Lab 4, Friday, January 21

Swap roles (the person sharing their screen) each time you complete a step.

Step 1: If you did not do so yesterday, work together today to complete Lab 3.

Step 2: Modify the setup code so that it creates the same number of red and blue vertices that are connected in pairs using links, but with the red and blue vertices in random locations. Then make sure that your three procedures from Lab 3 still work. If they don't work, modify them to make them work.

Step 3: If you have time, start to implement the Dijkstra algorithm (without agents) for computing the single-source shortest paths (SSSP) from a root vertex to all other vertices in a connected positively weighted graph. (This is the start of HW4) Some sample pseudocode for this algorithm is on the following page.

You do not need to submit Labs 3 or 4. But you need to submit HW1 below. You will use what you learn in Labs 3 and 4 for HW1.

HW1, due Tuesday January 26 in the course DropBox by 8am EST

Submit HW1 individually but indicate your lab partner on the comments. Be sure to use the indicated name (with dashes, and not underscores),

Instructions; Implement two versions of the Dijkstra SSSP algorithm on a connected weighted graph. The edge weights are positive integers, not related to Euclidean distance. The user should be allowed a random graph. Allow a random root or user selected root for the search.

One version should be a standard algorithm animated only by highlighting the edges as the algorithm runs, and showing the resulting SSSP tree.

The second version should use agents to perform the search. Note that the speed of the agent should be set so that length of travel time along an edge is proportional to the weight (and not Euclidean length) of an edge. Both algorithms should correctly compute the SSSP tree from the given root.

BONUS: Let the user choose between a random graph, a graph from a file, or graph from user input.

SUBMIT: Name your code as follows: **`Lab2-LastnameFirstinitial.nlogo`**. Submit the final code to the drop-box before the next class period. HW will be graded on:

- Correctness and functionality.
- Style and structure. (Good use of top-down design and procedural abstraction.n)
- Use of Netlogo primitives.
- Commenting.