Recursion

Example applications
&
Problems to work in class
Recursion for geometric problems

Efficient algorithms for 2D and 3D graphical and spatial algorithms are often recursive

Examples:

• Simplify a river displayed on a map (from thousands of points to tens)
• Detect collisions between objects in a video game
Simplifying lines (in a map)

Essentially all mapping programs use the Ramer-Douglas-Peucker recursive algorithm to simplify complex lines like rivers and roads.

A river may be represented by thousands of points, but we want to draw with only a fraction of them (especially when zoomed out).
Simplifying lines (in a map)

Original line, to be simplified

Subdivide at point C that is furthest from line from first to last point.
Recursively simply segments before and after C
Recursively subdivide until close enough

Progress case: Some points are too far from the line segment. Divide at worst point, simplify both.

Base case: Good enough
No omitted point is more than allowed error from straight line
Collision Detection

Problem: Thousands of game objects ... which do I bump into as I move forward? Do this fast enough for a 20 frame-per-second video game.

Solution: Recursively subdivide objects in space (To directly compare to just the few “closest” objects)
Which lego-Darths do I collide with?
Half the Lego-Darths on each side of division
Path

Approximately half
Path

Approximately half
Compare only to the lego-Darths in the region my motion passes through.

Path

Approximately half
R-tree (linked) data structure
(linked structures next week topic)
Let’s design and code!

• Reverse an array
• Compare strings for “almost equal”
  • (ignoring non-letters)
Reverse an array

Consider sub-array from i to j
(Starting with i=0, j=a.length – 1)

What are the base cases?
What is the progress case?
Reverse a[ i ... j ]

Cases depending on j-i (length of sub-array):

1 or less: Base case, nothing to do

More than 1: Swap a[i] with a[j], then reverse a[i+1 ... j-1]

Let’s write this as Java
“Almost equal” (except for non-letters)

Compare strings S1 and S2
Return true if they are “almost” equal (matching except for non-letter characters)

Use Character.isLetter(ch) to test a character

What are the base cases? What is the progress case?