# CMPSCI 611: Advanced Algorithms

## Lecture 2: More Divide and Conquer

Andrew McGregor

Last Compiled: September 10, 2017

# Outline

# Outline

# Divide and Conquer Methodology

- General Goal: Solve problem $P$ on an instance $I$ of "size" $n$.
- Divide & Conquer:
  1. Transform $I$ into smaller instances $I_1, \ldots, I_a$ each of "size" $n/b$
  2. Solve problem $P$ on each of $I_1, \ldots, I_a$ by recursion
  3. Combine the solutions to get a solution of $I$
- Example (Merge Sort): To sort $n$ numbers, divide into 2 sets of size $\frac{n}{2}$, sort each set, and merge.

# Analyzing Divide and Conquer Algorithms

Let $T(n)$ be running time of algorithm on instance of size $n$. Then

$$T(1) = \Theta(1), \qquad T(n) \leq aT(n/b) + O(n^{\alpha})$$

where $O(n^{\alpha})$ is time to create the subproblems and combine solutions.

## Theorem (Master Theorem)

*If $a, b, \alpha$ are constants,*

$$T(n) = \begin{cases} O(n^{\alpha}) & \text{if } \alpha > \log_b a \\ O(n^{\log_b a}) & \text{if } \alpha < \log_b a \\ O(n^{\alpha} \log n) & \text{if } \alpha = \log_b a \end{cases}$$

- Example (Merge Sort): $a = b = 2$ and $\alpha = 1$. Therefore the running time is $O(n \log n)$.

# Outline

# First Attempt at Matrix Multiplication

Given two $n \times n$ matrices $A$ and $B$, multiply them together to get $C$:

$$c_{ij} = \sum_{k \in [n]} a_{ik} b_{kj}$$

Naive algorithm works in $O(n^3)$ time. Try Divide and Conquer...

- Divide $A$ and $B$ into four $n/2 \times n/2$ sub-matrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

- And note

$$C = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_2 & P_3 + P_4 \\ P_5 + P_6 & P_7 + P_8 \end{pmatrix}$$

  where $P_1 = A_{11}B_{11}$ and $P_2 = A_{12}B_{21}$ etc.

- Bad News: $T(n) = 8T(n/2) + \Theta(n^2)$ gives $T(n) = \Theta(n^3)$

# Strassen's Algorithm: General Technique + Creativity

- Along comes Volker Strassen in 1969...

# Strassen's Algorithm

Break the problem into 7 sub-problems:

$$\begin{aligned}
P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
P_2 &= (A_{21} + A_{22})(B_{11}) \\
P_3 &= (A_{11})(B_{12} - B_{22}) \\
P_4 &= (A_{22})(-B_{11} + B_{21}) \\
P_5 &= (A_{11} + A_{12})(B_{22}) \\
P_6 &= (-A_{11} + A_{21})(B_{11} + B_{12}) \\
P_7 &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{aligned}$$

Then

$$AB = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 - P_2 + P_3 + P_6 \end{pmatrix}$$

Good: $T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$ gives $T(n) = \Theta(n^{2.81})$.

Improvements: $O(n^{2.376})$ by Coppersmith, Winograd 1990, $O(n^{2.3736})$ by Stothers 2010, $O(n^{2.3729})$ by Williams 2011, $O(n^{2.3728})$ by Le Gall 2014.

# Outline

# Finding Minimum Distance between Points on a Plane

Problem: Given $n$ distinct points $p_1, \ldots, p_n \in \mathbb{R}^2$, find

$$\text{minimum distance between any two points} = \min_{i \neq j} d(p_i, p_j)$$

How long does naive algorithm take? $O(n^2)$

We'll do it in $O(n \log n)$ steps.

For simplicity, assume no two points have the same $x$ or $y$ coordinate.

# Minimum Distance Algorithm

1. Divide points $P$ with a vertical line into $P_L$ and $P_R$ where

$$|P_L| = |P_R| = n/2$$

2. Recursively find minimum distance within $P_L$ and $P_R$:

$$\delta_L = \min_{p,q \in P_L : p \neq q} d(p,q)$$
$$\delta_R = \min_{p,q \in P_R : p \neq q} d(p,q)$$

3. Compute $\delta_M = \min_{p \in P_L, q \in P_R} d(p,q)$ and return

$$\min(\delta_L, \delta_R, \delta_M)$$

Note: If Step 3 takes $O(n^2)$ time, we get

$$T(n) \leq 2T(n/2) + O(n^2) \Longrightarrow T(n) = O(n^2)$$

If we can do Step 3 in $\Theta(n)$ time, we get $T(n) = O(n \log n)$.

# Making Step 3 Efficient

- Need to find $\min(\delta_L, \delta_R, \delta_M)$ where $\delta_M = \min_{p \in P_L, q \in P_R} d(p, q)$
- Suppose that the dividing line is $x = m$ and $\delta = \min(\delta_L, \delta_R)$
- Once we know $\delta$, only need $O(n)$ comparisons to find $\min(\delta, \delta_M)$
  1. Only compare $(p_1, p_2) \in P_L, (q_1, q_2) \in P_R$ if

$$m - \delta < p_1 \leq q_1 < m + \delta$$

     and

$$|p_2 - q_2| < \delta$$

  2. Each point $p \in P_L$ only gets compared with $O(1)$ points in $P_R$
- Need to identify the relevant comparisons in $O(n)$ time
  1. Make two copies of points sorted by each coordinate
  2. Ensure both lists are passed to each recursion sorted
  3. Given sorted lists, it's easy to find the relevant points

# Details: How many points does $p$ get compared with?

- All the points that $p$ gets compared with lie in a $2\delta \times \delta$ rectangle.
- Since each is at least $\delta$ away from the others, we can draw circles of radius $r = \delta/2$ around each and these circles do not overlap.
- The area of each circles that overlaps the box is at least $\pi r^2/4$.
- Since the total area of the rectangle is $2\delta^2$, the total number of points must be at most $2\delta^2/(\pi r^2/4) = 32/\pi = 10.19\ldots$.
- Better constants are possible but the exact constant isn't important.