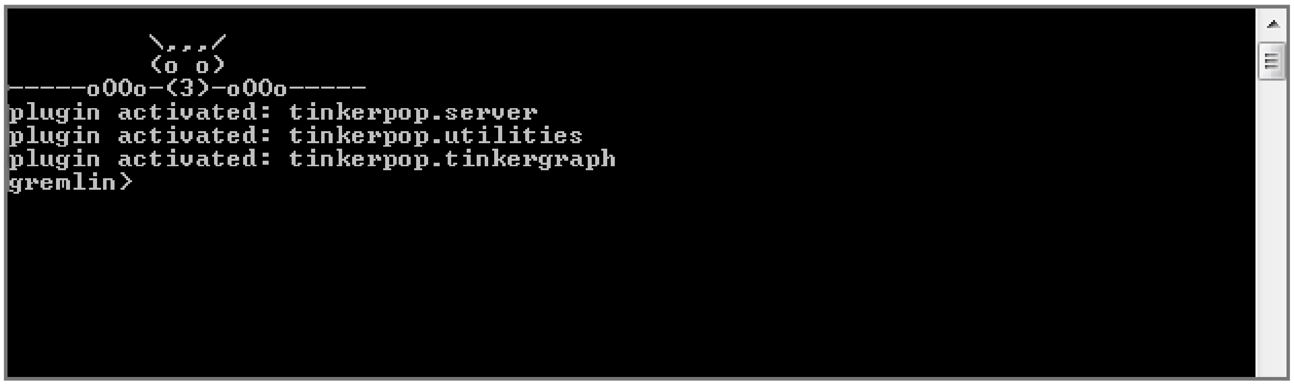
HW4: assigned 11/4, due 11/14 by 11:59 PM

Total points: 1+1+2+2=6 [plus bonus: 0.25+0.75=1]

For this **graph traversal** assignment, you'll need to download TinkerPop console 3.3.4. (http://mirror.cc.columbia.edu/pub/software/apache/tinkerpop/3.3.4/apache-tinkerpop-gremlin-console-3.3.4-bin.zip) Simply unzip the file, and run bin/gremlin.bat if you're on a PC, or bin/gremlin.sh on the Mac - you should then see this (http://tinkerpop.apache.org/docs/current/tutorials/getting-started/):



```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

MacBook users [thanks to Gitesh Chopra for the heads-up]:

• make sure you have installed Java Version 8

• install Maven: https://maven.apache.org/install.html (https://maven.apache.org/install.html)

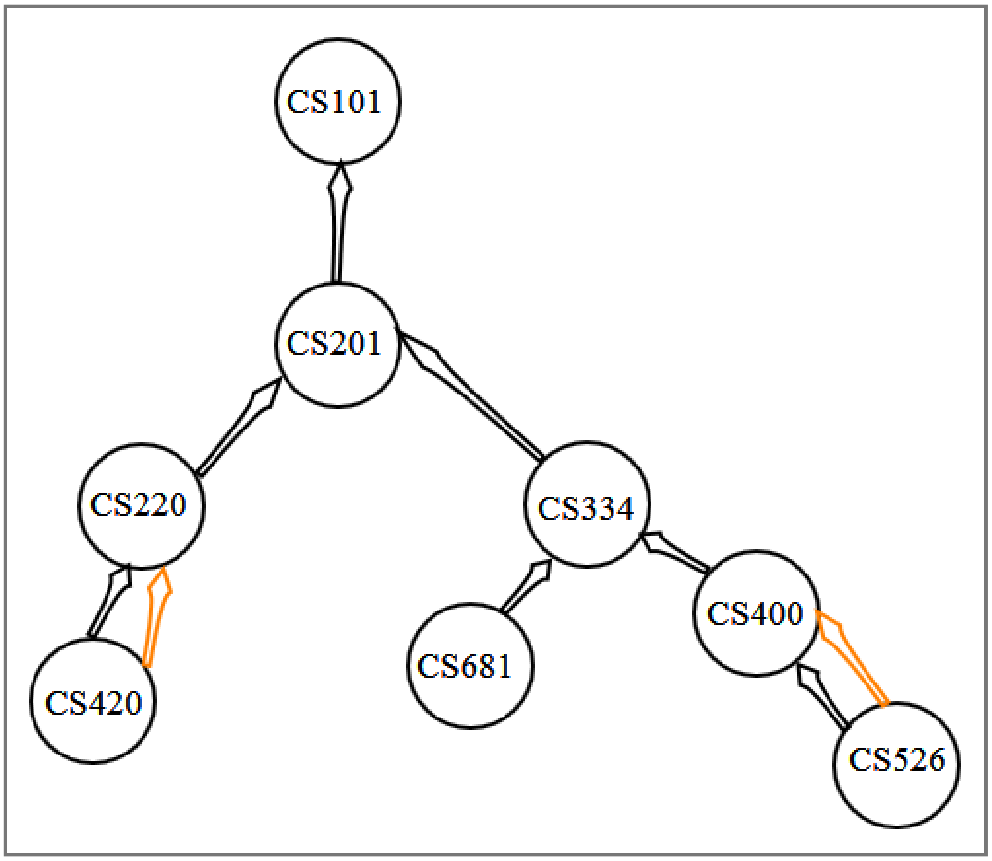
• if your macOS is High Sierra, place the extracted Gremlin console folder under APPLICATIONS, then run gremlin.sh

That was quick and painless, congrats - now you're all set to explore graph computing :)

Spend some time, going through tutorials (http://tinkerpop.apache.org/) (just search online for more). In particular, learn how to create an empty graph for traversal, and add to it, vertices and edges. Also, look up commands such as unfold(), filter(), repeat(), out(), etc. - you'll need these for the graph queries you'll need to write.

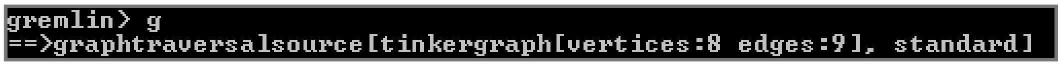
NOTE: posting these queries on SO (StackOverflow) etc. and submitting responses as YOUR answers in considered CHEATING, and goes against what I keep emphasizing, which is 'LEARNING.' If you do this you are going to get penalized accordingly.

Consider the following graph that lists courses (circles) and prereq relationships (black arrows). Eg. CS101 is a prereq for CS201, and CS334, for CS400 (and so on). The two orange arrows are 'coreqs' (courses that may be taken together, possibly upon instructor approval for example). The orange arrows obviously lead to double connections: between CS420 and CS220, and between CS526 and CS400. If you'd like to think of nodes in terms of nouns, and edges in terms of verbs, our two kinds of edges would be 'requires pre-req' and 'is a co-req of'.



Q1. **Write a Gremlin command** that creates the above graph [hint - you will also need a 'traversal' for it]. The command could be a multi-statement one, or a single line one (with function chaining).

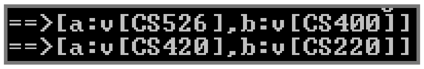
If you called the traversal 'g', echoing it on the console would display this:



```
gremlin> g
==>graphtraversalsource[tinkergraph[vertices:8 edges:9], standard]
```

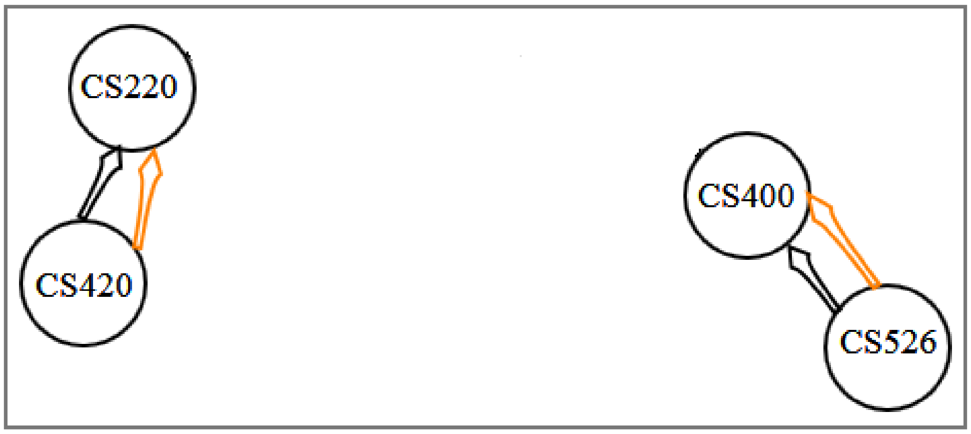
As you can see, the result confirms that we seem to have correctly built up the graph shown in the diagram (using other queries you can print out all the nodes and arcs, to ensure this 100%).

Q2. **Write a query** that will output JUST the doubly-connected nodes, ie. something like

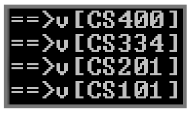


```
==>[a:v[CS526],b:v[CS400]]
==>[a:v[CS420],b:v[CS220]]
```

As you can see, the output matches what we'd expect, given our doubly-connected nodes:



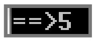
Q3. **Write a query** that will output all the ancestors (for us, these would be prereqs) of a given vertex. Eg. running it on CS526 would produce



```
==>v[CS400]
==>v[CS334]
==>v[CS201]
==>v[CS101]
```

As expected, the prereqs chain for 526 is 400->334->201->101

Q4. **Write a query** that will output the max depth starting from a given node (provides a count (including itself) of all the connected nodes till the deepest leaf). This would give us a total count of the longest sequence of courses that can be taken, after having completed a prereq course. Eg. running it on CS101 would produce



```
==>5
```

This reflects the fact that 101->201->334->400->526 is the longest chain (with length 5) in our course list.

For all four questions Q1-Q4, include explanatory notes (in your OWN words) for how (why) the query works, ie. what the various graph functions do and how they fit together. Your explanations need to be complete, and need to make sense (simply stringing together TinkerPop docs for the commands is unacceptable). And, if you "collaborate" on this (we find that your wording resembles someone elses'), YOU WILL GET A ZERO for ALL the questions!

BONUS (0.25+0.75=1 point). Leonhard Euler is credited with founding the now-vast, hugely-important field of graph theory, when he investigated the '7 bridges of Konigsberg' problem. You are going to use Gremlin to verify his answer to the famous problem.

This (https://en.wikipedia.org/wiki/Seven_Bridges_of_Konigsberg) wikipedia entry has all the graph-theory knowledge you need, to do the bonus!

a. **Write a command** that creates a graph of the 7 bridges and 4 regions of Konigsberg.

b. **Write an 'Eulerian circuit' detection query** that runs on your graph, that outputs 'false' (to signify what Euler showed - that there is NO path through the bridges and cities where, starting at any region of the city you traverse each bridge just once, and return to the starting point, ie. there is no closed path, aka circuit). If the graph were modified (eg. an existing bridge is removed, and a new one is added) so there is now an Eulerian circuit, your query needs to emit 'true' for such a case. Note - there is no need to output the sequence of nodes in the circuit (when there is a circuit), just outputting 'true' will do; in other words, your query just needs to report on the presence/absence of a circuit, which is much easier to do than to make it compute an Eulerian closed path if one does exist.

'One of these days' you could create a graph of US' freeway map (pics/freeway-map.gif) [showing the major freeways and cities they connect], modify your query to look for an Eulerian path as well [in addition to circuit], and run your query on it to see if an Eulerian circuit or path exists :) You could implement Fleury's algorithm (http://www.geeksforgeeks.org/fleurys-algorithm-for-printing-eulerian-path/) for this. Note that this algorithm would output a sequence of edges (vertex pairs) that comprise a loop/path if one does exist [so you could traverse the circuit/path for real!].

If you want even more practice with Gremlin, you can try implementing a variety of graph algorithms using it, including MST, shortest path, connected components, clique detection, maximum flow.. Here are two good sources of graph algorithms: GeeksForGeeks (http://www.geeksforgeeks.org/category/graph/), Stony Brook Algorithm Repository (http://www3.cs.stonybrook.edu/~algorith/major_section/1.4.shtml).

Euler would have killed to get his hands on Gremin! But this is a Catch-22 of sorts..

ALTERNATIVE BONUS (0.25*4=1 point).

If you are unable to formulate a query for detecting Eulerian circuits for the BONUS above, you can do the following INSTEAD (NO need to attempt BOTH bonuses, you will not get double bonus points if you do!) - you might find this to be a much simpler alternative :)

Express Q1 as TWO commands [or ONE] (0.25 points); **Q2, Q3, Q4 as a SINGLE command each** (0.25*3=0.75 points).

What if your answers for Q1-Q4 already look like the above (compacted)? Well, then you get a free bonus point, congrats :) You can submit your Q1-Q4 verbatim, as your bonus as well.

What to submit: a single text file with the four commands/queries and explanatory notes for each; if you attempt the bonus question, a separate text file with your answers (graph construction, query).

Note: none of your queries need to contain full-blown Java, Python, etc. code (with function/class defs, imports, loops and branches) - instead, they all need to be written using just the built-in Gremlin classes and methods available at the console level (eg. Graph, values(), select() etc.). You CAN use shell-level commands such as assert() if you like.

Note again: for Q2,Q3,Q4, on account of function-chaining magic, the queries could be as few as a couple of lines, BUT THEY DON'T -HAVE- TO BE - multi-statement queries are totally acceptable! Likewise, your Q1 graph creation commands can be composed of multiple statements; or, you can create a graph-spec JSON file and read the file and create your graph - we accept any and all techniques.

That's pretty much it, have fun :) Learning Tinkerpop/Gremlin can be VERY useful for your data-science future! This system is capable of processing graphs with one TRILLION (no kidding) edges..
