Algorithm Design and Analysis

LECTURE 13
Divide and Conquer
• Closest Pair of Points
• Convex Hull
• Strassen Matrix Mult.

Adam Smith
Midterm Exam #1

- Willard Building Room 76
- Tuesday night, 8:15pm

- You may bring: one (1) double-sided, hand-written 8.5” x 11” sheet of notes on colored paper
  - Hint: use its preparation as a study aid
Review questions

• Find the solution to the recurrence using MT:

\[ T(n) = 8T(n/2) + cn. \]

(Answer: \( \log_b(a) = 3 > 1 \), solution is \( \Theta(n^3) \).)

• Draw the recursion tree for this recurrence.
  a. What is its height?

(Answer: \( h = \log n \).)

  b. What is the number of leaves in the tree?

(Answer: \( 8^h = 8^{\log n} = n^{\log 8} = n^3 \).)
Review questions: recursion tree

Solve $T(n) = 8T(n/2) + cn$:

Total $= cn(1 + 4 + 4^2 + 4^3 + \ldots + n^2)$

$= \Theta(n^3)$ geometric series
Appendix: geometric series

\[ 1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x} \quad \text{for } x \neq 1 \]

\[ 1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1 \]
Closest Pair of Points

Closest pair. Given $n$ points in the plane, find a pair with smallest Euclidean distance between them.

Fundamental geometric primitive.
- Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.
- Special case of nearest neighbor, Euclidean MST, Voronoi.

Brute force. Check all pairs of points $p$ and $q$ with $\Theta(n^2)$ comparisons.

1-D version (points on a line): $O(n \log n)$ easy via sorting

Assumption. No two points have same $x$ coordinate.

↑
to make presentation cleaner
Closest Pair of Points: First Attempt

Divide. Sub-divide region into 4 quadrants.
Closest Pair of Points: First Attempt

Divide. Sub-divide region into 4 quadrants.
Obstacle. Impossible to ensure \( \frac{n}{4} \) points in each piece.
Closest Pair of Points

Algorithm.

- **Divide**: draw vertical line $L$ so that roughly $\frac{1}{2}n$ points on each side.
Closest Pair of Points

Algorithm.
- **Divide**: draw vertical line $L$ so that roughly $\frac{1}{2}n$ points on each side.
- **Conquer**: find closest pair in each side recursively.
Closest Pair of Points

Algorithm.

- **Divide**: draw vertical line $L$ so that roughly $\frac{1}{2}n$ points on each side.
- **Conquer**: find closest pair in each side recursively.
- **Combine**: find closest pair with one point in each side. \( \text{seems like } \Theta(n^2) \)
- Return best of 3 solutions.
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance $< \delta$.

$\delta = \min(16, 21)$
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance < $\delta$.
- Observation: only need to consider points within $\delta$ of line $L$.

\[ \delta = \min(16, 21) \]
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance < $\delta$.

- Observation: only need to consider points within $\delta$ of line $L$.
- Sort points in $2\delta$-strip by their $y$ coordinate.

$\delta = \min(16, 21)$
Closest Pair of Points

Find closest pair with one point in each side, assuming that distance < $\delta$.

- Observation: only need to consider points within $\delta$ of line L.
- Sort points in $2\delta$-strip by their $y$ coordinate.
- **Theorem**: Only need to check distances of those within 11 positions in sorted list!

$\delta = \min(16, 21)$
**Def.** Let $s_i$ be the point in the $2\delta$-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 $\delta$.

**Proof.**
- 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

Closest-Pair(p₁, ..., pₙ) {
    Compute separation line \( L \) such that half the points are on one side and half on the other side.

    \( δ_1 = \text{Closest-Pair(left half)} \)
    \( δ_2 = \text{Closest-Pair(right half)} \)
    \( δ = \min(δ_1, δ_2) \)

    Delete all points further than \( δ \) from separation line \( L \)

    Sort remaining points by \( y \)-coordinate.

    Scan points in \( y \)-order and compare distance between each point and next 11 neighbors. If any of these distances is less than \( δ \), update \( δ \).

    return \( δ \).
}
Closest Pair of Points: Analysis

Running time.

T(n) ≤ 2T(n/2) + O(n log n) ⇒ T(n) = O(n log² n)

Q. Can we achieve O(n log n)?

A. Yes. Don't sort points in strip from scratch each time.
   - Sort entire point set by x-coordinate only once
   - Each recursive call takes as input a set of points sorted by x coordinates and returns the same points sorted by y coordinate (together with the closest pair)
   - Create new y-sorted list by merging two output from recursive calls

Total time (n) = O(n log(n)) + T(n)

T(n) ≤ 2T(n/2) + O(n) ⇒ T(n) = O(n log n)
Divide and Conquer in low-dimensional geometry

- Powerful technique for low-dimensional geometric problems
  - Intuition: points in different parts of the plane don’t interfere too much
- Example: convex hull in $O(n \log (n))$ time a la MergeSort
  1. Convex-Hull(left-half) $T(n/2)$
  2. Convex-Hull(right-half) $T(n/2)$
  3. Merge (see Cormen et al., Chap 33) $\Theta(n)$
Matrix multiplication

Input: \( A = [a_{ij}], \ B = [b_{ij}] \).

Output: \( C = [c_{ij}] = A \cdot B \).

\[
\begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1n} \\
c_{21} & c_{22} & \cdots & c_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
c_{n1} & c_{n2} & \cdots & c_{nn}
\end{bmatrix} =
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1n} \\
b_{21} & b_{22} & \cdots & b_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n1} & b_{n2} & \cdots & b_{nn}
\end{bmatrix}
\]

\[c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}\]
Standard algorithm

for $i \leftarrow 1$ to $n$
    do for $j \leftarrow 1$ to $n$
        do $c_{ij} \leftarrow 0$
            for $k \leftarrow 1$ to $n$
                do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Running time $= \Theta(n^3)$
Divide-and-conquer algorithm

**Idea:**

$n \times n$ matrix = $2 \times 2$ matrix of $(n/2) \times (n/2)$ submatrices:

\[
\begin{bmatrix}
  r & s \\
  t & u
\end{bmatrix} = \begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix} \cdot \begin{bmatrix}
  e & f \\
  g & h
\end{bmatrix}
\]

\[
C = A \cdot B
\]

\[
\begin{align*}
  r &= ae + bg \\
  s &= af + bh \\
  t &= ce + dh \\
  u &= cf + dg
\end{align*}
\]

8 muts of $(n/2) \times (n/2)$ submatrices
4 adds of $(n/2) \times (n/2)$ submatrices
Analysis of D&C algorithm

\[ T(n) = 8T(n/2) + \Theta(n^2) \]

\# submatrices \quad submatrix size \quad work adding submatrices

\[ n^{\log_{ba}} = n^{\log_{2} 8} = n^3 \implies \text{CASE 1} \implies T(n) = \Theta(n^3). \]

No better than the ordinary algorithm.

9/24/2008

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Strassen’s idea

- Multiply $2 \times 2$ matrices with only 7 recursive mults.

\[
\begin{align*}
    P_1 &= a \cdot (f - h) \\
    P_2 &= (a + b) \cdot h \\
    P_3 &= (c + d) \cdot e \\
    P_4 &= d \cdot (g - e) \\
    P_5 &= (a + d) \cdot (e + h) \\
    P_6 &= (b - d) \cdot (g + h) \\
    P_7 &= (a - c) \cdot (e + f) \\
\end{align*}
\]

\[
\begin{align*}
    r &= P_5 + P_4 - P_2 + P_6 \\
    s &= P_1 + P_2 \\
    t &= P_3 + P_4 \\
    u &= P_5 + P_1 - P_3 - P_7 \\
\end{align*}
\]

7 mults, 18 adds/subs.

Note: No reliance on commutativity of multiplication!
Strassen’s idea

• Multiply $2 \times 2$ matrices with only 7 recursive mults.

\[
P_1 = a \cdot (f - h) \\
P_2 = (a + b) \cdot h \\
P_3 = (c + d) \cdot e \\
P_4 = d \cdot (g - e) \\
P_5 = (a + d) \cdot (e + h) \\
P_6 = (b - d) \cdot (g + h) \\
P_7 = (a - c) \cdot (e + f )
\]

\[
r = P_5 + P_4 - P_2 + P_6 \\
= (a + d)(e + h) + d(g - e) - (a + b)h + (b - d)(g + h) \\
= ae + ah + de + dh + dg - de - ah - bh + bg + bh - dg - dh \\
= ae + bg
\]
1. **Divide:** Partition $A$ and $B$ into $(n/2) \times (n/2)$ submatrices. Form terms to be multiplied using $+$ and $-$.

2. **Conquer:** Perform 7 multiplications of $(n/2) \times (n/2)$ submatrices recursively.

3. **Combine:** Form product matrix $C$ using $+$ and $-$ on $(n/2) \times (n/2)$ submatrices.

$$T(n) = 7 \, T(n/2) + \Theta(n^2)$$
Analysis of Strassen

\[ T(n) = 7 T(n/2) + \Theta(n^2) \]

\[ n^{\log_{27}} \approx n^{2.81} \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^{\log 7}). \]

• Number 2.81 may not seem much smaller than 3.
• But the difference is in the exponent.
• The impact on running time is significant.
• Strassen’s algorithm beats the ordinary algorithm on today’s machines for \( n \geq 32 \) or so.

**Best to date** (of theoretical interest only): \( \Theta(n^{2.376\ldots}) \).