Which Algorithm is Best

Goals

1. Select appropriate ADT and then data structure for an application.

2. Use variations of standard data structures and algorithms to understand how changes affect correctness and time complexity.

3. Intro into algorithm design (Main topic of CSE 441T)

4. Quickly determine how efficient an alg or data structure will be
5. Be able to implement what you learn and apply to a variety of applications.

Example Problem

Find the closest pair of points
Brute force alg
Consider every pair of points
Compute distance
remember min

Let $n$ be # of points
Will this alg be efficient as
$n$ gets large?
Suppose we think of 4-5 algorithms if we've argued they always yield the correct answer.

How can we determine which will be most efficient, especially as \( n \) gets large?

Why not implement all of them and run them?
Problems
- Takes a lot of time to implement & test
- Time complexity often depends on input itself
- What data size should you use?
What does time complexity depend on?

- input itself
- input size \( (n = 100 \ vs \ n = 1,000,000) \)
- hardware (computer) \( \} \) machine dependent
- compiler \( \} \) machine dependent

Focus on a machine-independent analysis.
Asymptotic Time Complexity

machine independent, rough measure of time complexity as a function of input size \( n \)

“Back of the envelope” calculation
What are we going to measure?

# of statements (lines of code) executed

need to be careful that all statements in your high-level language take “roughly” the same time.
Input size vs Execution time
Input Size vs Lines of Code

Lines of code executed: merge sort vs. insertion sort

Number of elements in array

Lines of code executed:
- Insertion Sort
- Merge Sort
How do we account for dependencies in the input?

Worst-case analysis — consider the input of the given size that is slowest.

Expected-case (average-case) analysis — assume some distribution over the data.
$n^2$ vs $n \log n$
Asymptotic Growth Rates

# statement s

\begin{align*}
C & \quad \text{constant} \\
C \cdot \log n & \quad \text{logarithmic} \\
C \cdot n & \quad \text{linear} \\
\underline{n \cdot \log n} & \quad \text{quadratic} \\
C \cdot n^2 & \quad \text{quadratic} \\
C \cdot n^3 & \quad \text{cubic}
\end{align*}
What is the asymptotic time complexity for brute force algorithm of comparing all pairs of points?

- Compute distance and compare with best so far
- Do this once for each pair of points.
If you've taken CSE 240 or equivalent
I hope you can quickly say

\[ C(n,2) = \binom{n}{2} = \frac{n(n-1)}{2} \]

\[ = \frac{n^2}{2} - \frac{n}{2} \]

\[ \leq C \cdot n^2 \]
Otherwise we can compute the number of pairs of points as:

\[ \text{# points it is compared against} = \frac{n(n-1)}{2} \]

1 + 2 + \ldots + (n-2) + (n-1)

(n-1)/2 pairs