1. PageRank intuition
    - Go to http://projects.si.umich.edu/netlearn/GUESS/pagerank.html.
    - You'll see a small, directed, network, and by clicking on the 'iterate' button, you will be calculating the PageRank of each node. It will take several iterations for the algorithm to converge. At each iteration, the probability that a random walker is found at any given node A is proportional to the probability that it was on a node B with a directed edge to A, divided by the outdegree of node B.
    - The edge width in the visualization is proportional to the probability that a node transitions from B to A. Can you explain the different widths you see?
    - Approximately how many iterations does the algorithm take to converge?
    - Try increasing the teleportation probability. How does this influence the PageRanks assigned to the nodes?
    - Try allowing sinks. Without sinks allowed, once a random walker reaches a node with no outgoing edges, it jumps randomly to another node. With sinks allowed, it stays at that node with probability (1-teleportation) and jumps to a random node with probability = teleportation. What affect does allowing sinks have on the distribution of PageRanks?

2. LexRank - summarizing text
    - Select a piece of text (10-20 sentences) that you would like to summarize and paste it in the appropriate box of the LexRank demo: http://tangra.si.umich.edu/clair/lexrank/.
    - If you are unable to paste text into the text box (where currenty there is some text about an Iraqi official), try a different browser. I've tested this on the DIAD PCs and it works there...
    - There are two parameters you can vary: (1) the cosine similarity threshold determines how similar two sentences have to be in order to share and edge. (2) the salience threshold determines how high a sentence's PageRank has to be in order for that sentence to be included in the summary. Vary the cosine similarity threshold and record the most salient sentence. Does the most salient sentence change as you vary the threshold? Accordingly, report on a cosine similarity threshold that gave you the best result (if applicable).
    - Compare the 1 sentence summary to the 2 or 3-sentence summary. In your opinion, how much do the 2nd and 3rd sentences add (in terms of adding more information). Would you have chosen them, or a different sentence? Relate your answer to the structure of the lexical similarity graph.
3. Word translation
    - In Pajek, open the file words.net. It contains words related to the English word 'spring'. English words are labeled as e_someword, 'f' for French, 'g' for German, 'h' for Croatian (hrvatski). Your task will be to find translations of the different english meanings of 'spring' into croatian using this network.
    - Find the pairwise dissimilarities of nodes in the network
    - Cluster > Create complete cluster

- Operations > Dissimilarity (experiment with different ones)
- You will be prompted to save the dendrogram to an .eps file. View it and interpret it. (*I*)
- Draw a network using the dissimilarity values
- + in the draw menu (Options > Values of lines > dissimilarities)
- + Layout (use your favorite spring layout algorithm) (*I*)
- + Are related nodes close together (if you don't know french or german, ask a neighbor)?
- Save a permuted matrix that should have grouped related terms together
- + make sure you re-select the original matrix and not the full dissimilarity matrix
- + File > Network > Export matrix to EPS > using permutation
- How many different meanings of the word 'spring' are captured? Which Croatian words correspond to each meaning?
- Did you find any same-language synonyms using the hierarchical clustering?

4. PageRank: make your own network

Construct a small, directed network (about 10 nodes) in GDF format and load it into GUESS. For a reminder of what a directed network looks like in .gdf format, consult the file test.gdf. Construct it such that you have at least one node that will have low indegree but high PageRank.

Compute the PageRank of each node by typing

```
g.nodes.pagerank
```

Color by PageRank

```
colorize(pagerank,green,yellow)
```

Compute the indegree

```
g.nodes.indegree
```

Size the nodes by indegree

```
resizeLinear(indegree,minsize,maxsize) // (you are choosing minsize and maxsize)
```

Turn in an image of your network (*I*). Point out a node that has high PageRank but low indegree. Explain qualitatively how this came about.

*an aside: You can also use the GUESS toolbar pageranktoolW.py, if you'd like to see how the algorithm converges...