Lab Assignment 07

Large Scale Network Analysis with SNAP

# Assignment Overview

This assignment will introduce you to **Stanford Network Analysis Platform (SNAP)** and provide hands-on experience with analyzing large-scale networks. You will find SNAP to be a useful tool as you model maps as a graph which can then be analyzed and manipulated using SNAP.

You will use one of the Datasets in SNAP to create a graph which represents that data set and compute the following statistics:

1. Number of nodes. (What does each node represent?)
2. Number of edges. (What does each edge represent?)
3. Calculate the cluster coefficient.
4. Calculate the Diameter of the graph (longest shortest path).

## First, you will need to install Snap.py

* Requires Python 2.x
  + [Download and install Python 2.x](http://www.python.org/) (http://www.python.org)
  + [Download the snap.py for your platform](http://snap.stanford.edu/snappy) (http://snap.stanford.edu/snappy)
  + Installation:
    - Follow instruction on the Snap.py webpage
    - Python setup.py install
  + The most important step:
    - Import the snap module!

$ python

>>> import snap

# Background

SNAP is a high-performance system for analysis and manipulation of large networks, as it has over 70 network datasets ready to use and analyze.

**Snap.py Naming Conventions:**

* **Variable types/names**:
  + ...Int: an integer operation, variable: GetValInt()
  + ...Flt: a floating point operation, variable; GetValFlt()
  + ...Str: a string operation, variable; GetDateStr()
* **Classes vs. Graph Objects:** 
  + T...: a class type; TUNGraph
  + P...: type of a graph object; PUNGraph
* **Data Structures:** 
  + ...V: a vector, variable TIntV InNIdV
  + ...VV: a vector of vectors (i.e., a matrix), variable FltVV

TFltVV ... a matrix of floating point elements

* ...H: a hash table, variable NodeH

TIntStrH ... a hash table with TInt keys, TStr values

* ...HH: a hash of hashes, variable NodeHH

TIntIntHH ... a hash table with TInt key 1 and TInt key 2

* ...Pr: a pair; type TIntPr
* Get...: an access method, GetDeg()
* Set...: a set method, SetXYLabel()
* ...I: an iterator, NodeI
* Id: an identifier, GetUId()
* NId: a node identifier, GetNId()
* EId: an edge identifier, GetEId()
* Nbr: a neighbor, GetNbrNId()
* Deg: a node degree, GetOutDeg()
* Src: a source node, GetSrcNId()
* Dst: a destination node, GetDstNId()

**Basic Types in SNAP.py**

* TInt: Integer
* TFlt: Float
* TStr: String

For more information check out the [Snap.py Reference Manual](•%09http://snap.stanford.edu/snappy/doc/reference/index-ref.html) (http://snap.stanford.edu/snappy/doc/reference/index-ref.html)

# Assignment Description / Specification

## Now let’s start working with SNAP.py :

1. **Vector Example**: open your python shell and try this code out:

|  |
| --- |
| v = snap.TIntV()  v.Add(1)  v.Add(2)  v.Add(3)  v.Add(4)  v.Add(5)  print v.Len()  print v[3]  v[3] = 2\*v[2]  print v[3]  for item in v:  print item  for i in range(0, v.Len()):  print i, v[i] |

1. **Hash-table Example**: open your python shell and try this code out:

|  |
| --- |
| h = snap.TIntStrH()  h[5] = "apple"  h[3] = "tomato"  h[9] = "orange"  h[6] = "banana"  h[1] = "apricot"  print h.Len()  print "h[3] =", h[3]  h[3] = "peach"  print "h[3] =", h[3]  for key in h:  print key, h[key] |

* **Basic Graph and Network Classes**:
  + **Graphs vs. Networks Classes**
    - TUNGraph: undirected graph
    - TNGraph: directed graph
    - TNEANet: multigraph with attributes on nodes and edges
  + Object types start with **P…, since they use wrapper classes for garbage collection**
    - PUNGraph, PNGraph, PNANet
  + **Guideline**
    - For class methods (functions) use **T**
    - For object instances (variables) use **P**

1. **Graph Example**: open your python shell and try this code out:

|  |
| --- |
| G1 = snap.TNGraph.New() # directed Graph  G1.AddNode(1)  G1.AddNode(5)  G.AddNode(12) # add nodes before adding edges  G.AddEdge(5,1)  G.AddEdge(1,5)  G.AddEdge(5,12)G2=snap.TUNGraph.New()  N1=snap.TNEANet.New()  For NI in G1.Nodes(): # Node traversal  Print “node id %d, out-degree %d, in-degree %d”%(NI.GetId(),NI.GetOutDeg(),NI.GetInDeg())  for NI in G1.Nodes():  print "node id %d, out-degree %d, in-degree %d"  % (NI.GetId(), NI.GetOutDeg(), NI.GetInDeg())  for EI in G1.Edges():  print "(%d, %d)" % (EI.GetSrcNId(), EI.GetDstNId())  for NI in G1.Nodes(): for DstNId in NI.GetOutEdges():  print "(%d %d)" % (NI.GetId(), DstNId) |

D. **Drawing Graphs with Snap.py**

* In order to visualize the graphs, you have to install [GraphViz](http://www.graphviz.org/) (http://www.graphviz.org/)
* Make sure that the directory containing GraphViz is in your environmental variable   
  $ PATH

|  |
| --- |
| G1 = snap.TNGraph.New() # directed Graph  G1.AddNode(1)  G1.AddNode(5)  G.AddNode(12) # add nodes before adding edges  G.AddEdge(5,1)  G.AddEdge(1,5)  G.AddEdge(5,12)G2=snap.TUNGraph.New()  N1=snap.TNEANet.New()  For NI in G1.Nodes(): # Node traversal  Print “node id %d, out-degree %d, in-degree %d”%(NI.GetId(),NI.GetOutDeg(),NI.GetInDeg())  for NI in G1.Nodes():  print "node id %d, out-degree %d, in-degree %d"  % (NI.GetId(), NI.GetOutDeg(), NI.GetInDeg())  for EI in G1.Edges():  print "(%d, %d)" % (EI.GetSrcNId(), EI.GetDstNId())  for NI in G1.Nodes(): for DstNId in NI.GetOutEdges():  print "(%d %d)" % (NI.GetId(), DstNId)  ##Draw Macintosh HD:Users:mahaallouzi:Desktop:Screen Shot 2016-10-31 at 3.24.23 PM.png  snap.DrawGViz(G1, snap.gvlDot, "G1.png", "G1", NIdName) |

E. **An overview of network analytics**

* You can get a network using more than one way such as the following:
  + From a real-world dataset
  + Generate a synthetic network
  + From an existing network
* Calculate network properties
  + Quick summary of network properties
  + Global connectivity: connected components
  + Local nodes in the network: node centrality
  + Neighborhood connectivity: triads, clustering coefficient
  + Graph traversal: breadth and depth first search
  + Group of nodes: community detection
  + Global graph properties: spectral graph analysis
  + Core nodes: K-core decomposition
* Basic Graph Generators
  + Complete, circle, grid, star, tree graphs

**Example**: open your python shell and try this code out:

|  |
| --- |
| GG=snap.GenGrid(snap.PUNGraph,4,3)  GG=snap.Gentree(snap.PUNGraph,4,2)  Macintosh HD:Users:mahaallouzi:Desktop:Screen Shot 2016-10-31 at 4.15.39 PM.pngMacintosh HD:Users:mahaallouzi:Desktop:Screen Shot 2016-10-31 at 4.18.01 PM.png |

# Exercise

Use one of the Datasets in SNAP to create a graph which represents that data set and compute the following statistics:

1. Number of nodes. (What does each node represent?)
2. Number of edges. (What does each edge represent?)
3. Calculate the cluster coefficient.
4. Calculate the Diameter of the graph (longest shortest path).

# Deliverables

1. Submit your lab07.txt in Blackboard.