

```

# Tony Hyun Kim
# CS 224w, PS 4, Problem 2
#-----
from __future__ import division
import snap

# Generate the random graph according to Barabasi-Albert PA model
#-----
n = 1000
G = snap.GenPrefAttach(n, 4, snap.TRnd()) # Undirected graph

# I will use NG to store the betweenness centrality
# Note that the PNEANet type stores directed edges. Since the
# actual graph of interest (G) is undirected, we will access
# data in NG where the nodes of an edge are in increasing order
NG = snap.ConvertGraph(snap.PNEANet, G)
NG.AddFltAttrE("betweenCent", 0.0)

# Compute the exact betweenness centrality
#-----
for s in G.Nodes():
    sid = s.GetId()
    #print "sid: {}".format(sid)

    # Obtain the BFS tree as a network, so that we can use attributes
    Ts = snap.ConvertGraph(snap.PNEANet,
                          snap.GetBfsTree(G, sid, True, False))
    Ts.AddIntAttrN("sigma", 0) # Number of shortest paths
    Ts.AddFltAttrE("delta", 0.0) # Dependency

    # Girvan-Newman algorithm requires us to walk down BFS tree for sigma
    # climb back up for delta
    fwd_order = [n.GetId() for n in Ts.Nodes()]
    fwd_order.pop(0) # Don't need to go over the source node
    rev_order = fwd_order[::-1]

    # Debug: Enumerate the tree structure
    '''
    for vid in fwd_order:
        v = Ts.GetNI(vid)
        Nout = [nid for nid in v.GetOutEdges()]
        Nin = [nid for nid in v.GetInEdges()]
        print "vid: {}, {}, {}".format(vid, Nout, Nin)
    '''

    # Perform downward BFS pass on sigma
    Ts.AddIntAttrDatN(sid, 1, "sigma")
    for vid in fwd_order:
        v = Ts.GetNI(vid)
        sigma = 0
        for parent_id in v.GetInEdges():
            sigma += Ts.GetIntAttrDatN(parent_id, "sigma")
        Ts.AddIntAttrDatN(vid, sigma, "sigma")
        #print "vid={}, sig_sv={}".format(vid, sigma)

    # Perform upward BFS pass on delta
    for wid in rev_order:
        w = Ts.GetNI(wid)

        multiplier = 1
        for child_id in w.GetOutEdges():
            multiplier += Ts.GetFltAttrDatE(Ts.GetEId(wid, child_id), "delta")

        for vid in w.GetInEdges():
            delta = Ts.GetIntAttrDatN(vid, "sigma")/Ts.GetIntAttrDatN(wid, "sigma")
            delta *= multiplier
            Ts.AddFltAttrDatE(Ts.GetEId(vid, wid), delta, "delta")

    '''
    for ei in Ts.Edges():
        print "edge ({}), {}, delta={}".format(
            ei.GetSrcNId(), ei.GetDstNId(),
            Ts.GetFltAttrDatE(ei.GetId(), "delta"))

```

```
    ...

# Accumulate deltas
for ei in Ts.Edges():
    e = sorted([ei.GetSrcNId(), ei.GetDstNId()])
    e_bc = NG.GetEId(e[0], e[1])
    bc = NG.GetFltAttrDatE(e_bc, "betweenCent")
    bc += Ts.GetFltAttrDatE(ei.GetId(), "delta")
    NG.AddFltAttrDatE(e_bc, bc, "betweenCent")

# Display the final betweenness centrality
for ei in G.Edges():
    e = sorted([ei.GetSrcNId(), ei.GetDstNId()])
    e_bc = NG.GetEId(e[0], e[1])
    bc = NG.GetFltAttrDatE(e_bc, "betweenCent")
    print "{} {} {:.4f}".format(e[0], e[1], bc)
```