## PROBLEM 7

Some random graphs have more structure than others. For the graph that represents the internet, nodes are web sites and there is a directed edge from one site, say x, to another site, say y, whenever site x has a link to site y. There are a few sites that are massively popular, like Google, Facebook, Twitter, and EdX, that have lots of incoming edges. However, most sites have very few incoming edges.

It is interesting to build random graphs that mimic this structure. The best way to do this is to grow the graph. There are $n$ nodes and initially no edges. Edges are added at random, but in a non-uniform way.

To add in new edges to a graph, first choose a node x at random. Then choose another node y with a probability proportional to the popularity, or *connectedness*, of the node. For example, if a node has $k$ edges it is twice as likely to be chosen than an node that has $\frac{k}{2}$ edges. Finally, add the edge from x to y to the graph.

Each node has a set of incoming edges and a set of outgoing edges. Each node has an *in-degree*, which is the number of incoming edges to the node, and an *out-degree*, the number of outgoing edges from the node. An edge cannot connect a node to itself and there can be at most one edge from a given x to a given y.

Suppose whenever a new edge—say it is `(x,y)`—is added, there are three things updated:

- node y is added as an outgoing edge (or "out-edge") of node x
- node x is added as an incoming edge (or "in-edge") of node y
- `(x,y)` is added to the list `allEdges`

## PROBLEM 7-1 (1/1 point)

Now consider how to write some code to add an edge to a graph `G`, with all the nodes of `G` contained in the list `G.allNodes` and all the edges of `G` contained in the list `G.allEdges`. The items in `G.allEdges` are pairs of nodes, such as `(x, y)`. `(x, y)` indicates a directed edge from node `x` to node `y`.

Assume we have built a graph `G` according to the above rules. Consider the lines of pseudocode:

```
z = random.choice(G.allNodes)
(x,y) = random.choice(G.allEdges)
add new edge z -> y
```

True or False? The node `y` is chosen with probability proportional to its popularity.

[✓ True] ✓

False

*You have used 1 of 1 submissions*

## PROBLEM 7-2 (1/1 point)

To avoid selecting a self-edge (an edge from `z` to `z`), all edges pointing to `z` are first removed from `allEdges` before making the choice.

True or False? The following Python expression creates a list of all edges that does not include any edges into node `z`:

```
allEdgesExceptZ = []
for (x,y) in G.allEdges:
    if y != z:
        allEdgesExceptZ.append((x, y))
```

[✓ True] ✓

False

*You have used 1 of 1 submissions*

## PROBLEM 7-3 (1/1 point)

The time to construct the list `allEdgesExceptZ` is $O\left(E^2\right)$, where $E$ is the number of edges.

True

[✓ False] ✓

*You have used 1 of 1 submissions*

## PROBLEM 7-4 (1/1 point)

Consider the following procedure used to initialize a graph with `n` nodes:

```
def initializeGraph(n): # n is an integer, the number of nodes in the graph
    G = siteGraph() # Initializes an empty graph, with G.graphNodes set to []
    for i in range(n):
        G.graphNodes.append(newNode(i)) # newNode takes one parameter, the number of the node
    for i in range(n):
        x = G.graphNodes[i]
        y = G.graphNodes[ (i+1) % n ]
        x.addOutEdge(y)
        y.addInEdge(x)
        G.allEdges.append((x, y))
    return graphNodes
```

True or False? The procedure `initializeGraph` ensures that there is at least one path between any two nodes in the graph.

[✓ True] ✓

False

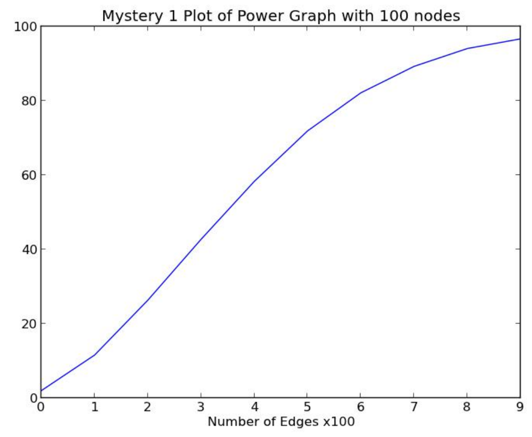
*You have used 1 of 1 submissions*

## PROBLEM 7-5 (4/4 points)

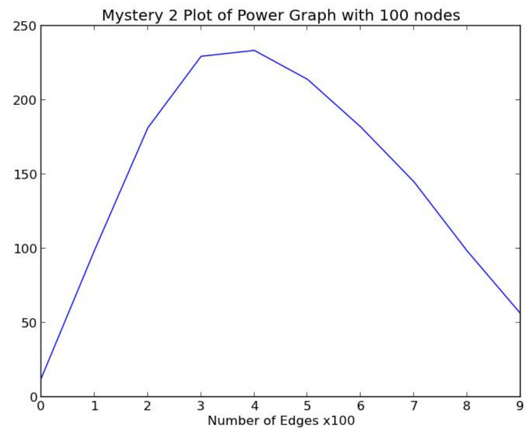
Assume a random power-graph is created using the procedures explained above with 100 nodes. The following values are computed and plotted as a function of the number of edges, according to the following code:

```
maxDegrees, meanDegrees, meanDegreeVariances, meanShortestPaths = [],[],[],[]
graph = initializeGraph(n)
for nEdges in range(n, n*n, n*n/10 ):
   graph.addEdges(nEdges)
   maxDegrees.append(graph.maxDegree())
   meanDegrees.append(graph.meanDegree())
   meanDegreeVariances.append(graph.meanDegreeVariances())
   meanShortestPaths.append(graph.meanShortestPath())
```

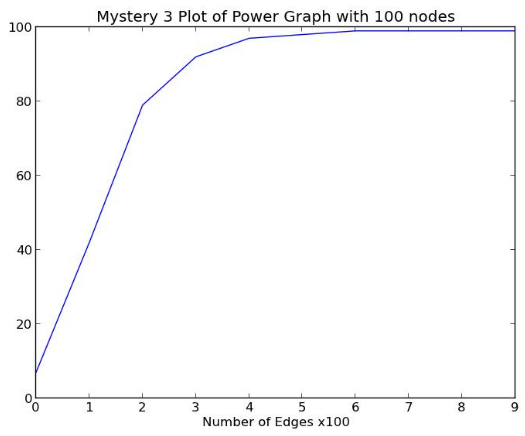
For each of the following plots, indicate which list was used to generate the plot:



- ○ `maxDegrees`
- [✓ `meanDegrees`] ✓
- ○ `meanDegreeVariances`
- ○ `meanShortestPaths`



- ○ `maxDegrees`
- ○ `meanDegrees`
- [✓ `meanDegreeVariances`] ✓
- ○ `meanShortestPaths`

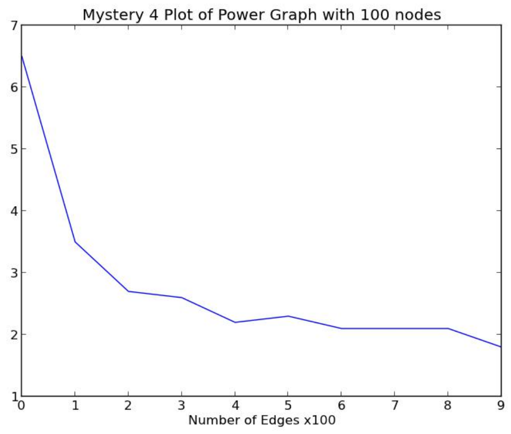


[✓ `maxDegrees`] ✓

`meanDegrees`

`meanDegreeVariances`

`meanShortestPaths`



`maxDegrees`

`meanDegrees`

`meanDegreeVariances`

[✓ `meanShortestPaths`] ✓

*You have used 2 of 2 submissions*