CS4102 Algorithms

Fall 2019
**Warm up**

Given 5 points on the unit equilateral triangle, show there’s always a pair of distance ≤ $\frac{1}{2}$ apart
If points $p_1, p_2$ in same quadrant, then $\delta(p_1, p_2) \leq \frac{1}{2}$

Given 5 points, two must share the same quadrant

Pigeonhole Principle!
Today’s Keywords

- Divide and Conquer
- Closest Pair of Points
• Chapter 4
Homeworks

• Hw1 due **Saturday, September 14** at 11pm
  – Written (use Latex!) – Submit BOTH pdf and zip!
  – Asymptotic notation
  – Recurrences
  – Divide and Conquer

• Hw2 released today, due Thursday Sept 19 at 11pm
  – Programming assignment (Python or Java)
  – Divide and conquer
Recurrence Solving Techniques

- Tree
- Guess/Check
- “Cookbook”
- Substitution
Master Theorem

\[ T(n) = aT\left(\frac{n}{b}\right) + f(n) \]

Case 1: if \( f(n) = O(n^{\log_b a - \varepsilon}) \) for some constant \( \varepsilon > 0 \), then \( T(n) = \Theta(n^{\log_b a}) \)

Case 2: if \( f(n) = \Theta(n^{\log_b a}) \), then \( T(n) = \Theta(n^{\log_b a} \log n) \)

Case 3: if \( f(n) = \Omega(n^{\log_b a + \varepsilon}) \) for some constant \( \varepsilon > 0 \), and if \( af\left(\frac{n}{b}\right) \leq cf(n) \) for some constant \( c < 1 \) and all sufficiently large \( n \), then \( T(n) = \Theta(f(n)) \)
3 Cases

\[ T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + a^3 f\left(\frac{n}{b^3}\right) + \cdots + a^L f\left(\frac{n}{b^L}\right) \]

Case 1:
Most work happens at the leaves

Case 2:
Work happens consistently throughout

Case 3:
Most work happens at top of tree
Historical Aside: Master Theorem

Jon Bentley  
Dorothea Haken  
James Saxe

No Picture Found
Substitution Method

\[ T(n) = 2T(\sqrt{n}) + \log_2 n = 2T(n^{1/2}) + \log_2 n \]

Let \( n = 2^m \), i.e. \( m = \log_2 n \)

\[ T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m \] Rewrite in terms of exponent!

Let \( S(m) = 2S\left(\frac{m}{2}\right) + m \) Case 2!

Let \( S(m) = \Theta(m \log m) \) Substitute Back

Let \( T(n) = \Theta(\log n \log \log n) \)

I don’t like the \( \frac{1}{2} \) in the exponent

Now the variable is in the exponent on both sides!

S will operate exactly as T, just redefined in terms of the exponent

\[ S(m) = T(2^m) \]
My Yard
There has to be an easier way!
Constraints: Trees and Plants

Need to find:
Closest Pair of Trees - how wide can the robot be?
Given:
A list of points

Return:
Pair of points with smallest distance apart
Given:
A list of points

Return:
Pair of points with smallest distance apart

Algorithm: $O(n^2)$
Test every pair of points, return the closest.

We can do better! $\Theta(n \log n)$
Closest Pair of Points: D&C

Divide: How?
At median x coordinate

Conquer:
Closest Pair of Points: D&C

**Divide:**
At median x coordinate

**Conquer:**
Recursively find closest pairs from Left and Right

**Combine:**
Closest Pair of Points: D&C

Divide:
At median x coordinate

Conquer:
Recursively find closest pairs from Left and Right

Combine:
Return min of Left and Right pairs  Problem?
Combine:

2 Cases:

1. Closest Pair is completely in Left or Right

2. Closest Pair Spans our “Cut”

Need to test points across the cut
Combine:

2. Closest Pair Spanned our “Cut”

Need to test points across the cut

Compare all points within \( \delta = \min\{\delta_L, \delta_R\} \) of the cut.

How many are there?
Combine:

2. Closest Pair Spanned our “Cut”

Need to test points across the cut

Compare all points within $\delta = \min\{\delta_L, \delta_R\}$ of the cut.

How many are there?

$$T(n) = 2T\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right)^2 = \Theta(n^2)$$
Spanning the Cut

Combine:

2. Closest Pair Spanned our “Cut”

Need to test points across the cut

We don’t need to test all pairs!

Only need to test points within $\delta$ of one another
Combine:

2. Closest Pair Spanned our “Cut”

Need to test points across the cut

Divide the “runway” into square cubbies of size $\frac{\delta}{2}$

Each cubby will have at most 1 point!
Reducing Search Space

Combine:

2. Closest Pair Spanned our “Cut”

Need to test points across the cut

Divide the “runway” into square cubbies of size \( \frac{\delta}{2} \)

- How many cubbies could contain a point \(< \delta\) away?

Each point compared to \(\leq 15\) other points
**Initialization:** Sort points by $x$-coordinate

**Divide:** Partition points into two lists of points based on $x$-coordinate (split at the median $x$)

**Conquer:** Recursively compute the closest pair of points in each list

   Base case?

**Combine:**
- Construct list of points in the runway ($x$-coordinate within distance $\delta$ of median)
- Sort runway points by $y$-coordinate
- Compare each point in runway to 15 points above it and save the closest pair
- Output closest pair among left, right, and runway points
**Initialization:** Sort points by $x$-coordinate

**Divide:** Partition points into two lists of points based on $x$-coordinate (split at the median $x$)

**Conquer:** Recursively compute the closest pair of points in each list

**Base case?**

**Combine:**
- Construct list of points in the runway ($x$-coordinate within distance $\delta$ of median)
- Sort runway points by $y$-coordinate
- Compare each point in runway to 15 points above it and save the closest pair
- Output closest pair among left, right, and runway points

But sorting is an $O(n \log n)$ algorithm – combine step is still too expensive! We need $O(n)$
Closest Pair of Points: Divide and Conquer

**Initialization:** Sort points by $x$-coordinate

**Divide:** Partition points into two lists of points based on $x$-coordinate (split at the median $x$)

**Conquer:** Recursively compute the closest pair of points in each list

**Base case?**

**Combine:**
- Construct list of points in the runway ($x$-coordinate within distance $\delta$ of median)
- Sort runway points by $y$-coordinate
- Compare each point in runway to 15 points above it and save the closest pair
- Output closest pair among left, right, and runway points

**Solution:** Maintain additional information in the recursion
- Minimum distance among pairs of points in the list
- List of points sorted according to $y$-coordinate

Sorting runway points by $y$-coordinate now becomes a **merge**
Listing Points in the Runway

Output on Left:
- Closest Pair: (1, 5), \( \delta_{1,5} \)
- Sorted Points: [3, 7, 5, 1]

Output on Right:
- Closest Pair: (4, 6), \( \delta_{4,6} \)
- Sorted Points: [8, 6, 4, 2]

Merged Points: [8, 3, 7, 6, 4, 5, 1, 2]

Runway Points: [8, 7, 6, 5, 2]

Both of these lists can be computed by a single pass over the lists
**Initialization:** Sort points by $x$-coordinate

**Divide:** Partition points into two lists of points based on $x$-coordinate (split at the median $x$)

**Conquer:** Recursively compute the closest pair of points in each list
  Base case?

**Combine:**
- Construct list of points in the runway ($x$-coordinate within distance $\delta$ of median)
- **Sort runway points by $y$-coordinate**
- Compare each point in runway to 15 points above it and save the closest pair
- Output closest pair among left, right, and runway points

**Initialization:** Sort points by $x$-coordinate

**Divide:** Partition points into two lists of points based on $x$-coordinate (split at the median $x$)

**Conquer:** Recursively compute the closest pair of points in each list

**Combine:**
- Merge sorted list of points by $y$-coordinate and construct list of points in the runway (sorted by $y$-coordinate)
- Compare each point in runway to 15 points above it and save the closest pair
- Output closest pair among left, right, and runway points
**Closest Pair of Points: Divide and Conquer**

**What is the running time?**

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

**Initialization:** Sort points by \( x \)-coordinate

**Divide:** Partition points into two lists of points based on \( x \)-coordinate (split at the median \( x \))

**Conquer:** Recursively compute the closest pair of points in each list

**Combine:**
- Merge sorted list of points by \( y \)-coordinate and construct list of points in the runway (sorted by \( y \)-coordinate)
- Compare each point in runway to 15 points above it and save the closest pair
- Output closest pair among left, right, and runway points

**Case 2 of Master’s Theorem**

\[ T(n) = \Theta(n \log n) \]
Matrix Multiplication

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\times
\begin{bmatrix}
2 & 4 & 6 \\
8 & 10 & 12 \\
14 & 16 & 18 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
2 + 16 + 42 & 4 + 20 + 48 & 6 + 24 + 54 \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
60 & 72 & 84 \\
132 & 162 & 192 \\
204 & 252 & 300 \\
\end{bmatrix}
\]

Run time? \(O(n^3)\)
Matrix Multiplication D&C

Multiply $n \times n$ matrices ($A$ and $B$)

Divide:

$$A = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{51} & a_{61} & a_{71} & a_{81} \\ a_{91} & a_{101} & a_{111} & a_{121} \\ a_{131} & a_{141} & a_{151} & a_{161} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} \\ b_{51} & b_{61} & b_{71} & b_{81} \\ b_{91} & b_{101} & b_{111} & b_{121} \\ b_{131} & b_{141} & b_{151} & b_{161} \end{bmatrix}$$
Matrix Multiplication D&C

Multiply $n \times n$ matrices ($A$ and $B$)

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

$$B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

Combine:

$$AB = \begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}$$

Run time? 

$$T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2$$

Cost of additions
Matrix Multiplication D&C

\[ T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2 \]

\[ T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) \]

\[ a = 8, \quad b = 2, \quad f(n) = n^2 \]

Case 1!

\[ n^{\log_b a} = n^{\log_2 8} = n^3 \]

\[ T(n) = \Theta(n^3) \]

We can do better...
Matrix Multiplication D&C

Multiply \( n \times n \) matrices \((A\) and \(B)\)

\[
A = \begin{bmatrix}
A_{1,1} & A_{1,2} \\
A_{2,1} & A_{2,2}
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
B_{1,1} & B_{1,2} \\
B_{2,1} & B_{2,2}
\end{bmatrix}
\]

\[
AB = \begin{bmatrix}
A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\
A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2}
\end{bmatrix}
\]

Idea: Use a Karatsuba-like technique on this
Strassen’s Algorithm

Multiply $n \times n$ matrices ($A$ and $B$)

$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$

$B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$

Calculate:

$Q_1 = (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2})$

$Q_2 = (A_{2,1} + A_{2,2})B_{1,1}$

$Q_3 = A_{1,1}(B_{1,2} - B_{2,2})$

$Q_4 = A_{2,2}(B_{2,1} - B_{1,1})$

$Q_5 = (A_{1,1} + A_{1,2})B_{2,2}$

$Q_6 = (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2})$

$Q_7 = (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2})$

Find $AB$:

$\begin{bmatrix} Q_1 + Q_4 - Q_5 + Q_7 & Q_3 + Q_5 \\ Q_2 + Q_4 & Q_1 - Q_2 + Q_3 + Q_6 \end{bmatrix}$

$\begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}$

Number Mults.: 7  
Number Adds.: 18

$T(n) = 7T\left(\frac{n}{2}\right) + \frac{9}{2}n^2$
Strassen’s Algorithm

\[ T(n) = 7T \left( \frac{n}{2} \right) + \frac{9}{2} n^2 \]

\[ a = 7, \ b = 2, \ f(n) = \frac{9}{2} n^2 \]

\[ n^{\log_b a} = n^{\log_2 7} \approx n^{2.807} \]

Case 1!

\[ T(n) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.807}) \]
Strassen’s Algorithm
Best possible is unknown

May not even exist!