```
# Tony Hyun Kim
# CS 224w, PS 4, Problem 4.7
# Implementation of HighestDegree
#-------------------------------------------------------------
from __future__ import division
import snap
import sys

# Some support functions
#-------------------------------------------------------------
def GetGraphCopy(G):
    G2 = snap.TUNGraph.New()
    for n in G.Nodes():
        G2.AddNode(n.GetId())
    for e in G.Edges():
        G2.AddEdge(e.GetSrcNId(), e.GetDstNId())
    return G2

def Ints2TIntV(vals):
    ret = snap.TIntV()
    for val in vals:
        ret.Add(val)
    return ret

def GetAnchoredKCore(G,A,K):
    '''
    Compute the anchored K-core by my algorithm
        given in Prob 4.3
    '''
    V0 = set()
    for n in G.Nodes():
        V0.add(n.GetId())

    while True:
        G1 = snap.GetSubGraph(G, Ints2TIntV(V0))

        V1 = set()
        for anchor in A:
            V1.add(anchor) # Always preserve the anchors

        for n in G1.Nodes():
            if (n.GetOutDeg() >= K):
                V1.add(n.GetId())

        if (V1.issubset(V0) & V0.issubset(V1)): # Set equivalence
            break

        V0 = V1

    return V1

# Load the graph
#-------------------------------------------------------------
#source = "toy3.txt"
source = sys.argv[1]
G = snap.LoadEdgeList(snap.PUNGraph, source, 0, 1)


# Maximum budget to consider
b_max = int(sys.argv[2])
b     = b_max

K = 2
A = set()            # Set of anchored nodes
Gs = GetGraphCopy(G) # "Scratch" graph

V1 = GetAnchoredKCore(G,A,K) # V1 is the equilibrium set
print "{}, {}".format(0, len(V1))

while b > 0:
    # Grab the node with the highest degree, but don't add
    #   nodes already in the equilibrium set, since that
    #   would be basically throwing away the budget
```

```
    n_id = snap.GetMxOutDegNId(Gs)
    if (not (n_id in V1)):
        A.add(n_id)
        b = b-1

        V1 = GetAnchoredKCore(G,A,K)
        #print "A={}".format(A)
        print "{}, {}".format(b_max-b, len(V1))

    Gs.DelNode(n_id)
```