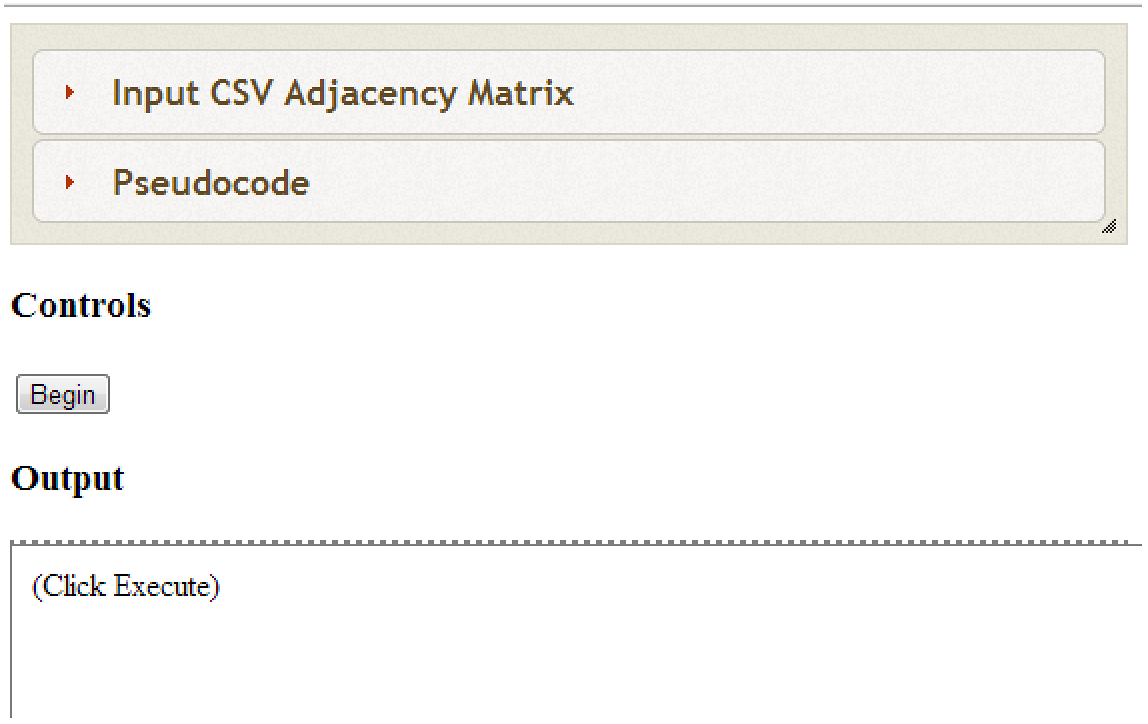
Pat Kujawa, CSCI 466 Networks, Dijkstra Project

README

I chose to do my project using the web-based graph visualization framework that Eric Spaulding and I have created under the direction of Mike Rosulek. I created an HTML page that uses our javascript libraries (it's all completely front-end) to load a CSV adjacency matrix as a graph, lay it out using a force-directed algorithm, and allow you to step through the execution.

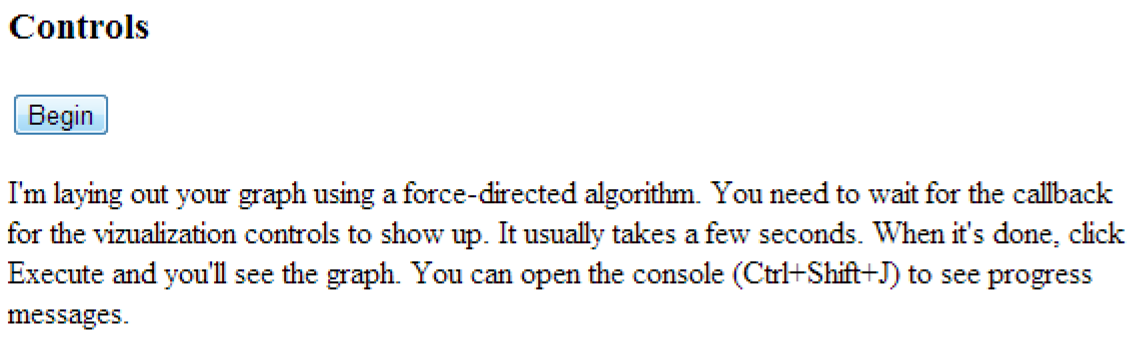
To use it, just unzip the massive collection that is our library and open "test-csv-adj-matrix,dijkstra,springy.html" in Chrome (or another browser, but I haven't tested in others). Assuming I don't have any absolute links or anything, the page should run and look like the following.



(Click Execute)

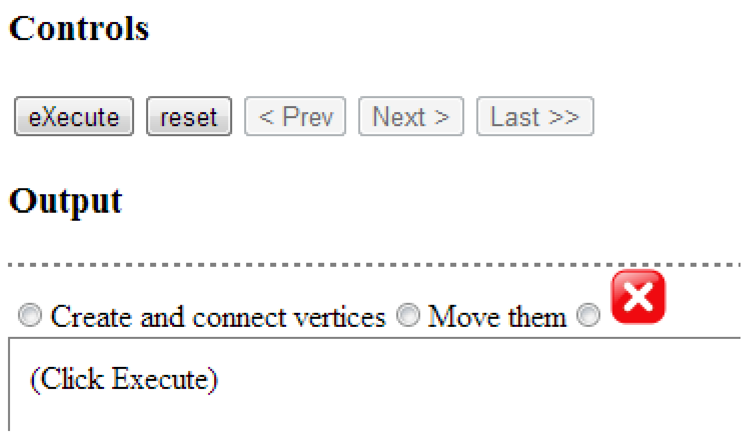
You can click on those accordion headings to input the adj matrix (there's one already there for convenience) or view the interactive pseudocode.

To start the algorithm, click **Begin**. You'll see the page change like so:



I'm laying out your graph using a force-directed algorithm. You need to wait for the callback for the vizualization controls to show up. It usually takes a few seconds. When it's done, click Execute and you'll see the graph. You can open the console (Ctrl+Shift+J) to see progress messages.

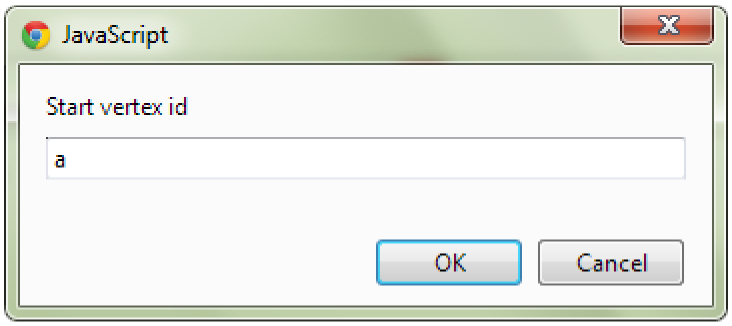
When it's done laying out the graph, you'll see controls to display the graph and run Dijkstra's. Click **eXecute** (or Alt+x) and you will get prompted for the names of the start and finish vertices.



Controls

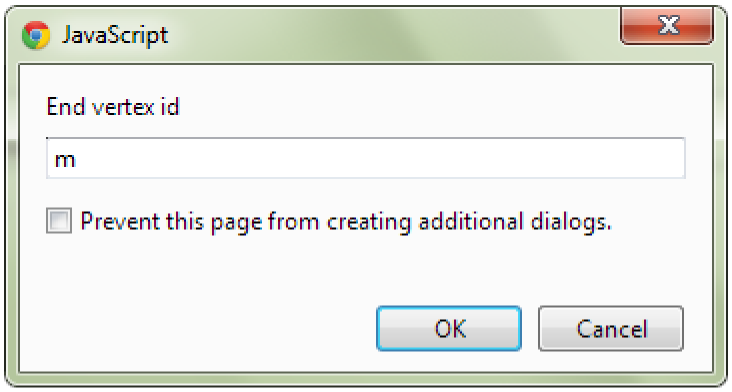
Output

(Click Execute)



Start vertex id

a

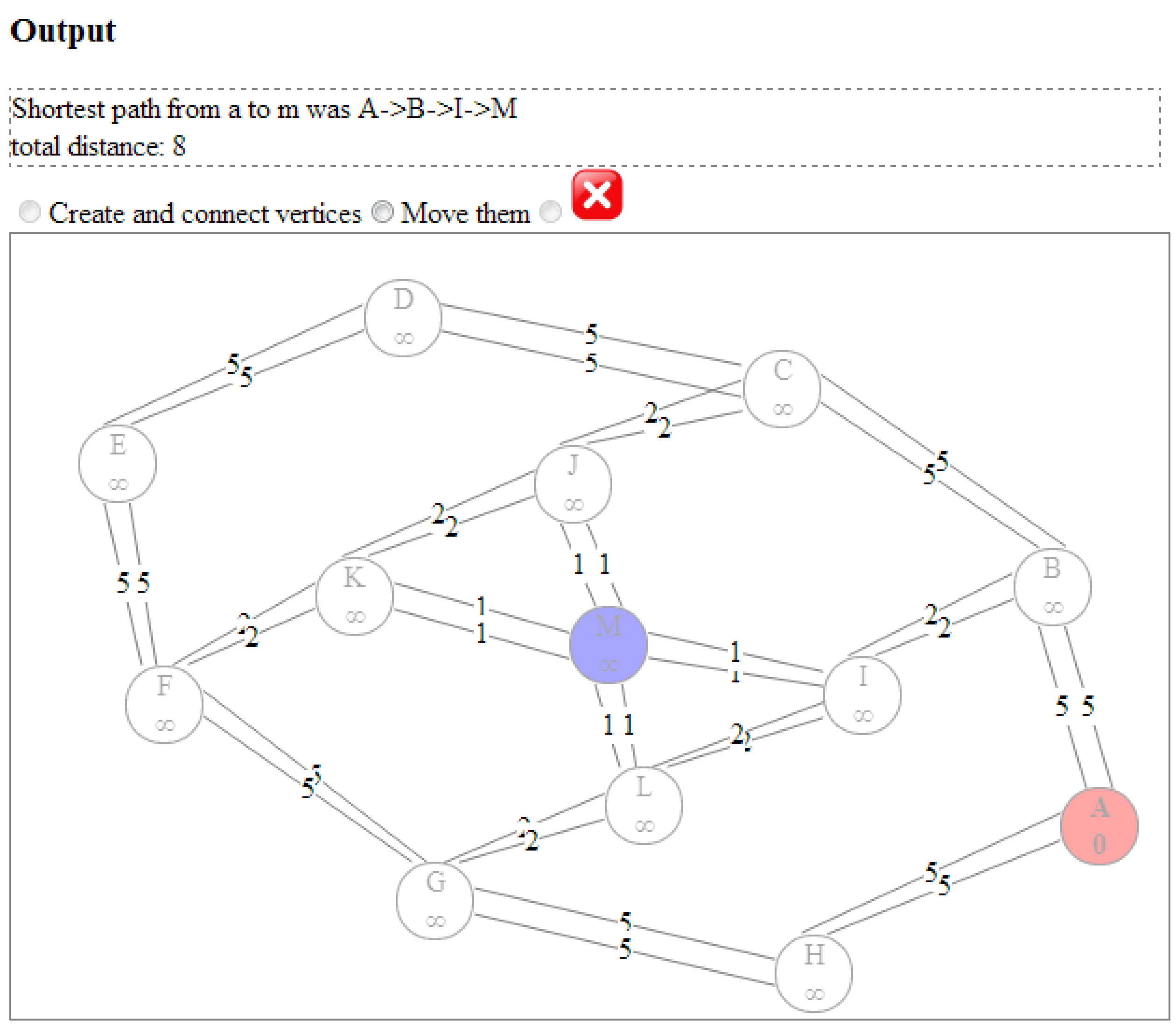


End vertex id

m

Prevent this page from creating additional dialogs.

Fill those in with the labels of the vertices (case shouldn't matter, but other errors/typos fail silently, although you can click execute again to try over) and you should see the graph, as well as output on what the shortest path was.



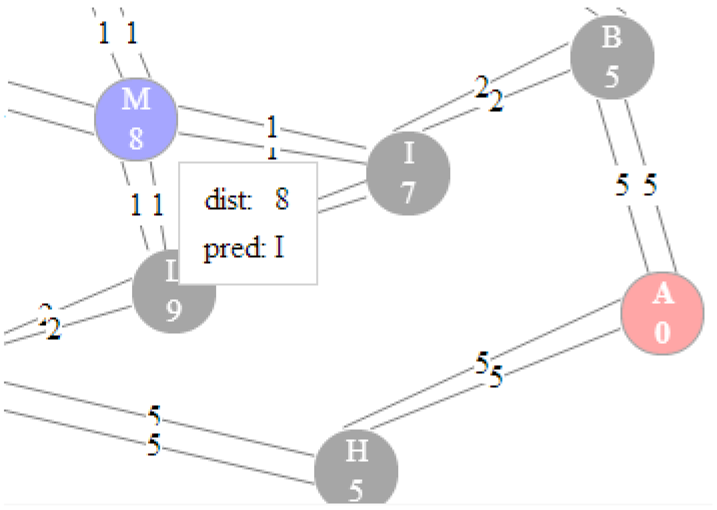
Output

```
Shortest path from a to m was A->B->I->M
total distance: 8
```

Create and connect vertices Move them

You can use Alt+P and Alt+N to step through the Prev and Next frames (and the pseudocode will highlight to indicate where you are, although I had issues with the graph position if the pseudocode accordion wasn't open the whole time). You can also click **Move them** to drag the vertices around and make it pretty (but once you click Next, you can't move them any more).

If you click **Last**, you can cut to the chase and show the distances on each vertex. (I somehow broke the path edge labeling, so it isn't obvious who is who's predecessor, but if you hover over a vertex, this info will be displayed.



Functions

Since I used javascript and our viz library, I have a ton of functions written. However, the important ones are in the av-GraphModel.js file. That's where we provide routines for querying adjacent nodes, loading data, querying edges, etc. The actual Dijkstra code is in the HTML file, at the bottom, inside of an object in a function (not pretty, I know). All it does is use an array as a priority queue and then run through the straightforward Dijkstra algorithm shown in the pseudocode.