CSE546: Computational Geometry
Extra Credit Project: 3-Slab of Minimum Height Applet

Brett Gilpin and Sophie Veksler

December 17, 2017

Project Summary: We created an applet to visualize the algorithm to find a $3-slab$ of minimum height from a collection of n points P in a plane. A $3 - slab$ is defined to be the region bounded by a pair of non-vertical parallel lines, such that there are at least 3 points in the region (including on the lines) and the height of a $3 - slab$ is the vertical distance between its two lines.

Background: The algorithm we used to develop this applet is based on the this year's Homework 4, Question 3. We add a set of at least 3 points to the plane, using these points to create a line arrangement in the Dual Plane. Once all of the desired points are added, the line arrangement is stored in a Doubly Connected Edge List (DCEL). Then, for each intersection in the line arrangement, we check to see which other Dual Line in the line arrangement is closest vertically to this intersection. We keep track of the shortest vertical distance so far, as well as which intersection and which line had the shortest vertical distance. Finally, we output the intersection in the dual plane with the shortest vertical distance to the nearest line, and that line. The intersection we output represents the line intersecting the two points in the primal plane that represent the intersecting lines in the dual plane, and the dual line we output represents the point through which the line parallel to the former will go through in the primal plane.

Note: in order for the applet to work, you must add at least 3 points. The applet does work for large point sets, but it takes a long time, so we recommend keeping the point set to 10 points or less. At the time of submission, calculating the 3-slab with 10 points in the set will take at most 3 minutes. This was calculated based on our display time.

Pseudocode:

```
P = set of points
DCEL = arrangement of dual lines
s = shortest line so far
b1, b2 = primal points / dual lines of intersection with shortest vertical distance
so far
b3 = primal point / dual line with shortest vertical distance to b1/b2 intersection
for point Pi in P:
     for point Pj in P:
          v = intersection of Pi* and Pj* in the dual plane
          check up to closest 4 intersections in DCEL to find least vertical distance
          contender = dual line with closest vertical distance to v
          vert = vertical line from v to contender

          if (vert.length > s.length):
               b1, b2 = Pi, Pj
               s = vert
               b3 = point in primal plane that is represented by contender
line1 = line intersecting b1 and b2
line2 = line intersecting b3 that is parallel to line1
return line1, line2
```

Visualization: Our visualization shows from the above pseudocode the step where we check in the DCEL to get the line with the closest vertical distance. We do this by changing the colors of the intersecting lines to aquamarine, and the line that we are checking against to orchid. At the end, we show the lines that are stored by changing them green and adding the two black lines as the 3-slab.

Our approach: First, we let the user add as any points as they want to the primal plane, and automatically draw these points as dual lines in the dual plane as well. Once the user is ready to calculate the 3-slab, they can click a button to start the calculation. This then converts the line arrangement into a DCEL, and checks each intersection in the DCEL to find the line with the closest vertical difference, keeping track of which intersection and which line that distance corresponds to. Lastly, we output two lines that show the 3-slab of minumum height. Note: we decided, for ease of implementation, to construct our DCEL in $O(n^2 logn)$ time. However, this can be done in $O(n^2)$ time, which brings the runtime of the whole algorithm to $O(n^2)$.
To implement our applet we used: HTML/CSS JavaScript (jQuery for ease of access to elements on page, fabric.js for ease of visualization).