Quiz!

Scheduling Discussion

<table>
<thead>
<tr>
<th></th>
<th>Sun</th>
<th>Mon</th>
<th>Tues</th>
<th>Thu-Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS Due</td>
<td>Reflect</td>
<td>Respond</td>
<td>Reflect</td>
<td>Quiz on Canvas/in class</td>
</tr>
</tbody>
</table>

Learning Goals
- Describe closest points D&C strategy
- Apply greedy design strategy

Divide + Conquer Example:

Closest Pair Problem:

\[
\begin{align*}
&\text{Distance between 2 points:} \\
&d(P_i, P_j) = \sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}
\end{align*}
\]

Input: Array containing locations of n points (unique x,y coordinates)
Output: Closest pair of points

Applications:

Q. What is the runtime of an exhaustive search algorithm for closest pair on n points?

A) \(O(n^2)\) \(O(n)\) \(O(n^2)\) \(O(2^n)\)
Divide + Conquer Example:

Closest Pair Problem:

Distance between 2 points:
\[ d(P_i, P_j) = \sqrt{(x_i-x_j)^2 + (y_i-y_j)^2} \]

Input: Array containing locations of \( n \) points (unique \( x, y \) coordinates)
Output: Closest pair of points

Applications:
- Air traffic control
- Robotics
- Detecting repeated sequences of DNA
- Creating 3-D images out of stereo images (matching closest regions that are the same)
- Geography Info Systems: detect double boundaries

Q. What is the runtime of an exhaustive search algorithm for closest pair on \( n \) points?

A) \( O(n^2) \) \[ O(n) \] \[ O(n^2) \] \[ O(2^n) \]

Need to check each pair. \( \binom{n}{2} = O(n^2) \) pairs. Calculating distance for each pair is \( O(1) \).
Q. Suppose the points are on a line:  
\[ x_1 \quad x_3 \quad x_2 \quad y=0 \]  
P_2 \quad P_3 \quad P_1

• Design an \( O(n \log n) \) algorithm to find the closest distance
• If time, try to prove correctness
Q. Suppose the points are on a line: Given array:

\[
\begin{bmatrix}
    x_1 & x_2 & x_3 \\
    y_1 & y_2 & y_3 \\
\end{bmatrix}
\]

- Write pseudo code for an \( O(n \log n) \) time algorithm
- If time, try to prove correctness

1. Sort \( \mathcal{O}(n \log n) \)

2. \( \text{MinDist} = \infty \)
   
   for \( i = 1 \) to \( n-1 \)
   
   if \( (x_{i+1} - x_i) < \text{MinDist} \)
   
   \( \text{MinDist} = x_{i+1} - x_i \)

   \( \text{II} \)

   Loop over sorted points, check distance only between adjacent points. Return Min distance found.

\[
\begin{bmatrix}
    x_1 & x_2 & x_3 & x_4 \\
    y_1 & y_2 & y_3 & y_4 \\
\end{bmatrix}
\]

* Closest pair is adjacent... why?
* Naive still uses \( O(n^2) \), if try to check all pairs
What if sort along X axis, Y axis?

Circled points are closest, but when sort, get separated.

Algorithm Sketch
1. Sort points by X coordinate

2. Divide:
   Split X into left half + right half
   
   \[ \begin{array}{c}
   & L & R \\
   & L & X & R \\
   & R & \rightarrow L \\
   & R & \rightarrow R \\
   \end{array} \]

3. Conquer: Find closest distance in each of L, R

Q: What size set of points should trigger base case of recursive algorithm?

A) 0  B) 1  C) 2  D) 3
What if sort along X axis, Y axis?

Circled points are closest, but when sort, get separated.

Algorithm Sketch

1. Sort points by X coordinate

2. Divide: Split X into left half + right half

3. Conquer: Find closest distance in each of L, R

Q: What size set of points should trigger base case of recursive algorithm?

A) 0  B) 1  C) ≤ 2  D) ≤ 3

Otherwise: 3 gets split into 2 and 1. Can't compare one point to itself.
4. Combine the midline.

If overall closest pair is on either side. But in trouble if closest pair crosses.

Let \( S \) be \( \min \{ CP(L), CP(R) \} \)

Claim*: Only need to look at a region within \( S \) of the midline.

Otherwise: Contradiction

distance greater than \( S \), so not the closest pair.
If squint, looks like points on a line!

1. Sort
2. For-loop to look at nearest neighbors