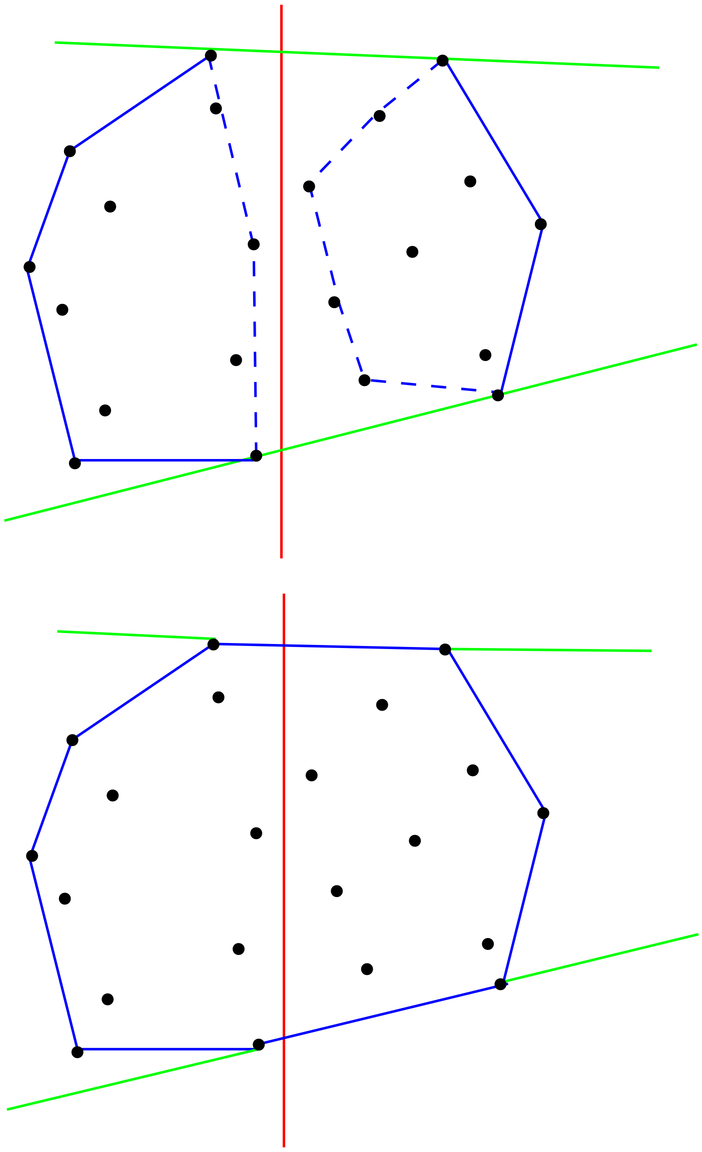
Convex Hull

- Is the smallest convex polygon containing all the points.
- Puts nails at each point
- Then enclose all the points within a rubber band by slipping it over them.
- The shape of the rubber band as it tightly stretches along the nails on boundary which defines a convex polygon.
- This polygon is the convex hull

Convex Hull

- A divide and conquer algorithm for finding convex hull proceeds as follows.
- Partion the set of points into two halves.
- Recursively find the convex hull of each half of points.
- Find upper and lower tangents of two convex hulls
- Delete the *inner chain* of each of the two convex hulls.
- The exterior chains and the two tangents define the convex hull for the full set of points.

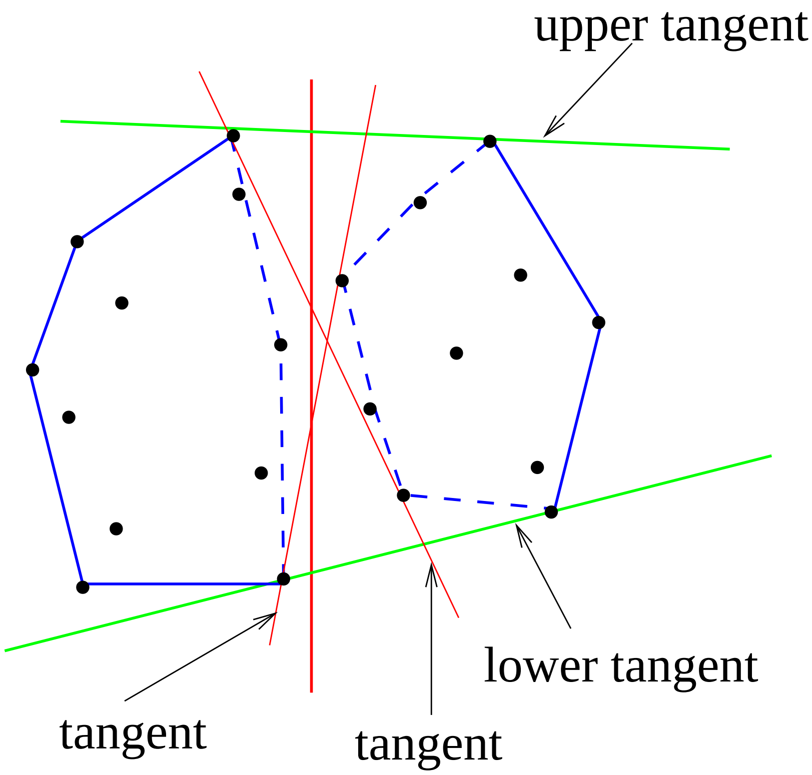
Convex Hull



Convex Hull

- How we divide the set of points in two halves?
  - Sort the set points according to x coordinates.
  - Find median of x values.
- What is an upper or a lower tangent?
  - Upper tangent – all points of both convex hulls are either on it or below it.
  - Lower tangent – all points of both convex hulls are either on it or above it.
- All other tangents have points from one hull above it, the points from other hull are below it.
- How such tangents can be found?
- What are the base case for termination of recursion?

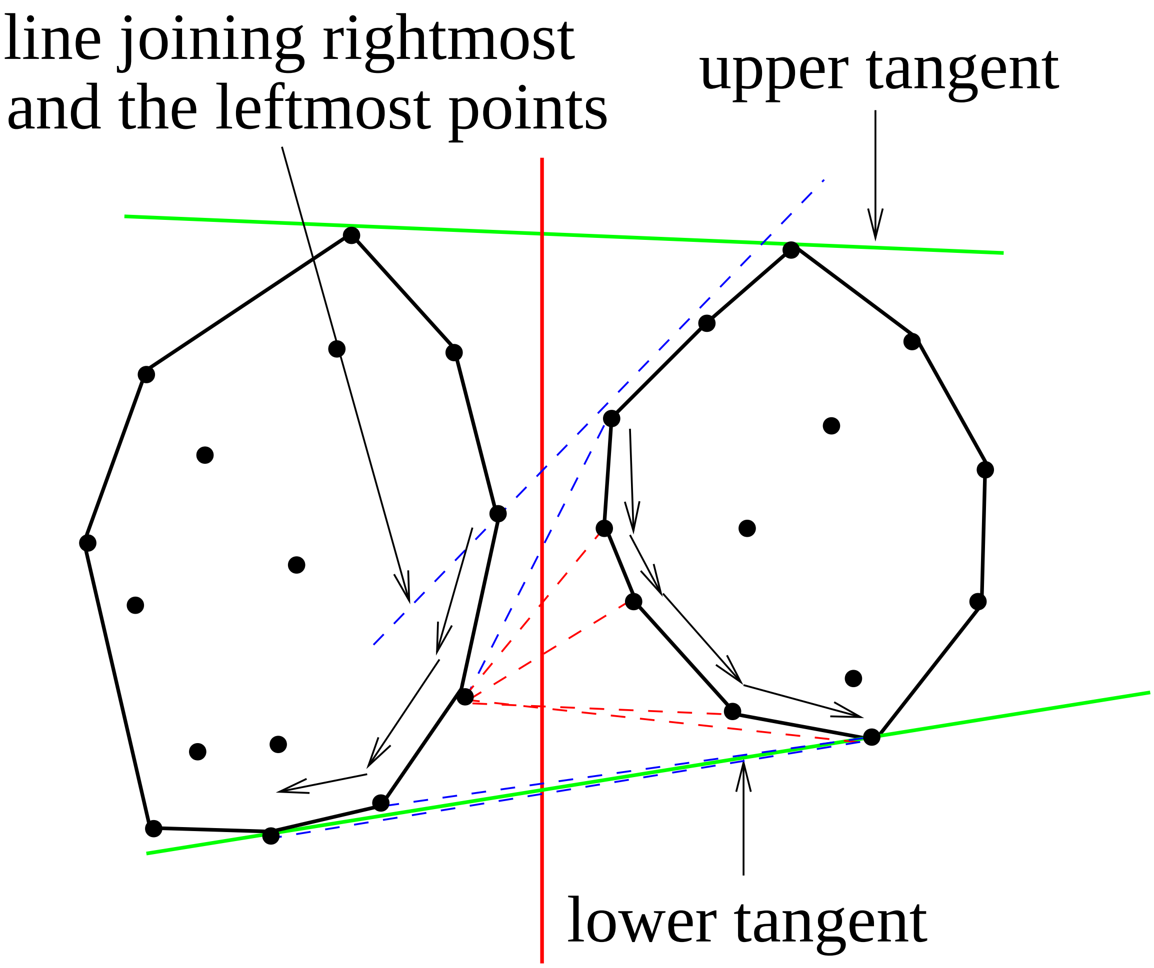
Convex Hull



Convex Hull

- Arrange the points in clock order in left hull.
- Arrange the points in anti-clock order in right hull.
- Find the right-most (maximum x coordinate) point, say p, of left hull
- Find the left-most (minimum x coordinate) point, say q, of right hull
- While $[p, q]$ is not the lower tangent of both hulls do
  - While $[p, q]$ is not a lower tangent of left hull let p be next point (clock-wise).
  - While $[p, q]$ is not a lower tangent of right hull let q be next point (anticlock-wise).
- Upper tangent can be found similarly by walking along the left inner chain in anti-clock and right inner chain in clock order.

Lower Tangent



red − iteration on points of right hull

blue − iteration on points of left hull

Running Time

- Initial sorting for partition – O($n \log n$).
- Recurrence for divide and conquer

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

- O(n) for merging by lower tangent and upper tangent.
  - Right-most and left-most points of hulls can be found in O(1) time as we already have sorted lists.
  - The exterior while loop for finding lower tangent cannot exceed O(n) time. Same for upper tangent.
- So running time is O($n \log n$).

Summary

- Divide and conquer paradigm decomposes the problem into subproblems of equal (reduced) sizes.
- Find solutions of each of the subproblems
- Combine the solutions of the subproblem to determine the solution of the whole.
- Examples considered are:
  - Quick sort,
  - Median finding,
  - Multiplications of integers,
  - Strassen's matrix multiplication,
  - Closest pair of points.
  - Convex hull.
- Many other problems: optimal BST, convex hull, etc can be solved by DAC.