CSE 421

Closest Pair of Points, Master Theorem, Integer Multiplication

Shayan Oveis Gharan
Finding the Closest Pair of Points
Closest Pair of Points (non geometric)

Given n points and arbitrary distances between them, find the closest pair. (E.g., think of distance as airfare – definitely not Euclidean distance!)

Must look at all $n \choose 2$ pairwise distances, else any one you didn’t check might be the shortest. i.e., you have to read the whole input
Closest Pair of Points (1-dimension)

Given n points on the real line, find the closest pair, e.g., given 11, 2, 4, 19, 4.8, 7, 8.2, 16, 11.5, 13, 1 find the closest pair

Fact: Closest pair is adjacent in ordered list
So, first sort, then scan adjacent pairs.
Time $O(n \log n)$ to sort, if needed, Plus $O(n)$ to scan adjacent pairs

Key point: do not need to calc distances between all pairs: exploit geometry + ordering
Closest Pair of Points (2-dimensions)

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 $\Theta(n^2)$ time.

Assumption: No two points have same x coordinate.
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 $\Theta(n^2)$?
Key Observation

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 $y(s_i)$. 
Closest Pair (2Dim Algorithm)

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

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

BUT, that’s only the number of distance calculations

What if we counted running time?

\[
T(n) \leq \begin{cases} 
1 & \text{if } n = 1 \\
2T \left( \frac{n}{2} \right) + O(n \log n) & \text{o.w.} 
\end{cases} \Rightarrow D(n) = O(n \log^2 n)
\]
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 $\delta$ and list of all points sorted by y
Sort points by y-coordinate by merging two pre-sorted lists.

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

$\Rightarrow D(n) = O(n \log n)$
Recurrences

Above: Where they come from, how to find them

Next: how to solve them
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 $T(n) = a \left(\frac{n}{b}\right)^k + cn^k$ 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 $T(n) = \Theta(n \log n)$
Proving Master Theorem

Problem size

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

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

\[ T(n) = \sum_{i=0}^{d=\log_b n} a^i c \left( \frac{n}{b^i} \right)^k \]
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: \( T(n) = aT\left(\frac{n}{b}\right) + cn^k, \ a > b^k \)

\[
T(n) = \sum_{i=0}^{\log_b n} a^i c \left(\frac{n}{b^i}\right)^k
= cn^k \sum_{i=0}^{\log_b n} \left(\frac{a}{b^k}\right)^i
= cn^k \frac{\left(\frac{a}{b^k}\right)^{\log_b n+1} - 1}{\left(\frac{a}{b^k}\right) - 1}
\]

\[
\frac{x^{k+1} - 1}{x - 1} \text{ for } x = \frac{a}{b^k}
\]
using \( x \neq 1 \)

\[
b^{k\log_b n} = (b^{\log_b n})^k = n^k
\]

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

\[
\leq 2c \ a^{\log_b n} = O(n^{\log_b a})
\]
Solve: \( T(n) = aT \left( \frac{n}{b} \right) + cn^k, \ a = b^k \)

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

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

\[
= cn^k \log_b n
\]
Master Theorem

Suppose \( T(n) = a \cdot T\left(\frac{n}{b}\right) + cn^k \) 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 \( n \leq b \).