CSE 421

Divide and Conquer: Finding Root Closest Pair of Points

Shayan Oveis Gharan
Master Theorem

Suppose $T(n) = a \, T\left(\frac{n}{b}\right) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Works even if it is $\left\lfloor \frac{n}{b} \right\rfloor$ instead of $\frac{n}{b}$.

We also need $a \geq 1, b > 1, k \geq 0$ and $T(n) = O(1)$ for $n \leq b$. 
Master Theorem

Suppose $T(n) = a T \left( \frac{n}{b} \right) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Example: For merge sort algorithm we have

$$T(n) = 2T \left( \frac{n}{2} \right) + O(n).$$

So, $k = 1, a = b^k$ and $T(n) = \Theta(n \log n)$
Finding the Closest Pair of Points
A Divide and Conquer Alg

**Divide:** draw vertical line $L$ with $\approx n/2$ points on each side.

**Conquer:** find closest pair on each side, recursively.

**Combine** to find closest pair overall

Return best solutions

seems like $\Theta(n^2)$?
Suppose $\delta$ is the minimum distance of all pairs in left/right of L.

$$\delta = \min(12, 21) = 12.$$ 

**Key Observation**: suffices to consider points within $\delta$ of line L. Almost the one-D problem again: Sort points in $2\delta$-strip by their y coordinate.

Only check pts within 11 in sorted list!
Almost 1D Problem

Partition each side of $L$ into $\frac{\delta}{2} \times \frac{\delta}{2}$ squares

Claim: No two points lie in the same $\frac{\delta}{2} \times \frac{\delta}{2}$ box.

Pf: Such points would be within

$$\sqrt{\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2} = \delta \sqrt{\frac{1}{2}} \approx 0.7\delta < \delta$$

Let $s_i$ have the $i^{th}$ smallest $y$-coordinate among points in the $2\delta$-width-strip.

Claim: If $|i - j| > 11$, then the distance between $s_i$ and $s_j$ is $> \delta$.

Pf: only 11 boxes within $\delta$ of $y(s_i)$. 
Recap: Finding Closest Pair

Point 42 has distance at least \( 2\delta \) from point 30.

At most 11 points ahead of 30 have distance \(<\delta\) from it.

So, enough to check distance
Distance of 30 to 19…41.
Closest Pair (2Dim Algorithm)

```cpp
Closest-Pair(p_1, ..., p_n) {
    if(n <= ??) return ??

    Compute separation line L such that half the points are on one side and half on the other side.

    \( \delta_1 = \text{Closest-Pair(left half)} \)
    \( \delta_2 = \text{Closest-Pair(right half)} \)
    \( \delta = \min(\delta_1, \delta_2) \)

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

    Sort remaining points \( p[1]...p[m] \) by y-coordinate.

    for \( i = 1..m \)
        for \( k = 1..11 \)
            if \( i+k <= m \)
                \( \delta = \min(\delta, \text{distance}(p[i], p[i+k])) \);

    return \( \delta \).
}
```
Closest Pair Analysis I

Let $D(n)$ be the number of pairwise distance calculations in the Closest-Pair Algorithm when run on $n \geq 1$ points:

$$D(n) \leq \begin{cases} 
1 & \text{if } n = 1 \\
2D \left( \frac{n}{2} \right) + 11 n & \text{o.w.} \Rightarrow D(n) = O(n \log n)
\end{cases}$$

BUT, that’s only the number of distance calculations

What if we counted running time?

$$T(n) \leq \begin{cases} 
1 & \text{if } n = 1 \\
2T \left( \frac{n}{2} \right) + O(n \log n) & \text{o.w.} \Rightarrow D(n) = O(n \log^2 n)
\end{cases}$$
Can we do better? (Analysis II)

Yes!!

Don’t sort by y-coordinates each time.
Sort by x at top level only.

This is enough to divide into two equal subproblems in O(n)

Each recursive call returns δ and list of all points sorted by y
Sort points by y-coordinate by merging two pre-sorted lists.

\[ T(n) \leq \begin{cases} 
1 & \text{if } n = 1 \\
2T \left( \frac{n}{2} \right) + O(n) & \text{o.w.} 
\end{cases} \Rightarrow D(n) = O(n \log n) \]
Master Theorem

Suppose $T(n) = a \frac{n}{b} T \left( \frac{n}{b} \right) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Works even if it is $\left\lfloor \frac{n}{b} \right\rfloor$ instead of $\frac{n}{b}$.

We also need $a \geq 1, b > 1, k \geq 0$ and $T(n) = O(1)$ for $n \leq b$. 
Master Theorem

Suppose \( T(n) = a \ T \left( \frac{n}{b} \right) + cn^k \) for all \( n > b \). Then,

- If \( a > b^k \) then \( T(n) = \Theta(n \log_b a) \)
- If \( a < b^k \) then \( T(n) = \Theta(n^k) \)
- If \( a = b^k \) then \( T(n) = \Theta(n^k \log n) \)

Example: For mergesort algorithm we have

\[
T(n) = 2T \left( \frac{n}{2} \right) + O(n).
\]

So, \( k = 1, a = b^k \) and \( T(n) = \Theta(n \log n) \)
Integer Multiplication
**Integer Arithmetic**

**Add:** Given two n-bit integers $a$ and $b$, compute $a + b$.

$O(n)$ bit operations.

**Multiply:** Given two n-bit integers $a$ and $b$, compute $a \times b$.

The “grade school” method:

$O(n^2)$ bit operations.
How to use Divide and Conquer?

Suppose we want to multiply two 2-digit integers (32,45).
We can do this by multiplying four 1-digit integers
Then, use add/shift to obtain the result:

\[ x = 10x_1 + x_0 \]
\[ y = 10y_1 + y_0 \]
\[ xy = (10x_1 + x_0)(10y_1 + y_0) = 100 x_1y_1 + 10(x_1y_0 + x_0y_1) + x_0y_0 \]

Same idea works when multiplying n-digit integers:
• Divide into 4 n/2-digit integers.
• Recursively multiply
• Then merge solutions
A Divide and Conquer for Integer Multiplaition

Let $x, y$ be two $n$-bit integers

Write $x = 2^{n/2}x_1 + x_0$ and $y = 2^{n/2}y_1 + y_0$

where $x_0, x_1, y_0, y_1$ are all $n/2$-bit integers.

\[
x = 2^{n/2} \cdot x_1 + x_0
\]
\[
y = 2^{n/2} \cdot y_1 + y_0
\]
\[
xy = (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0)
\]
\[
= 2^n \cdot x_1y_1 + 2^{n/2} \cdot (x_1y_0 + x_0y_1) + x_0y_0
\]

Therefore,

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

So,

\[
T(n) = \Theta(n^2).
\]
Key Trick: 4 multiplies at the price of 3

\[ x = 2^{n/2} \cdot x_1 + x_0 \]
\[ y = 2^{n/2} \cdot y_1 + y_0 \]
\[ xy = (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \]
\[ = 2^n \cdot x_1 y_1 + 2^{n/2} (x_1 y_0 + x_0 y_1) + x_0 y_0 \]

\[ \alpha = x_1 + x_0 \]
\[ \beta = y_1 + y_0 \]
\[ \alpha \beta = (x_1 + x_0)(y_1 + y_0) \]
\[ = x_1 y_1 + (x_1 y_0 + x_0 y_1) + x_0 y_0 \]
\[ (x_1 y_0 + x_0 y_1) = \alpha \beta - x_1 y_1 - x_0 y_0 \]
Key Trick: 4 multiplies at the price of 3

Theorem [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in $O(n^{1.585\ldots})$ bit operations.

To multiply two n-bit integers:
- Add two $n/2$ bit integers.
- Multiply three $n/2$-bit integers.
- Add, subtract, and shift $n/2$-bit integers to obtain result.

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O\left(n^{\log_2 3}\right) = O(n^{1.585\ldots})$$
Integer Multiplication (Summary)

• Naïve: \( \Theta(n^2) \)

• Karatsuba: \( \Theta(n^{1.585\ldots}) \)

• Amusing exercise: generalize Karatsuba to do 5 size \( n/3 \) subproblems
  This gives \( \Theta(n^{1.46\ldots}) \) time algorithm

• Best known algorithm runs in \( \Theta(n \log n) \) using fast Fourier transform
  but mostly unused in practice (unless you need really big numbers - a billion digits of \( \pi \), say)

• Best lower bound \( O(n) \): A fundamental open problem
Proving Master Theorem

Problem size

\[ T(n) = aT(n/b) + cn^k \]

# probs

<table>
<thead>
<tr>
<th>n/b^d</th>
<th>d = \log_b n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a^d</td>
</tr>
<tr>
<td>n/b</td>
<td>a</td>
</tr>
<tr>
<td>n/b^2</td>
<td>a^2</td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

cost

<table>
<thead>
<tr>
<th>n/b^d</th>
<th>d = \log_b n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c \cdot n^k</td>
</tr>
<tr>
<td>n/b</td>
<td>c \cdot a \cdot n^k/b^k</td>
</tr>
<tr>
<td>n/b^2</td>
<td>c \cdot a^2 \cdot n^k/b^{2k}</td>
</tr>
<tr>
<td>b</td>
<td>c \cdot n^k (a/b^k)^2</td>
</tr>
<tr>
<td>1</td>
<td>c \cdot n^k (a/b^k)^d</td>
</tr>
</tbody>
</table>

\[ T(n) = cn^k \sum_{i=0}^{d=\log_b n} \left( \frac{a}{b^k} \right)^i \]
A Useful Identity

**Theorem:** \[1 + x + x^2 + \cdots + x^d = \frac{x^{d+1} - 1}{x - 1}\]

**Pf:** Let \( S = 1 + x + x^2 + \cdots + x^d \)

Then, \( xS = x + x^2 + \cdots + x^{d+1} \)

So, \( xS - S = x^{d+1} - 1 \)

i.e., \( S(x - 1) = x^{d+1} - 1 \)

Therefore, \( S = \frac{x^{d+1} - 1}{x - 1} \)
Solve: \( T(n) = aT\left(\frac{n}{b}\right) + cn^k, \ a > b^k \)

\[
T(n) = cn^k \sum_{i=0}^{\log_b n} \left(\frac{a}{b^k}\right)^i
\]

\[
= cn^k \frac{\left(\frac{a}{b^k}\right)^{\log_b n+1} - 1}{\left(\frac{a}{b^k}\right) - 1}
\]

\[
\leq c \left(\frac{n^k}{b^k \log_b n}\right) \frac{\left(\frac{a}{b^k}\right)}{\left(\frac{a}{b^k}\right) - 1} a^{\log_b n}
\]

\[
\leq 2c a^{\log_b n} = O(n^{\log_b a})
\]

\[
\frac{x^{d+1} - 1}{x^d - 1} \text{ for } x = \frac{a}{b^k}
\]

\[
d = \log_b n \text{ using } x \neq 1
\]
Solve: $T(n) = aT \left( \frac{n}{b} \right) + cn^k$, $a = b^k$

\[ T(n) = cn^k \sum_{i=0}^{\log_b n} \left( \frac{a}{b^k} \right)^i \]
\[ = cn^k \log_b n \]
Master Theorem

Suppose $T(n) = a \ T \left( \frac{n}{b} \right) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Works even if it is $\left\lceil \frac{n}{b} \right\rceil$ instead of $\frac{n}{b}$.

We also need $a \geq 1, b > 1, k \geq 0$ and $T(n) = O(1)$ for $n \leq b$. 