Lec80: Link Analysis - Implementing Page Rank using Points Distribution Method - 1

- Create/take a directed graph with 'n' nodes.
- Assign 100 points to each node.
- Keep distributing points until convergence.
- Get nodes' raking as per the points accumulated
- Compare the ranks thus obtained with the ranks obtainrd from the inbuilt Page rank method

Lec81: Link Analysis - Implementing Page Rank using Points Distribution Method - 2

In [7]:

```
import networkx as nx
import random

def add_edges(G, p):
    #Randomly adding edges
    for i in G.nodes():
        for j in G.nodes():
            if i != j:
                r = random.random()
                if r <= p:
                    G.add_edge(i, j)
                else:
                    continue
    return G

def initialize_points(G):
    points = [100 for i in range(G.number_of_nodes())]
    return points

def distribute_points(G, points):
    prev_points = points
    new_points = [0 for i in range(G.number_of_nodes())]

    for i in G.nodes():
        out = G.out_edges(i)
        if len(out) == 0:
            new_points[i] += prev_points[i]
        else:
            share = float(prev_points[i]) / len(out)
            for each in out:
                new_points[each[1]] += share

    return G, new_points

def keep_distibuting_points(G, points):
    prev_points = points
    print 'Enter # to stop'
    while(1):
        G, new_points = distribute_points(G, prev_points)
        print new_points

        char = raw_input()
        if char == '#':
            break
        prev_points = new_points
    return G, new_points


def main():
    # Create/take a directed graph with 'n' nodes.
    G = nx.DiGraph()#empty graph
    G.add_nodes_from([i for i in range(10)])
    G = add_edges(G, 0.3)

    # Assign 100 points to each node.
    points = initialize_points(G)

    # Keep distributing points until convergence.
    G, points = keep_distibuting_points(G, points)
    # Get nodes' raking as per the points accumulated
    # Compare the ranks thus obtained with the ranks obtainrd from the inbuilt Page

main()
```

```
Enter # to stop
[86.66666666666667, 53.333333333333336, 50.0, 150.0, 83.3333333333333
4, 70.0, 233.33333333333334, 53.333333333333336, 75.0, 145.0]

[81.66666666666667, 43.888888888888886, 71.66666666666667, 140.5555555
5555554, 121.66666666666667, 87.5, 218.05555555555554, 38.333333333333
33, 85.0, 111.66666666666667]

[87.74074074074073, 44.22222222222223, 65.48611111111111, 145.76388888
888889, 112.33796296296296, 72.83333333333334, 202.12962962962965, 46.
16666666666667, 73.68055555555554, 149.63888888888889]

[85.15277777777779, 43.98302469135802, 61.58796296296297, 136.94135802
469137, 110.17592592592592, 89.55555555555556, 221.84413580246917, 39.
01388888888889, 73.61574074074075, 138.12962962962962]

[80.89958847736627, 43.107407407407415, 66.4567901234568, 145.22222222
222223, 112.10390946502059, 83.78796296296296, 212.9843106995885, 44.5
75, 74.96797839506173, 135.89483024691359]

[85.55326646090535, 41.960125171467766, 64.02292952674898, 141.0710433
813443, 112.43033693415639, 82.94101080246914, 215.69043638545958, 42.
922916666666666, 75.53357767489712, 137.8743569958848]

[83.47137317101053, 43.624471021947876, 64.41264038923184, 141.9183759
8593966, 111.43632151634661, 84.04389403292183, 214.92805355509833, 4
2.75371913580247, 75.38406742969823, 138.02708376200275]

[83.85381894433016, 42.90060454294316, 64.63813114426155, 141.94743367
53163, 111.94425647290811, 84.09035536694103, 215.00582930860386, 43.0
9144483024692, 75.10887295667581, 137.41925275777322]

[83.88362953119444, 42.97304757277855, 64.47660846288676, 142.00404361
4731, 111.79241968799218, 83.73140097022177, 215.09859517095973, 43.05
189304698217, 75.29717974227442, 137.69118219997907]

[83.88631997432748, 43.020645792186365, 64.51791068593457, 141.8817903
4070222, 111.8525918908449, 83.90502704844442, 215.07009848315863, 42.
96990293852881, 75.30059531623101, 137.5951175296417]

[83.86001940545846, 43.02222572135536, 64.52268606883625, 141.95910530
507174, 111.81661618240366, 83.85767782806705, 215.01285958388817, 43.
02846141272768, 75.25247609005406, 137.6678724021377]

[83.87775900931348, 43.0038350198303, 64.50877132631088, 141.922232427
09815, 111.82847309466814, 83.88443141907966, 215.0813297765772, 43.00
3054494033165, 75.26744560233588, 137.62266783075333]

[83.86382370493685, 43.01274212357167, 64.52129119910188, 141.94494511
733654, 111.8287020081346, 83.86482303584384, 215.0439258896606, 43.01
496626016039, 75.27185969116088, 137.6329209700929]

[83.87645071037832, 43.008979839877796, 64.5141670033807, 141.9308129
8326894, 111.82914870908692, 83.87083242327863, 215.05557130086635, 4
3.009312950180124, 75.26846460249536, 137.6362594772597]
```

#

Lec82: Link Analysis - Implementing Page Rank using Points Distribution Method - 3

In [13]:

```
import networkx as nx
import random
import numpy as np

def add_edges(G, p):
    #Randomly adding edges
    for i in G.nodes():
        for j in G.nodes():
            if i != j:
                r = random.random()
                if r <= p:
                    G.add_edge(i, j)
                else:
                    continue
    return G

def initialize_points(G):
    points = [100 for i in range(G.number_of_nodes())]
    return points

def distribute_points(G, points):
    prev_points = points
    new_points = [0 for i in range(G.number_of_nodes())]

    for i in G.nodes():
        out = G.out_edges(i)
        if len(out) == 0:
            new_points[i] += prev_points[i]
        else:
            share = float(prev_points[i]) / len(out)
            for each in out:
                new_points[each[1]] += share

    return G, new_points

def keep_distibuting_points(G, points):
    prev_points = points
    print 'Enter # to stop'
    while(1):
        G, new_points = distribute_points(G, prev_points)
        print new_points

        char = raw_input()
        if char == '#':
            break
        prev_points = new_points
    return G, new_points

def get_nodes_sorted_by_points(points):
    points_array = np.array(points)
    #returns the indices of the list sorted
    nodes_sorted_by_points = np.argsort(-points_array)# - for desc order
    return nodes_sorted_by_points

def main():
    # Create/take a directed graph with 'n' nodes.
    G = nx.DiGraph()#empty graph
    G.add_nodes_from([i for i in range(10)])
    G = add_edges(G, 0.3)
    # Assign 100 points to each node.
    points = initialize_points(G)
    print points

    # Keep distributing points until convergence.
    G, points = keep_distibuting_points(G, points)

    # Get nodes' raking as per the points accumulated
    nodes_sorted_by_points = get_nodes_sorted_by_points(points)
    print 'nodes_sorted_by_points : ', nodes_sorted_by_points

    # Compare the ranks thus obtained with the ranks obtainrd from the inbuilt Page
    pr = nx.pagerank(G)# Return a dictionary
    pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
    print 'Page Rank by Inbuilt Method'
    for i in pr_sorted:
        print i[0],

main()
```

```
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[0, 120.0, 78.33333333333334, 66.66666666666667, 270.0, 136.66666666
666669, 66.66666666666667, 86.66666666666667, 125.0, 50.0]

[0, 182.7777777777778, 55.0, 62.22222222222223, 193.88888888888889,
 166.66666666666666, 135.55555555555557, 63.8888888888889, 88.333333
33333334, 51.66666666666667]

[0, 158.59259259259258, 63.361111111111114, 72.77777777777777, 172.2
962962962963, 143.9629629629296, 120.18518518518519, 55.62962962962
963, 151.5277777777778, 61.66666666666667]

[0, 159.98456790123458, 69.95370370370371, 68.54320987654322, 192.03
703703703707, 141.29320987654322, 105.41975308641975, 75.12037037037
038, 134.0925925925926, 53.55555555555556]

[0, 160.77633744855967, 66.81466049382718, 64.94958847736626, 196.24
804526748974, 149.6070987654321, 111.11008230452677, 67.518106995884
78, 124.19984567901234, 58.77623456790124]
```

- Difference in Ranking may arise due to sinking (nodes with no out edges) or nodes having equal weights

Lec83: Link Analysis - Implementing Page Rank using Points Distribution Method - 4

Handling nodes without out edges

In [15]:

```
import networkx as nx
import random
import numpy as np

def add_edges(G, p):
    #Randomly adding edges
    for i in G.nodes():
        for j in G.nodes():
            if i != j:
                r = random.random()
                if r <= p:
                    G.add_edge(i, j)
                else:
                    continue
    return G

def initialize_points(G):
    points = [100 for i in range(G.number_of_nodes())]
    return points

def distribute_points(G, points):
    prev_points = points
    new_points = [0 for i in range(G.number_of_nodes())]

    for i in G.nodes():
        out = G.out_edges(i)
        if len(out) == 0:
            new_points[i] += prev_points[i]
        else:
            share = float(prev_points[i]) / len(out)
            for each in out:
                new_points[each[1]] += share

    return G, new_points

##
def handle_points_sink(G, points):
    for i in range(len(points)):
        points[i] = float(points[i])*0.8

    n = G.number_of_nodes()
    tax = float(n*100*0.2)/n
    for i in range(len(points)):
        points[i] += tax
    return points

def keep_distibuting_points(G, points):
    prev_points = points
    print 'Enter # to stop'
    while(1):
        G, new_points = distribute_points(G, prev_points)
        print new_points
        ##
        new_points = handle_points_sink(G, new_points)

        char = raw_input()
        if char == '#':
            break
        prev_points = new_points
    return G, new_points


def get_nodes_sorted_by_points(points):
    points_array = np.array(points)
    #returns the indices of the list sorted
    nodes_sorted_by_points = np.argsort(-points_array)# - for desc order
    return nodes_sorted_by_points

def main():
    # Create/take a directed graph with 'n' nodes.
    G = nx.DiGraph()#empty graph
    G.add_nodes_from([i for i in range(10)])
    G = add_edges(G, 0.3)

    # Assign 100 points to each node.
    points = initialize_points(G)
    print points

    # Keep distributing points until convergence.
    G, points = keep_distibuting_points(G, points)

    # Get nodes' raking as per the points accumulated
    nodes_sorted_by_points = get_nodes_sorted_by_points(points)
    print 'nodes_sorted_by_points : ', nodes_sorted_by_points

    # Compare the ranks thus obtained with the ranks obtainrd from the inbuilt Page
    pr = nx.pagerank(G)# Return a dictionary
    pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
    print 'Page Rank by Inbuilt Method'
    for i in pr_sorted:
        print i[0],

main()
```

```
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[166.66666666666669, 50.0, 225.0, 91.66666666666667, 75.0, 75.0, 50.
0, 158.33333333333334, 83.33333333333334, 25.0]

[133.33333333333334, 20.0, 241.66666666666669, 96.66666666666667, 12
6.66666666666667, 91.66666666666667, 65.0, 135.0, 40.0, 50.0]

[116.44444444444446, 30.0, 230.33333333333334, 117.7777777777778, 11
6.66666666666669, 72.33333333333334, 62.33333333333334, 146.77777777
77778, 54.0, 53.33333333333334]

[124.26666666666668, 31.33333333333334, 226.28888888888892, 112.1333
3333333335, 107.64444444444445, 67.57777777777778, 62.0666666666666
7, 163.00000000000003, 54.62222222222234, 51.06666666666667]

[122.28740740740741, 30.42666666666667, 235.72888888888895, 108.8474
0740740743, 109.96444444444447, 70.97333333333334, 61.52444444444445
5, 156.34518518518522, 53.644444444444446, 50.25777777777779]
```