Divide and Conquer / Dynamic Programming

Presentation by
V. Balasubramanian
SSN College of Engineering

Agenda

- Towers of Hanoi
- Factorial (Recursive / Iterative)
- Fibonacci series
- Divide and Conquer:
  - Strassen Multiplication,
  - Closest-Pair and Convex-Hull Problems
- Dynamic Programming:
  - Computing a Binomial Coefficient

Divide-and-Conquer

- The most-well known algorithm design strategy:
  - Divide instance of problem into two or more smaller instances
  - Solve smaller instances recursively
  - Obtain solution to original (larger) instance by combining these solutions

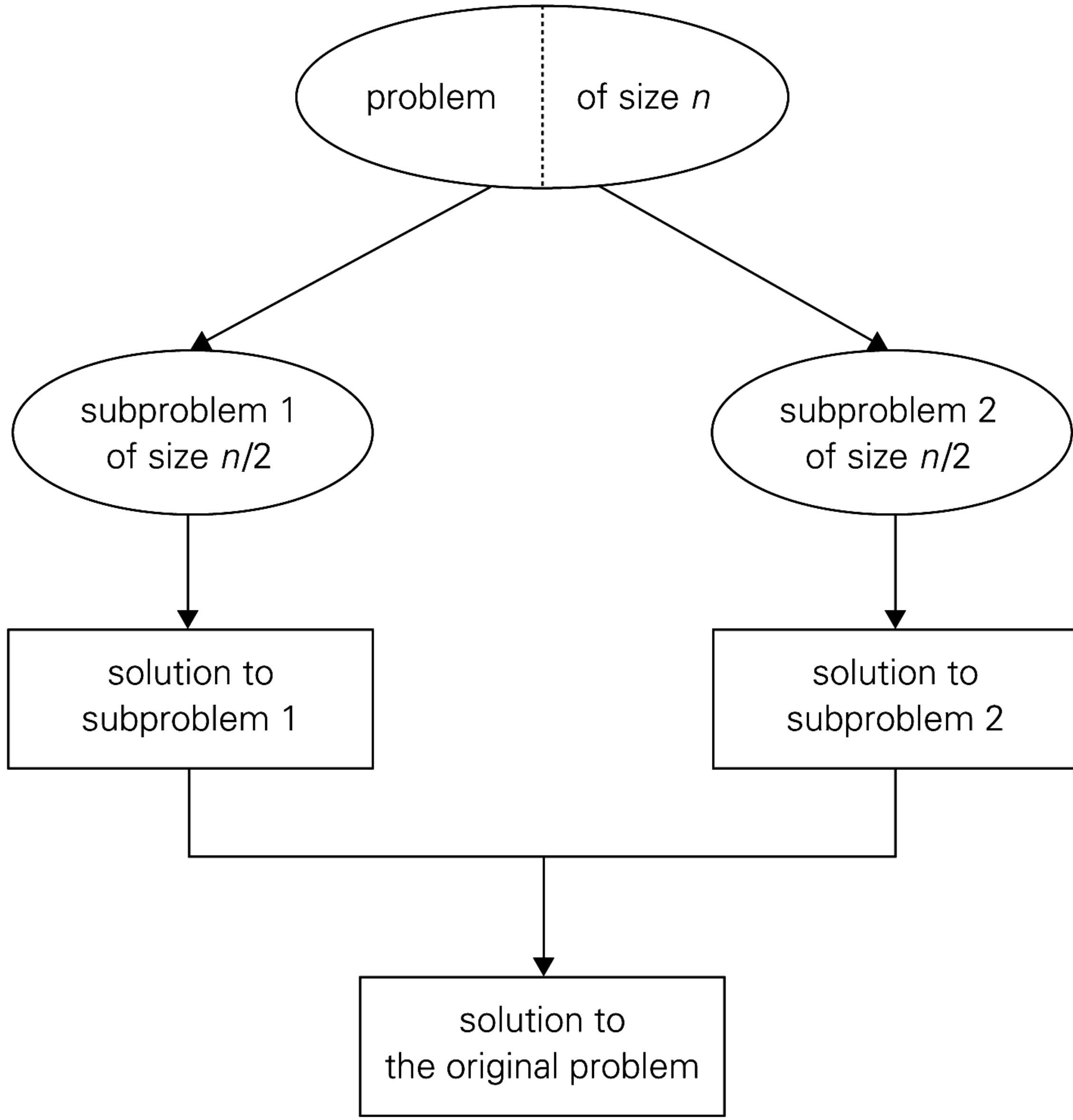


FIGURE 4.1 Divide-and-conquer technique (typical case)

Master's Theorem

$T(n) = aT(n/b) + f(n)$ where $f(n) \in \Theta(n^d)$, $d \geq 0$

Master Theorem:

If $a < b^d$, $T(n) \in \Theta(n^d)$

If $a = b^d$, $T(n) \in \Theta(n^d \log n)$

If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

Integer Multiplication

Consider the problem of multiplying two (large) *n*-digit integers represented by arrays of their digits such as:

A = 12345678901357986429 B = 87654321284820912836

Example

- High Precision computing
- PC 64 bit computation
- Weight of Neutrino 100 decimal digits.
- Square root
- PI for million digits
- RSA primes with 1000’s of bit long.

Example

The grade-school algorithm:

$$
\begin{array}{rrrrr}
 & a_1 & a_2 & \ldots & a_n \\
 & b_1 & b_2 & \ldots & b_n \\
\hline
(d_{10}) & d_{11} & d_{12} & \ldots & d_{1n} \\
(d_{20}) & d_{21} & d_{22} & \ldots & d_{2n} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\hline
(d_{n0}) & d_{n1} & d_{n2} & \ldots & d_{nn}
\end{array}
$$

Efficiency: n^2 one-digit multiplications

Contd…

- If you multiply 2 1000 digit number, then 1000,000 one digit multiplication needs to be done.
- Cryptography

Analysis

- *Conventional Multiplication reuires* n^2 digit multiplications

Anatoly Karatsuba

- Russian mathematician
- 1960
- 23 years
- Karatsuba Multiplication

Divide and Conquer

- split an n-digit integer into two integers of approximately n/2 bits

$$\underbrace{567{,}832}_{6\text{ digits}} = \underbrace{567}_{3\text{ digits}} \times 10^3 + \underbrace{832}_{3\text{ digits}}$$

$$\underbrace{9{,}423{,}723}_{7\text{ digits}} = \underbrace{9423}_{4\text{ digits}} \times 10^3 + \underbrace{723}_{3\text{ digits}}$$

Example

- 965107x102635
- Normal multiplication = n^2 digit multiplication = 36
- For 1000 digit number = 1000,000

Contd…

$$\underbrace{u}_{n \text{ digits}} = \underbrace{x}_{\lceil n/2 \rceil \text{ digits}} \times 10^m + \underbrace{y}_{\lfloor n/2 \rfloor \text{ digits}}$$

$$m = \left\lfloor \frac{n}{2} \right\rfloor.$$

If we have two n-digit integers

$$u = x \times 10^m + y$$
$$v = w \times 10^m + z,$$

their product is given by

$$\begin{aligned} uv &= (x \times 10^m + y)(w \times 10^m + z) \\ &= xw \times 10^{2m} + (xz + wy) \times 10^m + yz. \end{aligned}$$

U,V large integers

$m = \lfloor n/2 \rfloor$;

$x = u$ **divide** 10^m; $y = u$ **rem** 10^m;

$w = v$ **divide** 10^m; $z = v$ **rem** 10^m;

return $prod(x,w) \times 10^{2m} + (prod(x,z) + prod(w,y)) \times 10^m + prod(y,z)$;

Analysis

$$W(n) = 4W\left(\frac{n}{2}\right) + cn \qquad \text{for } n > s,\ n \text{ a power of } 2$$

$$W(s) = 0.$$

$$W(n) \in \Theta(n^{\lg 4}) = \Theta(n^2).$$

Contd…

- $U = XY = X * 10^{n/2} + Y$
- $V = WZ = W * 10^{n/2} + Z$
- $UV = XW\ 10^{n} + YZ + (XZ+YW)\ 10^{n/2}$

$$xw, \qquad xz + yw, \qquad \text{and} \qquad yz,$$

$$r = (x + y)(w + z) = xw + (xz + yw) + yz,$$

$$xz + yw = r - xw - yz.$$

Algorithm

$m = \lfloor n/2 \rfloor;$

$x = u$ **divide** 10^m; $y = u$ **rem** 10^m;

$w = v$ **divide** 10^m; $z = v$ **rem** 10^m;

$r = prod2(x + y, w + z);$

$p = prod2(x, w);$

$q = prod2(y, z);$

return $p \times 10^{2m} + (r - p - q) \times 10^m + q;$

Analysis

$$M(n) = 3M(n/2) \quad \text{for } n > 1, \quad M(1) = 1.$$

Solving it by backward substitutions for $n = 2^k$ yields

$$\begin{aligned} M(2^k) &= 3M(2^{k-1}) = 3[3M(2^{k-2})] = 3^2 M(2^{k-2}) \\ &= \cdots = 3^i M(2^{k-i}) = \cdots = 3^k M(2^{k-k}) = 3^k. \end{aligned}$$

Since $k = \log_2 n$,

$$M(n) = 3^{\log_2 n} = n^{\log_2 3} \approx n^{1.585}.$$

Analysis

- 1000 digit number no of operations = 56000.

Further Faster algorithms

- Fast Fourier Transforms, Borodin and Munro (1975) developed a
- $0(n(\lg n)^2)$ algorithm for multiplying large integers.

Exercise

- What are the smallest and largest numbers of digits the product of two decimal n-digit integers can have?

Solution

- Smallest n digit number = 10(n-1 times) = 10^{n-1}
- $10^{n-1} * 10^{n-1} = 10^{2n-2}$ =2n-1digits = 1 followed by 2n-2 zeros
- Largest n digit number = 9999(n times) = 10^n-1
- $10^n-1*10^n-1 = 10^{2n}-1$ = 2n digits

a=a1a0, b=b1b0

$$c = a * b = (a_1 10^{n/2} + a_0) * (b_1 10^{n/2} + b_0)$$
$$= (a_1 * b_1)10^n + (a_1 * b_0 + a_0 * b_1)10^{n/2} + (a_0 * b_0)$$
$$= c_2 10^n + c_1 10^{n/2} + c_0,$$

$c_2 = a_1 * b_1$ is the product of their first halves,

$c_0 = a_0 * b_0$ is the product of their second halves,

$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$ is the product of the sum of the a's halves and the sum of the b's halves minus the sum of c_2 and c_0.

Exercise

- Compute 2101 * 1130 by applying the divide-and-conquer algorithm outlined.

Solution

For $2101 * 1130$:

$$
\begin{aligned}
c_2 &= 21 * 11 \\
c_0 &= 01 * 30 \\
c_1 &= (21 + 01) * (11 + 30) - (c_2 + c_0) = 22 * 41 - 21 * 11 - 01 * 30.
\end{aligned}
$$

Contd…

For $21 * 11$:

$$\begin{aligned} c_2 &= 2 * 1 = 2 \\ c_0 &= 1 * 1 = 1 \\ c_1 &= (2+1) * (1+1) - (2+1) = 3 * 2 - 3 = 3. \\ \text{So, } 21 * 11 &= 2 \cdot 10^2 + 3 \cdot 10^1 + 1 = 231. \end{aligned}$$

For $01 * 30$:

$$\begin{aligned} c_2 &= 0 * 3 = 0 \\ c_0 &= 1 * 0 = 0 \\ c_1 &= (0+1) * (3+0) - (0+0) = 1 * 3 - 0 = 3. \\ \text{So, } 01 * 30 &= 0 \cdot 10^2 + 3 \cdot 10^1 + 0 = 30. \end{aligned}$$

Contd…

For $22 * 41$:

$$\begin{aligned} c_2 &= 2 * 4 = 8 \\ c_0 &= 2 * 1 = 2 \\ c_1 &= (2+2) * (4+1) - (8+2) = 4 * 5 - 10 = 10. \\ \text{So, } 22 * 41 &= 8 \cdot 10^2 + 10 \cdot 10^1 + 2 = 902. \end{aligned}$$

Hence

$$2101 * 1130 = 231 \cdot 10^4 + (902 - 231 - 30) \cdot 10^2 + 30 = 2,374,130.$$

Strassen's Multiplication

$$\begin{array}{cc} & \overset{\longleftarrow n/2 \longrightarrow}{} \\ n/2 & \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right] \end{array} = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \times \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right]$$

Idea

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

Levitin

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

Strassen

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

he product C is given by

$$C = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}.$$

Example

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 8 & 9 & 1 & 2 \\ 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 \end{bmatrix}.$$

Contd...

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\
&= \left(\begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \right) \times \left(\begin{bmatrix} 8 & 9 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 9 & 1 \\ 4 & 5 \end{bmatrix} \right) \\
&= \begin{bmatrix} 3 & 5 \\ 11 & 13 \end{bmatrix} \times \begin{bmatrix} 17 & 10 \\ 7 & 9 \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
M_1 &= \begin{bmatrix} 3 & 5 \\ 11 & 13 \end{bmatrix} \times \begin{bmatrix} 17 & 10 \\ 7 & 9 \end{bmatrix} \\
&= \begin{bmatrix} 3 \times 17 + 5 \times 7 & 3 \times 10 + 5 \times 9 \\ 11 \times 17 + 13 \times 7 & 11 \times 10 + 13 \times 9 \end{bmatrix} = \begin{bmatrix} 86 & 75 \\ 278 & 227 \end{bmatrix}
\end{aligned}
$$

Exercise

$$\begin{bmatrix} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{bmatrix}$$

Solution

$$C = \left[\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right] = \left[\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right] \left[\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right]$$

where

$$A_{00} = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}, \quad A_{01} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_{10} = \begin{bmatrix} 0 & 1 \\ 5 & 0 \end{bmatrix}, \quad A_{11} = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix},$$

$$B_{00} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}, \quad B_{01} = \begin{bmatrix} 0 & 1 \\ 0 & 4 \end{bmatrix}, \quad B_{10} = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}, \quad B_{11} = \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix}.$$

Contd…

$$
\begin{aligned}
M_1 &= (A_{00} + A_{11})(B_{00} + B_{11}) = \begin{bmatrix} 4 & 0 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix}, \\
M_2 &= (A_{10} + A_{11})B_{00} = \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix}, \\
M_3 &= A_{00}(B_{01} - B_{11}) = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -5 & 4 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix}, \\
M_4 &= A_{11}(B_{10} - B_{00}) = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix},
\end{aligned}
$$

Contd...

$$
\begin{aligned}
M_5 &= (A_{00} + A_{01})B_{11} = \begin{bmatrix} 3 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix}, \\
M_6 &= (A_{10} - A_{00})(B_{00} + B_{01}) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}, \\
M_7 &= (A_{01} - A_{11})(B_{10} + B_{11}) = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}.
\end{aligned}
$$

Contd...

$$
\begin{aligned}
C_{00} &= M_1 + M_4 - M_5 + M_7 \\
&= \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}, \\
C_{01} &= M_3 + M_5 \\
&= \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} + \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} = \begin{bmatrix} 7 & 3 \\ 1 & 9 \end{bmatrix}, \\
C_{10} &= M_2 + M_4 \\
&= \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ 5 & 8 \end{bmatrix}, \\
C_{11} &= M_1 + M_3 - M_2 + M_6 \\
&= \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} - \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ 7 & 7 \end{bmatrix}.
\end{aligned}
$$

Solution

$$C = \begin{bmatrix} 5 & 4 & 7 & 3 \\ 4 & 5 & 1 & 9 \\ 8 & 1 & 3 & 7 \\ 5 & 8 & 7 & 7 \end{bmatrix}.$$

Analysis

$$M(n) = 7M(n/2) \quad \text{for } n > 1, \quad M(1) = 1.$$

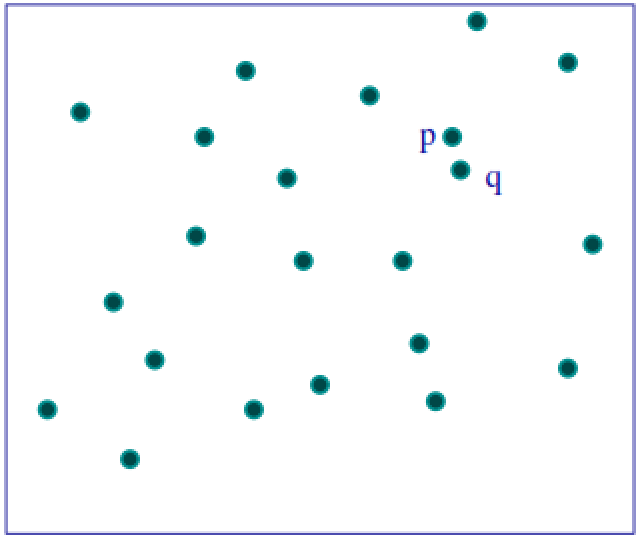
Since $n = 2^k$,

$$M(2^k) = 7M(2^{k-1}) = 7[7M(2^{k-2})] = 7^2M(2^{k-2}) = \cdots$$
$$= 7^iM(2^{k-i}) \cdots = 7^kM(2^{k-k}) = 7^k.$$

Since $k = \log_2 n$,

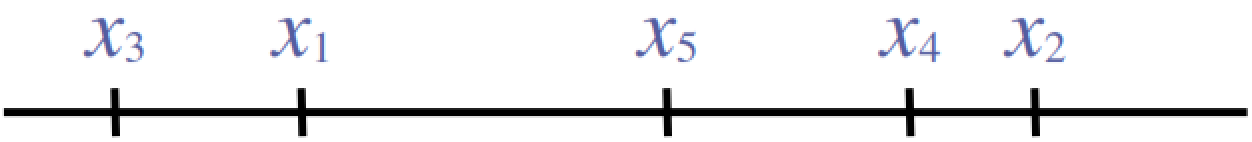
$$M(n) = 7^{\log_2 n} = n^{\log_2 7} \approx n^{2.807},$$

Closest Pair Problem



- Complexity = n^2
- For example, an air-traffic controller might be interested in two closest planes as the most probable collision candidates.
- A regional postal service manager - closest pair problem to find candidate post-office locations to be closed.

1D solution



Algorithm

- If n>3, we can divide the points into two subsets P_l and P_r of n/2 and n/2 points, respectively, by drawing a vertical line through the median m of their x coordinates so that n/2 points lie to the left of or on the line itself, and n/2 points lie to the right of or on the line. Then we can solve the closest-pair problem

Algorithm

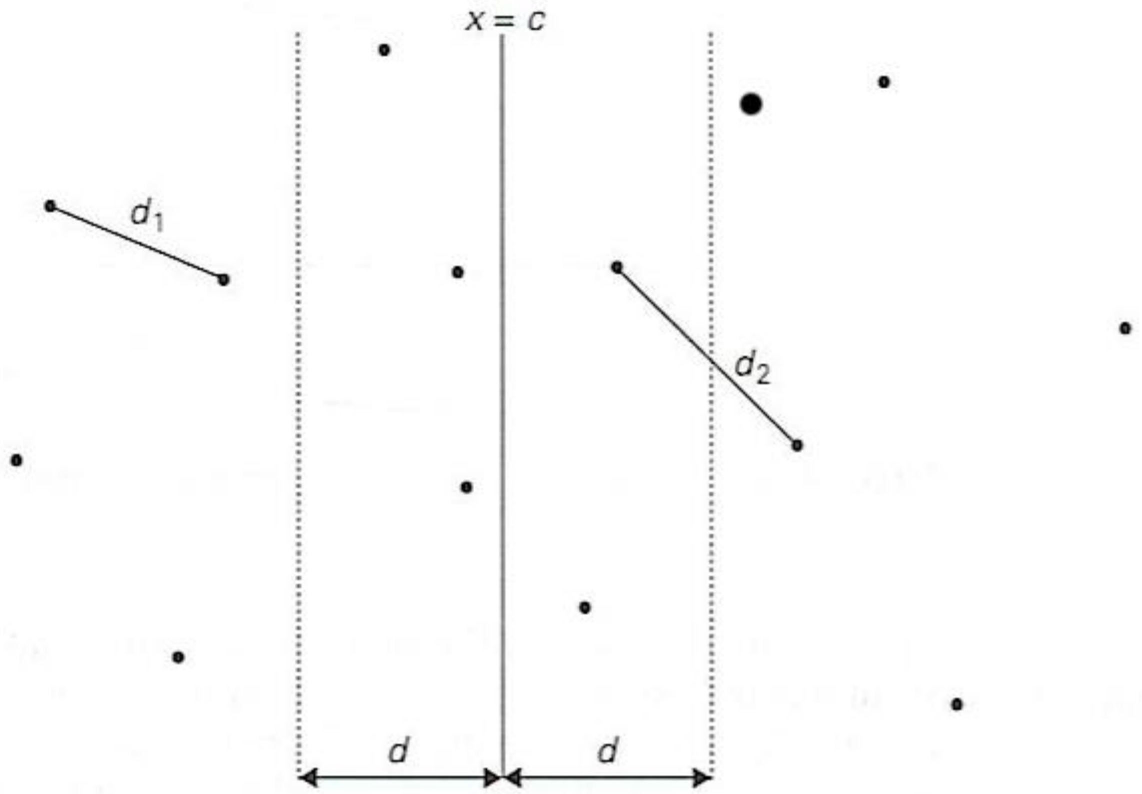
The algorithm:

- Input: A set S of n planar points.
- Output: The distance between two closest points.

Steps

Step 1 Divide the points given into two subsets S_1 and S_2 by a vertical line $x = c$ so that half the points lie to the left or on the line and half the points lie to the right or on the line.

Contd…



Step 2

Step 2: Find recursively the closest pairs for the left and right subsets.

Step 3

- Step 3:Set d = min{d1, d2}

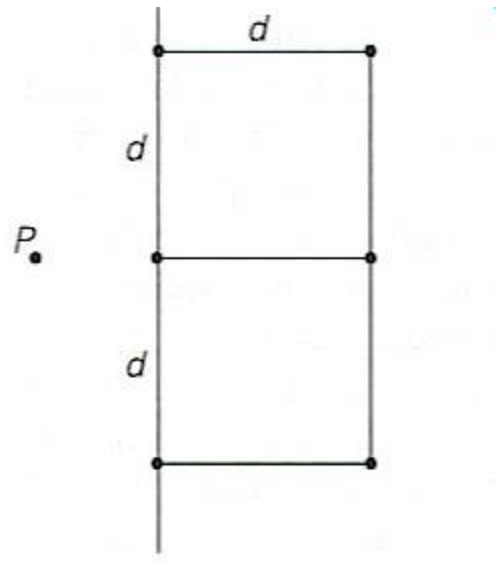
We can limit our attention to the points in the symmetric vertical strip of width 2d as possible closest pair. Let C1 and C2 be the subsets of points in the left subset S1 and of the right subset S2, respectively, that lie in this vertical strip. The points in C1 and C2 are stored in increasing order of their y coordinates, which is maintained by merging during the execution of the next step.

Step 4

- Step 4:

For every point $P(x,y)$ in C_1, we inspect points in C_2 that may be closer to P than d. There can be no more than 6 such points (because $d \leq d_2$)!

Worst Case Scenario



Contd…

Step 1: Sort points in S according to their y-values.

Step 2: If S contains only one point, return infinity as its distance.

Step 3: Find a median line L perpendicular to the X-axis to divide S into S_L and S_R, with equal sizes.

Contd…

- Step 4: Recursively apply Steps 2 and 3 to solve the closest pair problems of S_L and S_R. Let $d_L(d_R)$ denote the distance between the closest pair in S_L (S_R). Let $d = \min(d_L, d_R)$.

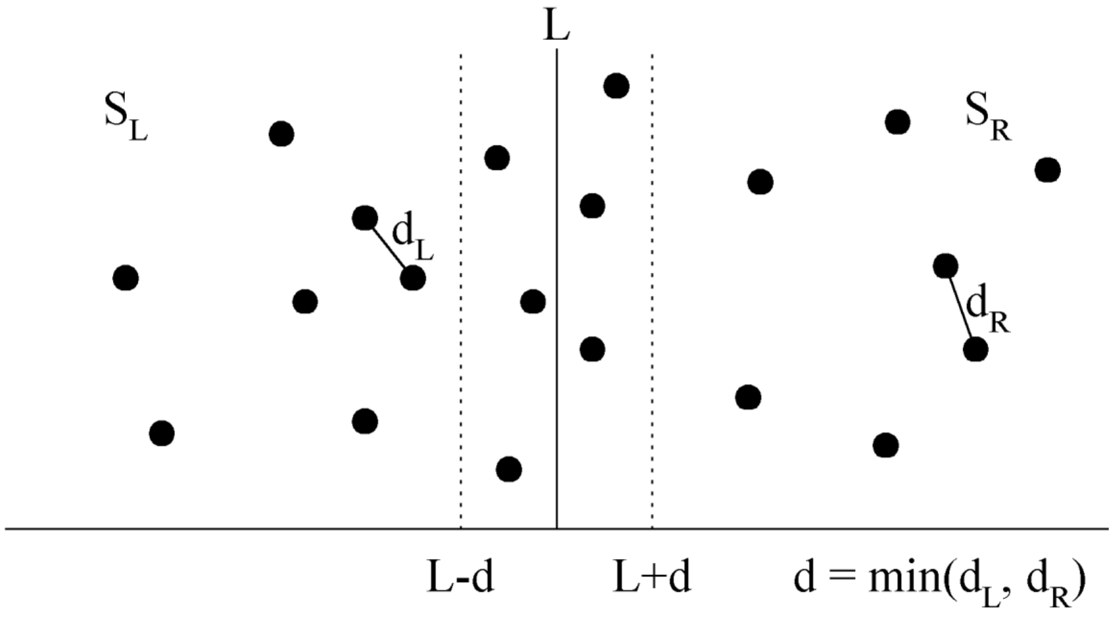
Contd…

- Step 5: For a point P in the half-slab bounded by L-d and L, let its y-value be denoted as y_P . For each such P, find all points in the half-slab bounded by L and L+d whose y-value fall within y_P+d and y_P-d. If the distance d' between P and a point in the other half-slab is less than d, let $d=d'$. The final value of d is the answer.

```
ALGORITHM  EfficientClosestPair(P, Q)
    //Solves the closest-pair problem by divide-and-conquer
    //Input: An array P of n ≥ 2 points in the Cartesian plane sorted in
    //       nondecreasing order of their x coordinates and an array Q of the
    //       same points sorted in nondecreasing order of the y coordinates
    //Output: Euclidean distance between the closest pair of points
    if n ≤ 3
        return the minimal distance found by the brute-force algorithm
    else
        copy the first ⌈n/2⌉ points of P to array P_l
        copy the same ⌈n/2⌉ points from Q to array Q_l
        copy the remaining ⌊n/2⌋ points of P to array P_r
        copy the same ⌊n/2⌋ points from Q to array Q_r
        d_l ← EfficientClosestPair(P_l, Q_l)
        d_r ← EfficientClosestPair(P_r, Q_r)
        d ←min{d_l, d_r}
        m ← P[⌈n/2⌉ − 1].x
        copy all the points of Q for which |x − m| < d into array S[0..num − 1]
        dminsq ← d^2
        for i ← 0 to num − 2 do
            k ← i + 1
            while k ≤ num − 1 and (S[k].y − S[i].y)^2 < dminsq
                dminsq ← min((S[k].x − S[i].x)^2+ (S[k].y − S[i].y)^2, dminsq)
                k ← k + 1
    return sqrt(dminsq)
```

Contd…

$$T(n) = \begin{cases} 2T(n/2)+O(n)+O(n) & , n > 1 \\ 1 & , n = 1 \end{cases}$$


L
S_L
S_R
d_L
d_R
L-d
L+d
d = min(d_L, d_R)

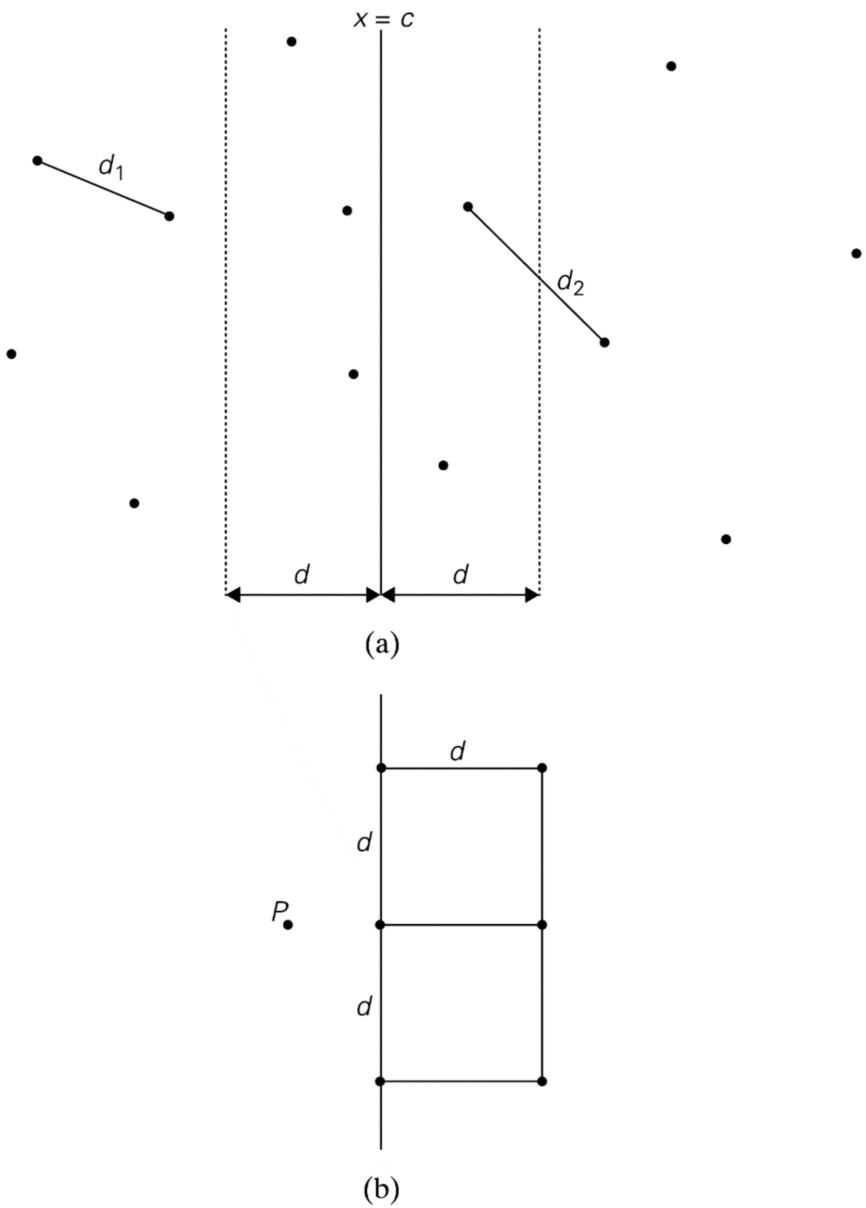


FIGURE 4.7 (a) Idea of the divide-and-conquer algorithm for the closest-pair problem.
(b) The six points that may need to be examined for point P.

Analysis

Running time of the algorithm is described by

$$T(n) = 2T(n/2) + M(n), \text{ where}$$

$M(n) \in O(n)$

By the Master Theorem (with $a = 2$, $b = 2$, $d = 1$)

$$T(n) \in O(n \log n)$$

Exercise

1. **a.** For the one-dimensional version of the closest-pair problem, i.e., for the problem of finding two closest numbers among a given set of n real numbers, design an algorithm that is directly based on the divide-and-conquer technique and determine its efficiency class.

   b. Is it a good algorithm for this problem?

Solution

- O(n log n)
- Without employing Divide and Conquer, we can sort it and do it.

    O(n log n)

Exercise

2. Prove that the divide-and-conquer algorithm for the closest-pair problem examines, for every point p in the vertical strip (see Figures 5.7a and 5.7b), no more than seven other points that can be closer to p than d_{min}, the minimum distance between two points encountered by the algorithm up to that point.

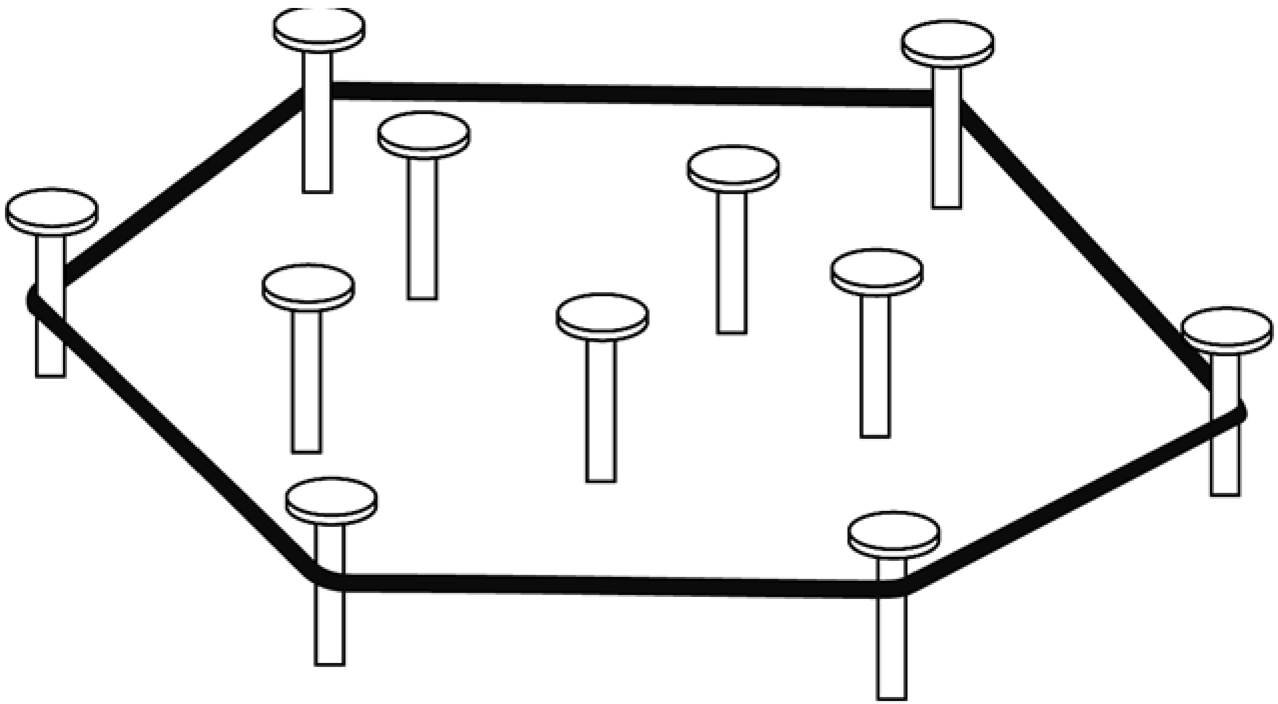
Convex Hull

- A shape or set is convex if for any two points that are part of the shape, the whole connecting line segment is also part of the shape.

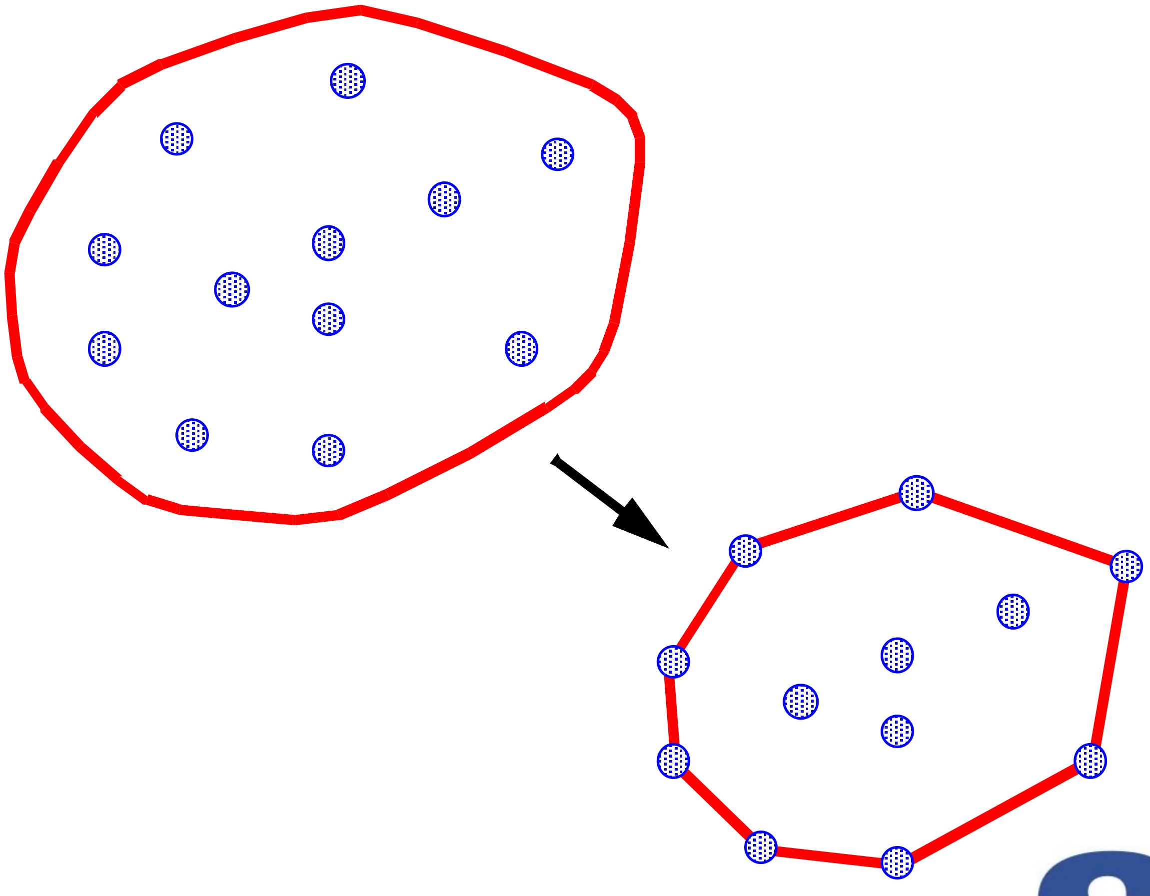
What is Convex Hull?

- Let S be a set of points in the plane.
- Intuition: Imagine the points of S as being pegs; the convex hull of S is the shape of a rubber-band stretched around the pegs.
- **Formal definition**: the convex hull of S is the smallest convex polygon that contains all the points of S.

Example



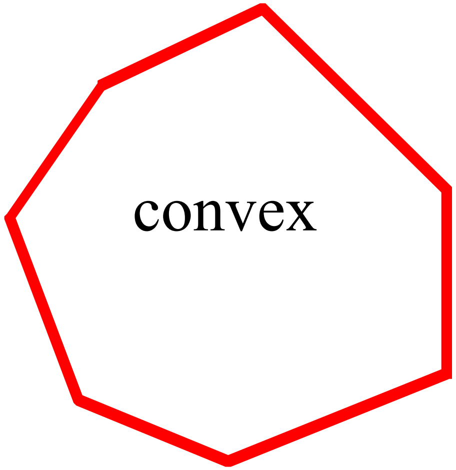
Contd…



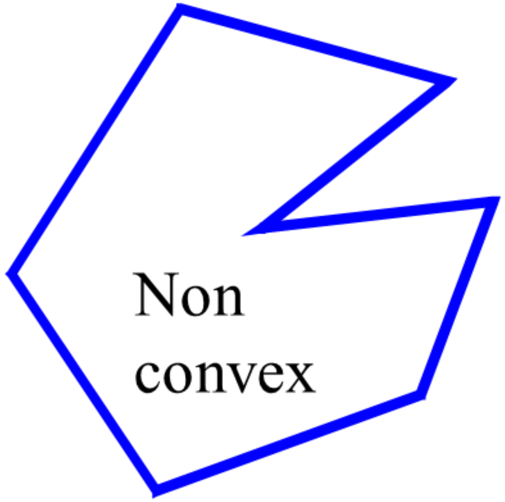
Contd…

- • A polygon P is said to be convex if:
- – P is non-intersecting; and
- – for any two points p and q on the boundary of P, segment pq lies entirely inside P

Convex



Not Convex



Algorithm

Convex hull: smallest convex set that includes given points

- Assume points are sorted by x-coordinate values
- Identify *extreme points* P_1 and P_2 (leftmost and rightmost)

Algorithm

- Compute *upper hull* recursively:
  - find point P_{max} that is farthest away from line P_1P_2
  - compute the upper hull of the points to the left of line P_1P_{max}
  - compute the upper hull of the points to the left of line $P_{max}P_2$
- Compute *lower hull* in a similar manner

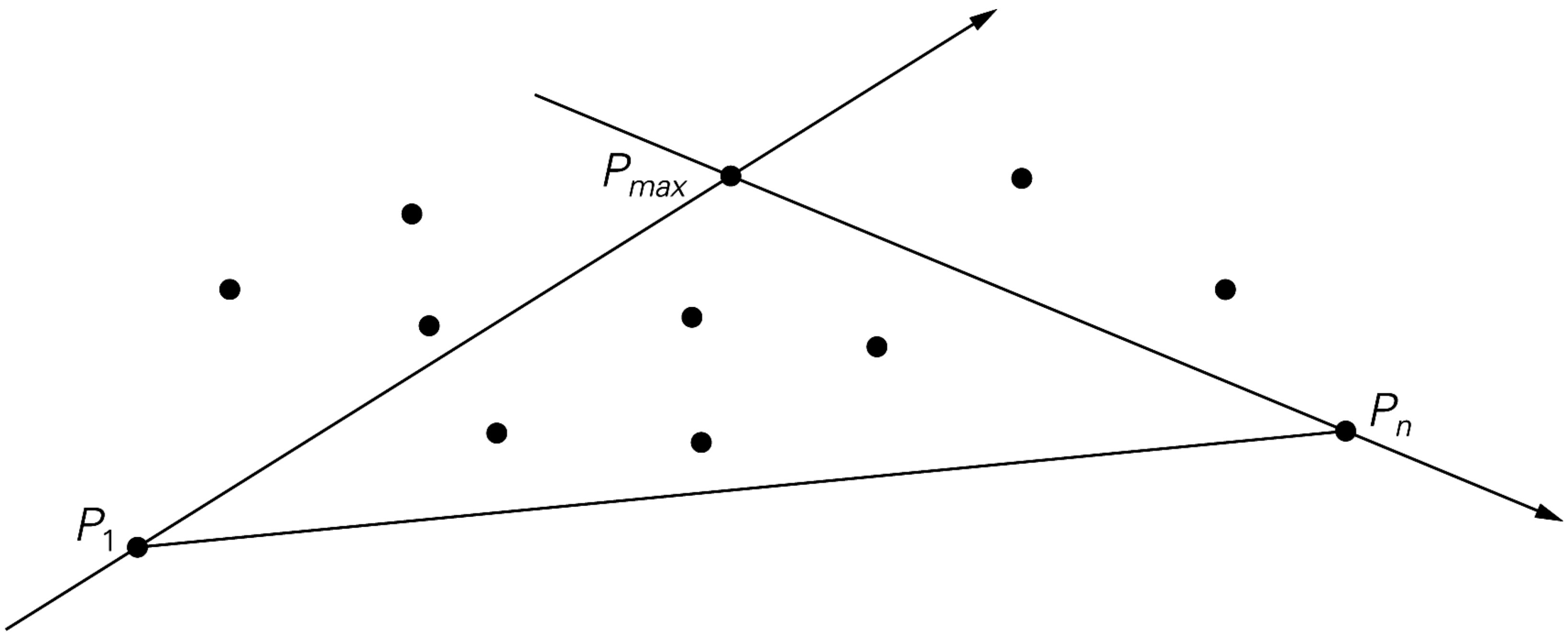
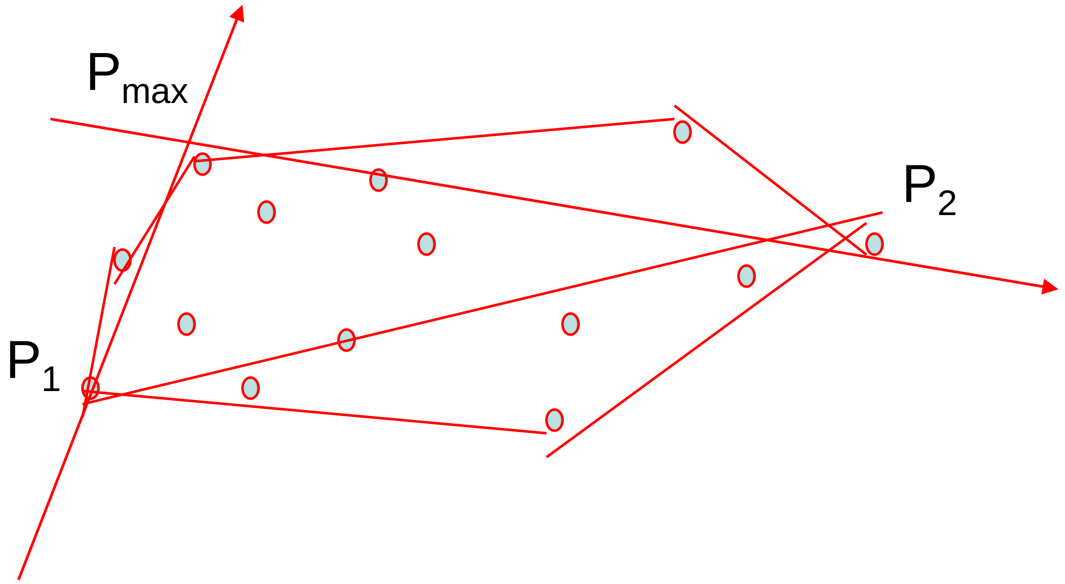


FIGURE 4.9 The idea of quickhull



P_{max}
P_2
P_1

Algorithm Analysis

- Finding point farthest away from line P_1P_2 can be done in linear time
- Time efficiency:
  - worst case: $\Theta(n^2)$ (as quicksort)
  - average case: $\Theta(n)$ (under reasonable assumptions about distribution of points given)
- If points are not initially sorted by x-coordinate value, this can be accomplished in $O(n \log n)$ time

Contd…

- Several O(n log n) algorithms for convex hull are known

Contd…

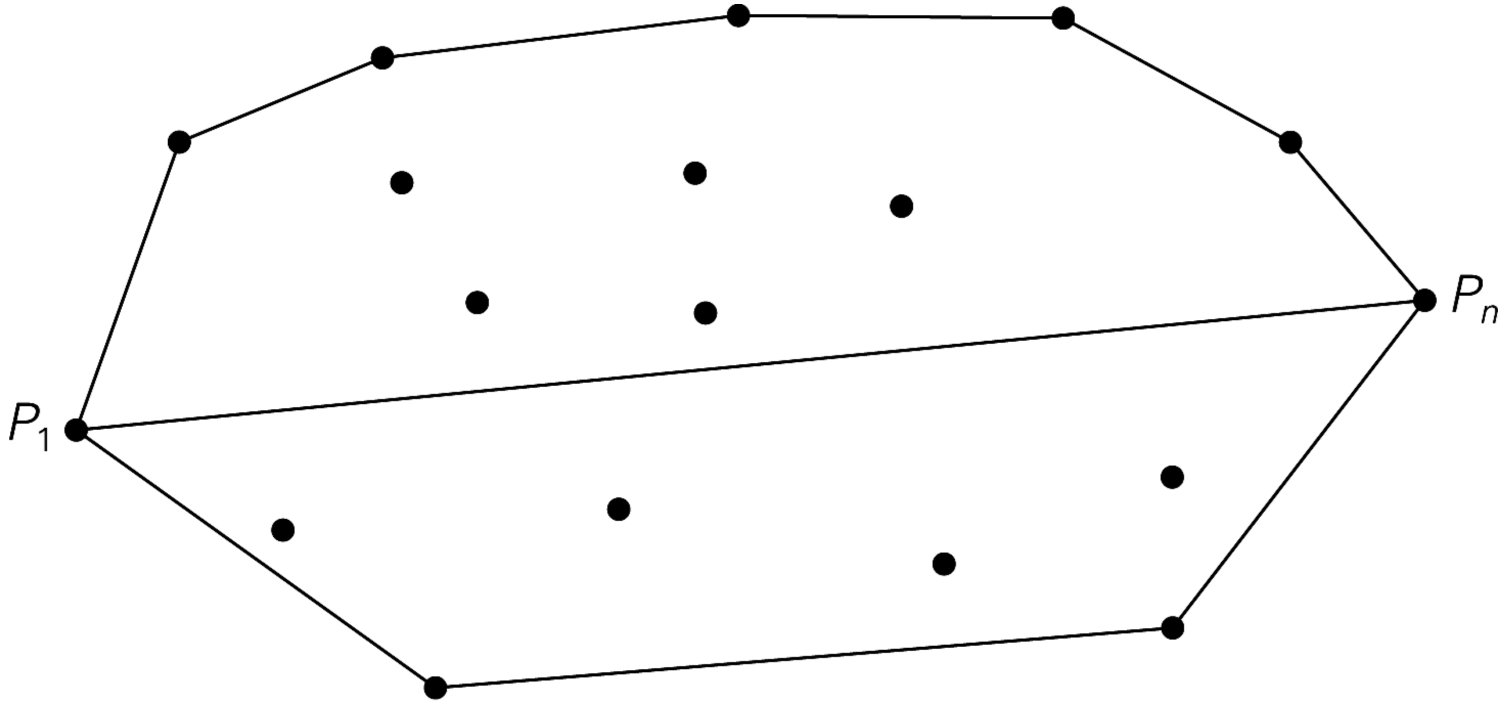
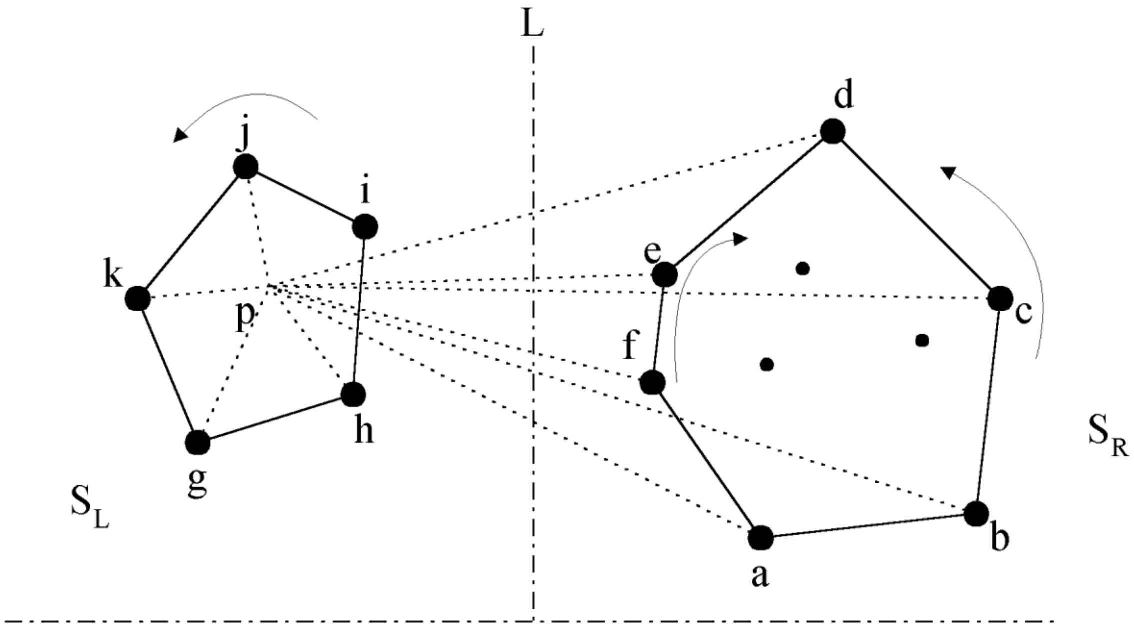


FIGURE 4.8 Upper and lower hulls of a set of points

The divide-and-conquer strategy



Merge Sequence

1. Select an interior point p.
2. There are 3 sequences of points which have increasing polar angles with respect to p.

   (1) g, h, i, j, k

   (2) a, b, c, d

   (3) f, e

Merge Sequence

3. Merge these 3 sequences into 1 sequence:

   g, h, a, b, f, c, e, d, i, j, k.
4. Apply Graham scan to examine the points one by one and eliminate the points which cause reflexive angles.

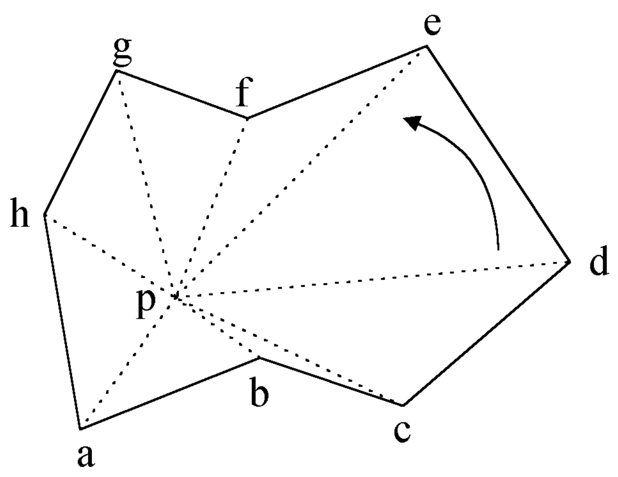
Divide-and-conquer for convex hull

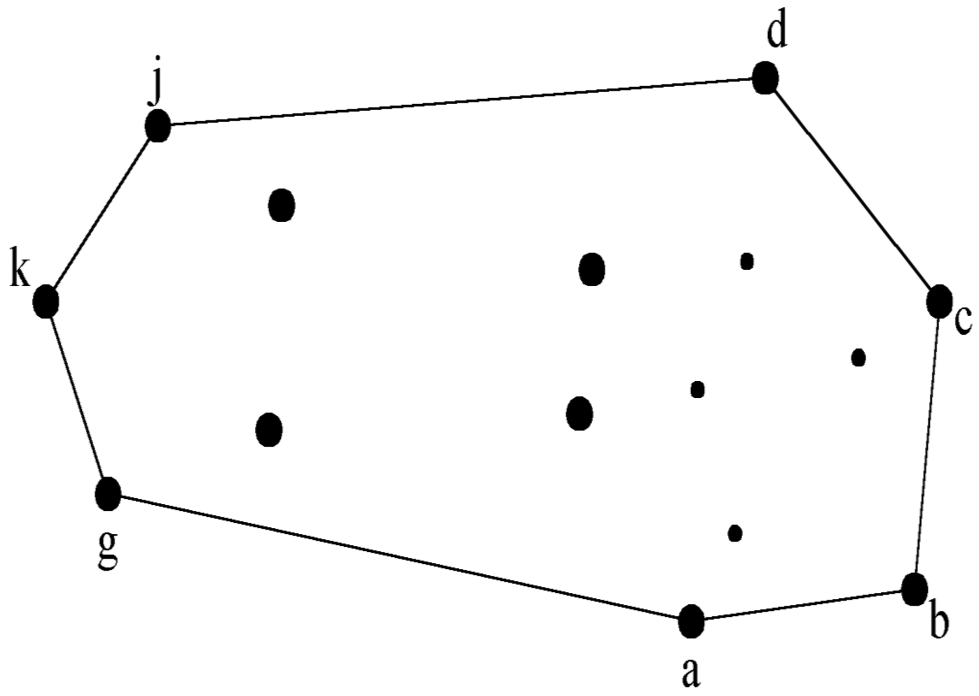
- Input : A set S of planar points
- Output : A convex hull for S

Step 1: If S contains no more than five points, use exhaustive searching to find the convex hull and return.

Step 2: Find a median line perpendicular to the X-axis which divides S into S_L and S_R, with equal sizes.

points b and f need to be deleted





Divide-and-conquer for convex hull

- Step 3: Recursively construct convex hulls for S_L and S_R, denoted as Hull(S_L) and Hull(S_R), respectively.
- Step 4: Apply the merging procedure to merge Hull(S_L) and Hull(S_R) together to form a convex hull.
- Time complexity:

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

Dynamic Programming

- Dynamic Programming is a general algorithm design technique for solving problems defined by recurrences with overlapping subproblems
- Invented by American mathematician Richard Bellman in the 1950s to solve optimization problems and later assimilated by CS

Dynamic Programming

- "Programming" here means "planning"
- Main idea:
  - set up a recurrence relating a solution to a larger instance to solutions of some smaller instances
- - solve smaller instances once
  - record solutions in a table
  - extract solution to the initial instance from that table

Algorithm

ALGORITHM *Binomial*(n, k)

//Computes $C(n, k)$ by the dynamic programming algorithm
//Input: A pair of nonnegative integers $n \geq k \geq 0$
//Output: The value of $C(n, k)$
for $i \leftarrow 0$ **to** n **do**
    for $j \leftarrow 0$ **to** $\min(i, k)$ **do**
        if $j = 0$ **or** $j = i$
            $C[i, j] \leftarrow 1$
        else $C[i, j] \leftarrow C[i-1, j-1] + C[i-1, j]$
return $C[n, k]$

	0	1	2	...	$k-1$	k
0	1					
1	1	1				
2	1	2	1			
⋮						
k	1					1
⋮						
$n-1$	1				$C(n-1, k-1)$	$C(n-1, k)$
n	1					$C(n, k)$

FIGURE 8.1 Table for computing the binomial coefficient $C(n, k)$ by the dynamic programming algorithm