CS 499 (Fall 2006)- Assignment 2
Due: Tuesday, 09/05/2006

(1) Here, we are going to look at shortest path planning in a geometric environment. Specifically, assume that you have buildings in two-dimensional space, which we assume to be axis-aligned rectangles. You are also given a starting and ending location, both outside of buildings. Your goal is to compute the shortest route (in terms of total distance traveled) from the starting to the ending location. Of course, you cannot go through walls. We make the following simplifying assumptions: (1) You are point-sized. (2) You can change direction arbitrarily at any point.

Let's say that the input is specified as follows: the first line is an integer n, the number of buildings. Then, each building is described by 4 real numbers x_w, x_e, y_s, s_n on a line, the x-coordinates of the west and east walls, and the y-coordinates of the south and north walls. Finally, a line with 4 real numbers x_0, y_0, x_1, y_1 specifies the starting and ending points of the route. Your program should output the length of the shortest route (if you want, you can also output the route), and work for fairly large n, say, up to 100,000. Please use heaps instead of a naïve implementation, preferably your own heap code from the previous homework.

Notice that in principle, the set of points/nodes here is infinite (the entire plane). So if you are going to use Dijkstra, you will need to think about how to reduce this to a finite number of nodes. This will take a bit of thinking.

Also, since we have not covered geometric algorithms yet, I am not expecting you to know how to check whether two lines intersect (which you will likely need). Here is code that checks if lines $a = [(x_{a1}, y_{a1}), (x_{a2}, y_{a2})]$ and $b = [(x_{b1}, y_{b1}), (x_{b2}, y_{b2})]$ intersect:

```
float signedarea (float x1, float y1, float x2, float y2, float x3, float y3)
{ return ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2; }

char intersect (float xa1, float ya1, float xa2, float ya2,
                float xb1, float yb1, float xb2, float yb2)
{
  return (( signedarea(xa1, ya1, xa2, yx2, xb1, yb1)
            *signedarea(xa1, ya1, xa2, ya2, xb2, yb2) <= 0)
          && ( signedarea(xb1, yb1, xb2, yb2, xa1, ya1)
              *signedarea(xb1, yb1, xb2, yb2, xa2, ya2) <= 0));
}
```

The idea here is that `signedarea` computes the area of the triangle given by the 3 points, with a positive sign if the points are in clockwise order, and negative sign if they are counter-clockwise (or maybe the other way 'round). Two lines intersect if and only if (1) the endpoints of b lie on different sides of the line a, and (2) the endpoints of a lie on different sides of the line b. This is equivalent to the triangles having different signs, which is what `intersect` tests.

(2) In order to be able to solve the previous problem, we made a lot of simplifying assumptions (point-sized person, arbitrary turns, only rectangular axis-aligned houses). Give some thoughts to how the problem would change if we tried to consider:

1. What happens if the person is not a point, but a circle? How about an axis-aligned rectangle or square that doesn't rotate? How about an arbitrary shape which is allowed to rotate?

2. What happens if buildings can be arbitrary polygons? How about if they are convex polygons (no "dents": formally, if two points are inside a building or on a wall of a building, then their connection must entirely be inside the building)?
3. What if we look at the "real" objective of time. Now, our person can accelerate gradually, but no more than a certain amount of speed per second, or slow down. Also, the person can change direction, but only gradually (think car or bike), say no more than a certain angle per second. How can we model the search now, if our goal is to arrive at the destination as fast as possible? Can you think of scenarios where the route found under this objective is very different from the one found in problem 1?

Obviously, any one of these, and combinations, would make the problem more difficult. Give some thought to which of the variations you feel you can address (and which you can't), what types of difficulties these are, etc. Notice that many of these are areas where top researchers are actively working, so no one is expecting you to come up with solutions. The goal is to get you thinking, and identify difficulties and what might or might not remedy them.