Homework 3: Question 4

In [1]:
```
from snap import *
from random import sample,choice
from ggplot import *
```

In [2]:
```
N=10670
M=22002
nodes=arange(N)
dia_sample = 20
```

Creating graphs

Create a random Gnm network

In [3]:
```
g_nm = PUNGraph_New()
for i in nodes: g_nm.AddNode(i)
while True:
    s,t = sample(nodes,2)
    g_nm.AddEdge(s,t)
    if g_nm.GetEdges() == M: break
```

In [4]:
```
g_nm.GetNodes(),g_nm.GetEdges()
```

Out[4]:
```
(10670, 22002)
```

Save graph

In [5]:
```
SaveEdgeList_PUNGraph(g_nm,"Gnm.txt")
```

Create a graph G_pa with preferential attachment

Start with a complete graph of 40 nodes

In [6]:
```
N_init = 40
edges = []
```

In [7]:
```
g_pa = PUNGraph_New()
```

In [8]:
```
for n in xrange(N_init):
    g_pa.AddNode(n)
    for m in xrange(n):
        g_pa.AddEdge(m,n)
        edges.append((m,n))
```

In [9]:
```
for n in nodes[N_init:]:
    g_pa.AddNode(n)

    for i in xrange(2):
        m = choice(choice(edges))
        g_pa.AddEdge(m,n)
        edges.append((m,n))

    if g_pa.GetEdges() == M: break
```

In [10]:
```
g_pa.GetNodes(),g_pa.GetEdges()
```

Out[10]: (10670, 21992)

In [11]:
```
SaveEdgeList_PUNGraph(g_pa,"Gpa.txt")
```

Load Autonomous network graph

In [12]:
```
g_as = LoadEdgeList_PUNGraph("oregon1_010331.txt")
SaveEdgeList_PUNGraph(g_as,"Gas.txt")
```

Q4.1) Deletion experiments for failure vs attack

Failure deletion

In [13]:
```
def failure1(graph,batchsize,percent):
    del_nodes = 0 # number of deleted nodes
    N = graph.GetNodes()
    stopN = (percent*N)/100 # number of nodes at which to stop
    X = [0]
    Y = [GetBfsEffDiam_PUNGraph(graph,dia_sample)]
    nodeset = set(range(N))
    while True:   # start deleting
        for d in sample(nodeset,batchsize):
            graph.DelNode(d)
            nodeset.remove(d)
        del_nodes += batchsize
        dia = GetBfsEffDiam_PUNGraph(graph,dia_sample)
        X.append((100.0*del_nodes)/N)
        Y.append(dia)
        if del_nodes >= stopN: break

    return X,Y
```

Attack deletion

In [14]:
```
def attack1(graph,batchsize,percent):
    del_nodes = 0 # number of deleted nodes
    N = graph.GetNodes()
    stopN = (percent*N)/100 # number of nodes at which to stop
    X = [0]
    Y = [GetBfsEffDiam_PUNGraph(graph,dia_sample)]
    nodeset = set(range(N))
    while True:   # start deleting
        for i in xrange(batchsize):
            d = GetMxDegNId_PUNGraph(graph)
            graph.DelNode(d)
            nodeset.remove(d)
        del_nodes += batchsize
        dia = GetBfsEffDiam_PUNGraph(graph,dia_sample)
        X.append((100.0*del_nodes)/N)
        Y.append(dia)
        if del_nodes >= stopN: break

    return X,Y
```

Plot for average diameter vs. deleted nodes

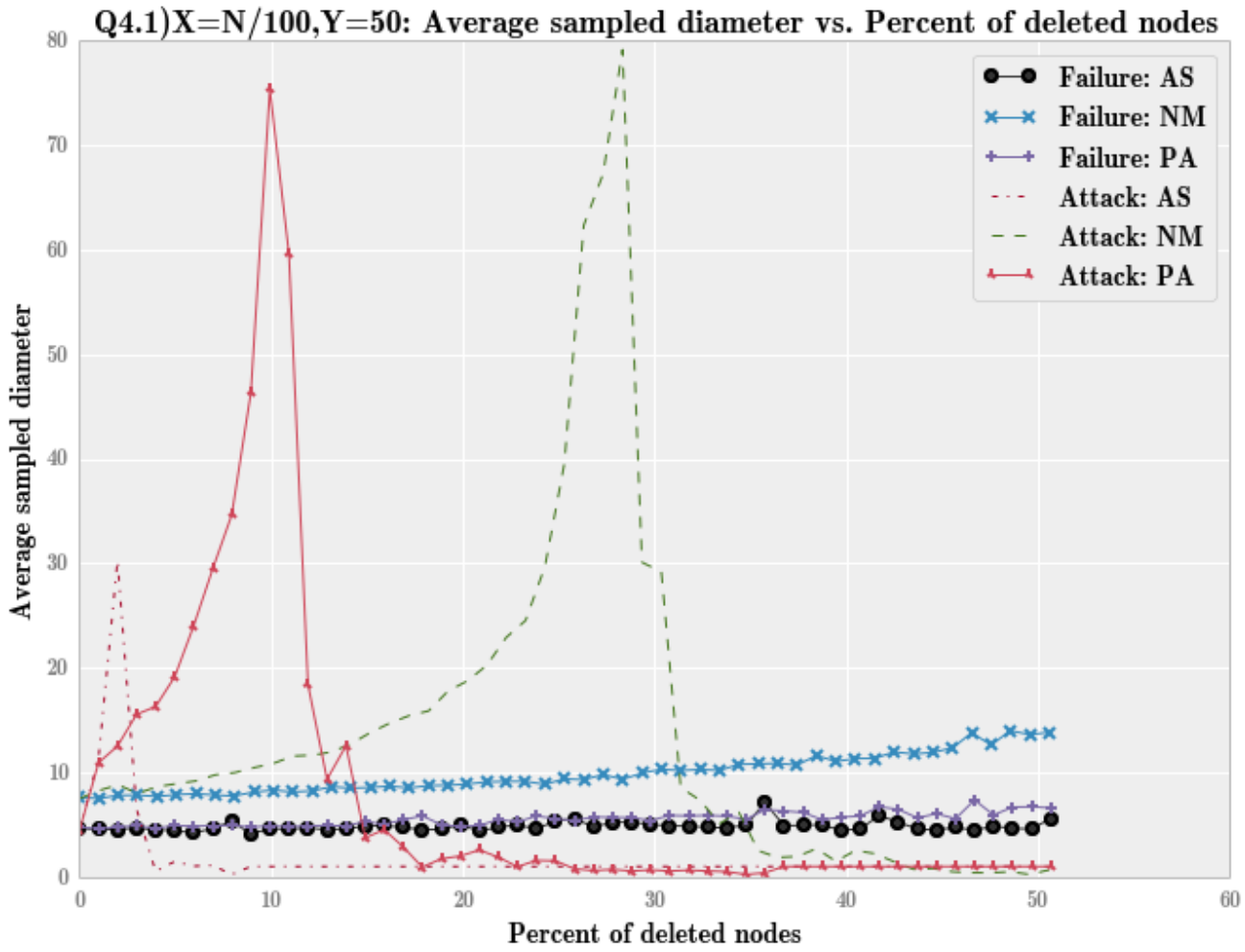
In [15]:
```
def plots(X,Y,xlab,ylab,tpref,failure_func,attack_func):
    g_nm = LoadEdgeListStr_PUNGraph("Gnm.txt")
    f_g_nm_x,f_g_nm_y = failure_func(g_nm,X,Y)
    g_as = LoadEdgeListStr_PUNGraph("Gas.txt")
    f_g_as_x,f_g_as_y = failure_func(g_as,X,Y)
    g_pa = LoadEdgeListStr_PUNGraph("Gpa.txt")
    f_g_pa_x,f_g_pa_y = failure_func(g_pa,X,Y)
    g_nm = LoadEdgeListStr_PUNGraph("Gnm.txt")
    a_g_nm_x,a_g_nm_y = attack_func(g_nm,X,Y)
    g_as = LoadEdgeListStr_PUNGraph("Gas.txt")
    a_g_as_x,a_g_as_y = attack_func(g_as,X,Y)
    g_pa = LoadEdgeListStr_PUNGraph("Gpa.txt")
    a_g_pa_x,a_g_pa_y = attack_func(g_pa,X,Y)
    p = plt.plot(f_g_as_x,f_g_as_y,'-o',f_g_nm_x,f_g_nm_y,'-x',f_g_pa_x,f_g_pa
_y,'-+',
                 a_g_as_x,a_g_as_y,'-.',a_g_nm_x,a_g_nm_y,'--',a_g_pa_x,a_g_pa
_y,'-4',
                 lw=1,mew=2)
    p = plt.legend(("Failure: AS","Failure: NM","Failure: PA",
                "Attack: AS","Attack: NM","Attack: PA"),loc="best")
    p = plt.title(tpref + ': ' + ylab + " vs. " + xlab)
    p = plt.xlabel(xlab)
    p = plt.ylabel(ylab)
```

Scenario 1: X = N/100, Y = 50

In [16]:
```
X = N/100
Y = 50
plots(X,Y,"Percent of deleted nodes","Average sampled diameter","Q4.1)X=N/100,
Y=50",
      failure1,attack1)
```

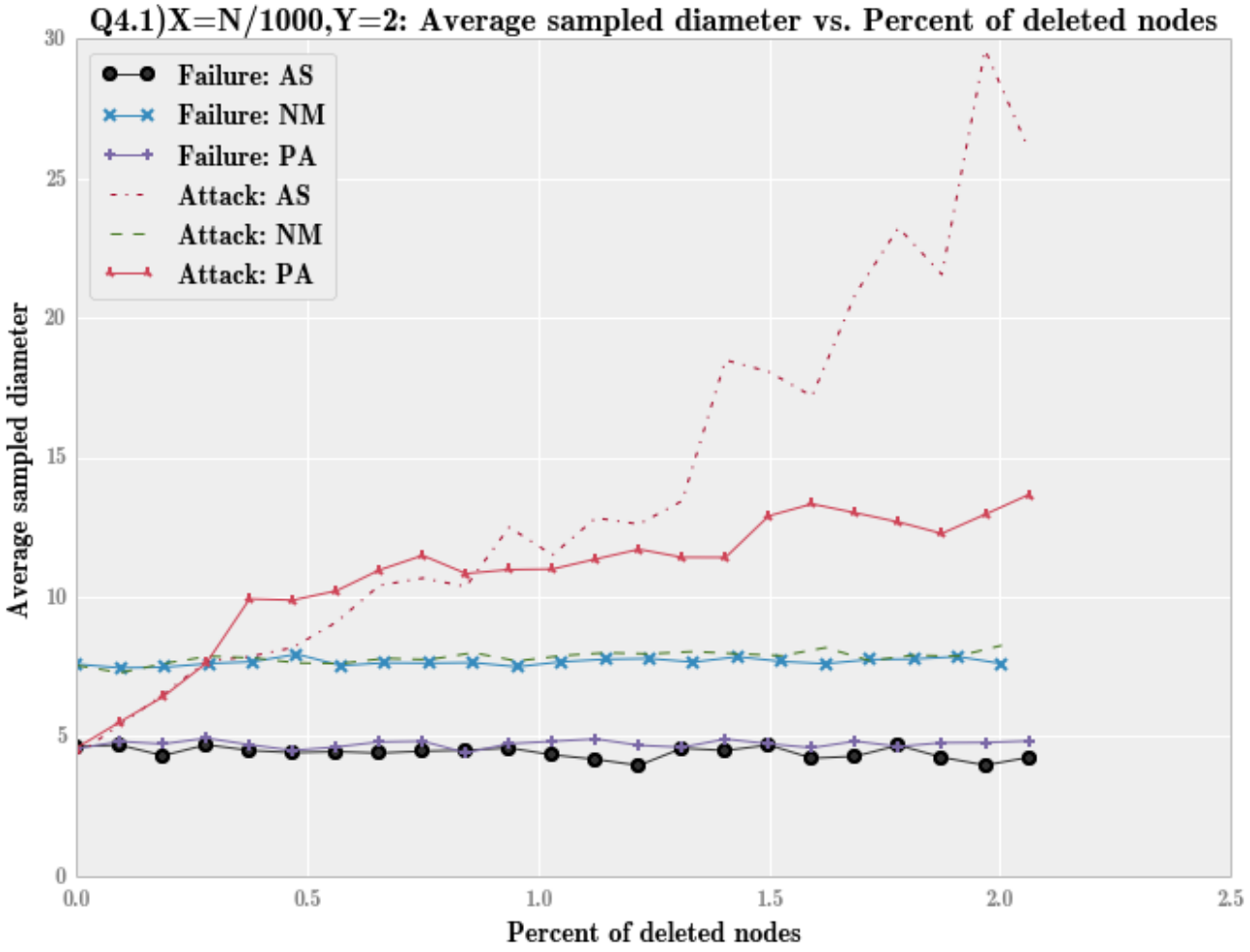
```
/usr/local/lib/python2.7/dist-packages/matplotlib/font_manager.py:1236: UserWa
rning: findfont: Font family ['serif'] not found. Falling back to Bitstream Ve
ra Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```

```
/usr/local/lib/python2.7/dist-packages/matplotlib/font_manager.py:1246: UserWa
rning: findfont: Could not match :family=Bitstream Vera Sans:style=normal:vari
ant=normal:weight=normal:stretch=normal:size=medium. Returning /usr/share/matp
lotlib/mpl-data/fonts/ttf/cmb10.ttf
  UserWarning)
/usr/local/lib/python2.7/dist-packages/matplotlib/font_manager.py:1246: UserWa
rning: findfont: Could not match :family=Bitstream Vera Sans:style=normal:vari
ant=normal:weight=normal:stretch=normal:size=large. Returning /usr/share/matpl
otlib/mpl-data/fonts/ttf/cmb10.ttf
  UserWarning)
/usr/local/lib/python2.7/dist-packages/matplotlib/font_manager.py:1246: UserWa
rning: findfont: Could not match :family=Bitstream Vera Sans:style=normal:vari
ant=normal:weight=normal:stretch=normal:size=x-large. Returning /usr/share/mat
plotlib/mpl-data/fonts/ttf/cmb10.ttf
  UserWarning)
```



Scenario 2: X = N/1000, Y = 2

In [17]:
```
X = N/1000
Y = 2
plots(X,Y,"Percent of deleted nodes","Average sampled diameter","Q4.1)X=N/1000
,Y=2",
      failure1,attack1)
```



Q4.2) Change in size of largest connected component

Failure deletion

In [18]:
```
def failure2(graph,batchsize,percent):
    del_nodes = 0 # number of deleted nodes
    N = graph.GetNodes()
    stopN = (percent*N)/100 # number of nodes at which to stop
    X = [0]
    Y = [float(GetMxWccSz_PUNGraph(graph))]
    nodeset = set(range(N))
    while True:   # start deleting
        for d in sample(nodeset,batchsize):
            graph.DelNode(d)
            nodeset.remove(d)
        del_nodes += batchsize
        lcc = float(GetMxWccSz_PUNGraph(graph)) # size of LCC
        X.append((100.0*del_nodes)/N)
        Y.append(lcc)
        if del_nodes >= stopN: break

    return X,Y
```

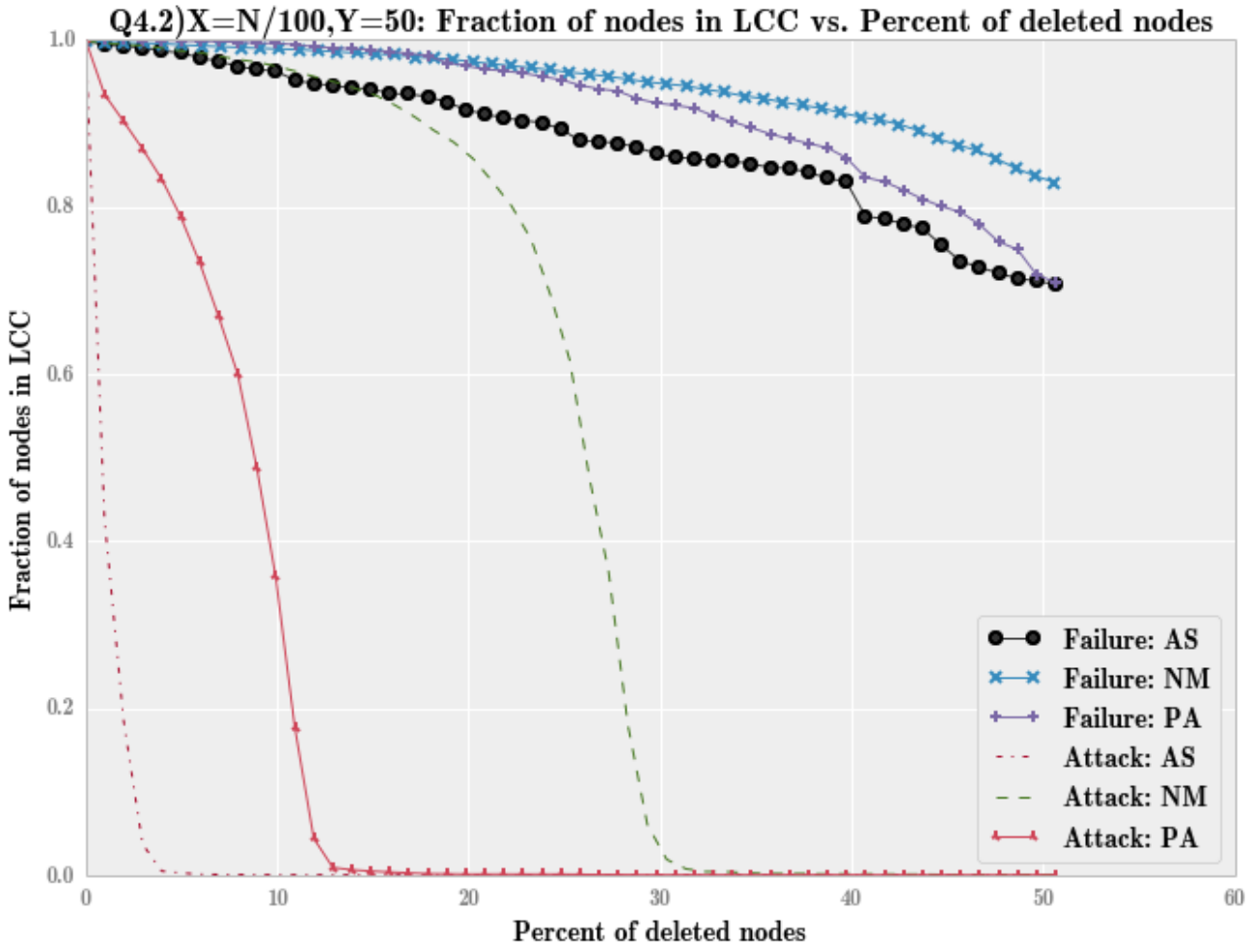
Attack deletion

In [19]:
```
def attack2(graph,batchsize,percent):
    del_nodes = 0 # number of deleted nodes
    N = graph.GetNodes()
    stopN = (percent*N)/100 # number of nodes at which to stop
    X = [0]
    Y = [float(GetMxWccSz_PUNGraph(graph))]
    nodeset = set(range(N))
    while True:   # start deleting
        for i in xrange(batchsize):
            d = GetMxDegNId_PUNGraph(graph)
            graph.DelNode(d)
            nodeset.remove(d)
        del_nodes += batchsize
        lcc = float(GetMxWccSz_PUNGraph(graph))
        X.append((100.0*del_nodes)/N)
        Y.append(lcc)
        if del_nodes >= stopN: break

    return X,Y
```

Plots of fraction in largest connected component vs. percent deleted nodes

In [20]:
```
X = N/100
Y = 50
plots(X,Y,"Percent of deleted nodes","Fraction of nodes in LCC","Q4.2)X=N/100,
Y=50",
      failure2,attack2)
```



In [20]:

In [20]:

In [20]:

In [20]:

In [ ]: