Overview

Data:

1D, 2D, 3D, ...
points, lines, rectangles, polygons,
in memory, on disk

Major problems:
- range searching
- intersection
- nearst neighbor
- point location
- containement

Scenarios:
- static problem
- dynamic problem
- allow pre-processing
- trade-off query time and space

Approaches:

1D
- B-tree
  - CPU and I/O good
  - range searching and others

2D

hierarchical space partition:	grid, quadtree [general], BSP,	kd-tree, range-tree [specialized]
hierarchical data partition:	R-tree [general]	

d-D (d >= 3)
- the 2D structures can be extended to d-dimensions
- but.. the curse of dimensionality (grow exponentially in d)

So given a problem, not necessarily one of the above, you can do two things: a data driven approach (specific or general), or a space-driven approach (specific or general).

Which means: take a shot at it with a quadtree, and with an R-tree. They are versatile structures but they do not give worst-case bounds. You can always try to do better by designing a customized solution specific to your problem.

How? using the insight you gained by looking at other structures.

Learn the techniques, get a big picture of the field.
Refine intuition and critical thinking.
Learn to search and find relevant results, and analyze.

Goals of this class:
- overview of spatial data problems and solutions
- ability to skim over a research paper and figure out the contribution, analyze it and put it in context
- get a better appreciation of algorithms and why we feed you big-oh analysis
- connect algorithms with applications and programming
- refine programming and ... debugging skills
- get to love C for its simplicity, power, and speed. And because it lets you shoot yourself in the back. Or not.