In a real-world scenario a web surfer searches a page over the internet in a million of pages and search engines needs to analyse the given input and identify the all-matching web pages for the user input. In order to ensure search engine recommending the right page The algo should measures the importance of each page. Page Ranking is a metric which we used as a importance metric while filtering the content and recommend any page.

**Implementation**

**Page Rank**: real number between 0 to 1.

**Damping factor:** The random surfer on the internet will always end up with the *dangling node* (having no edges) and given any point in time the dampling factor carried is ‘d’. The probability of surfer choosing the random next link is ‘d’ and We assumed d=0.85, 0.15% shared in each iteration and total PR values of all n nodes is 1. This will prevent the convergence of value 0 or 1.

**Generating the test data with ‘DorogovtsevMendesGenerator’**:

Dorogovtsev – Mendes algorithm used to generate the test graph with random links of size 1000 nodes. The algorithm ensures the generate node with random directed edges connected to different nodes to create a mock data of real-world scenario. We have used ‘graph stream’ library for generating the test data.

System.***out***.println("Generating the test graph using 'DorogovtsevMendesGenerator'.");

DorogovtsevMendesGenerator generator = **new** DorogovtsevMendesGenerator();

generator.setDirectedEdges(**true**, **true**);

ElemenetSinkImpl sinkSource = **new** ElemenetSinkImpl(graph);

graph.addElementSink(sinkSource);

generator.addSink(graph);

generator.begin();

**while** (graph.getNodeCount() < 1000) {

generator.nextEvents();

}

generator.end();

**generate test graph output:**

**Generating the test graph using 'DorogovtsevMendesGenerator'.**

**GENERATED GRAPH**

**-----------------**

node-0,adjacent nodes->[0-147[147->0], 0-843[843->0], 0-31[0->31], 0-142[142->0], 0-480[480->0], 0-265[0->265], 0-311[0->311], 0-123[0->123], 0-7[0->7], 0-337[337->0], 0-1[0->1], 0-366[366->0], 0-282[0->282], 0-567[0->567], 0-57[57->0], 0-131[131->0], 0-5[5->0], 0-663[663->0], 0-430[0->430], 0-172[172->0], 0-583[0->583], 0-239[0->239], 0-631[631->0], 0-12[12->0], 0-961[961->0], 0-52[0->52], 0-69[69->0], 0-457[457->0], 0-151[151->0], 0-470[470->0], 0-95[95->0], 0-25[0->25], 0-930[930->0], 0-656[656->0], 0-71[0->71], 0-236[0->236], 2-0[0->2], 0-413[0->413], 0-770[770->0], 0-15[15->0], 0-8[8->0], 0-80[0->80], 0-3[0->3], 0-94[0->94]]

node-1,adjacent nodes->[1-131[1->131], 1-552[552->1], 1-602[602->1], 1-226[226->1], 1-41[41->1], 1-70[70->1], 1-405[405->1], 1-9[9->1], 1-675[1->675], 1-84[84->1], 1-27[27->1], 1-509[509->1], 1-30[1->30], 1-291[1->291], 1-48[48->1], 1-143[143->1], 1-635[635->1], 1-422[422->1], 1-14[1->14], 1-335[1->335], 1-179[179->1], 1-252[252->1], 1-315[315->1], 1-58[58->1], 1-2[1->2], 1-368[1->368], 1-71[1->71], 1-81[81->1], 1-19[19->1], 1-78[78->1], 1-167[167->1], 1-107[107->1], 1-174[1->174], 1-330[1->330], 0-1[0->1], 1-116[1->116], 1-39[1->39], 1-16[16->1], 1-7[1->7], 1-346[346->1], 1-436[1->436], 1-887[887->1], 1-6[6->1], 1-64[64->1], 1-59[1->59], 1-223[223->1], 1-11[11->1], 1-536[1->536], 1-209[1->209], 1-441[441->1], 1-954[954->1], 1-285[1->285], 1-980[1->980], 1-133[133->1], 1-587[1->587], 1-949[949->1], 1-180[1->180], 1-640[1->640], 1-5[5->1], 1-473[473->1], 1-898[898->1], 1-941[1->941], 1-791[791->1], 1-32[1->32], 1-67[67->1], 1-487[487->1], 1-24[1->24], 1-22[22->1], 1-724[724->1], 1-62[1->62], 1-759[1->759], 1-319[1->319], 1-4[1->4], 1-569[569->1], 1-586[1->586], 1-87[87->1], 1-28[28->1], 1-690[1->690], 1-240[240->1], 1-810[810->1], 1-106[106->1], 1-75[75->1], 1-3[1->3], 1-74[1->74], 1-21[1->21], 1-56[56->1], 1-386[386->1], 1-112[112->1], 1-96[1->96], 1-704[1->704], 1-325[325->1], 1-327[327->1], 1-534[534->1], 1-231[1->231], 1-760[760->1], 1-12[12->1], 1-61[1->61], 1-407[1->407], 1-154[154->1], 1-99[1->99], 1-168[168->1], 1-318[318->1], 1-10[1->10], 1-353[353->1], 1-438[438->1], 1-642[642->1], 1-513[513->1], 1-136[1->136], 1-13[13->1], 1-228[1->228], 1-512[512->1], 1-162[1->162], 1-42[42->1], 1-188[188->1], 1-170[1->170], 1-488[1->488], 1-446[446->1], 1-26[26->1], 1-153[153->1], 1-347[1->347], 1-426[1->426], 1-35[35->1], 1-440[440->1], 1-55[55->1], 1-872[872->1], 1-46[1->46], 1-273[1->273], 1-20[1->20]]

node-2,adjacent nodes->[2-931[931->2], 2-158[2->158], 2-493[2->493], 2-816[2->816], 2-176[2->176], 2-30[2->30], 2-651[651->2], 2-193[2->193], 2-0[0->2], 2-88[2->88], 2-481[481->2], 1-2[1->2], 2-112[2->112]]

.

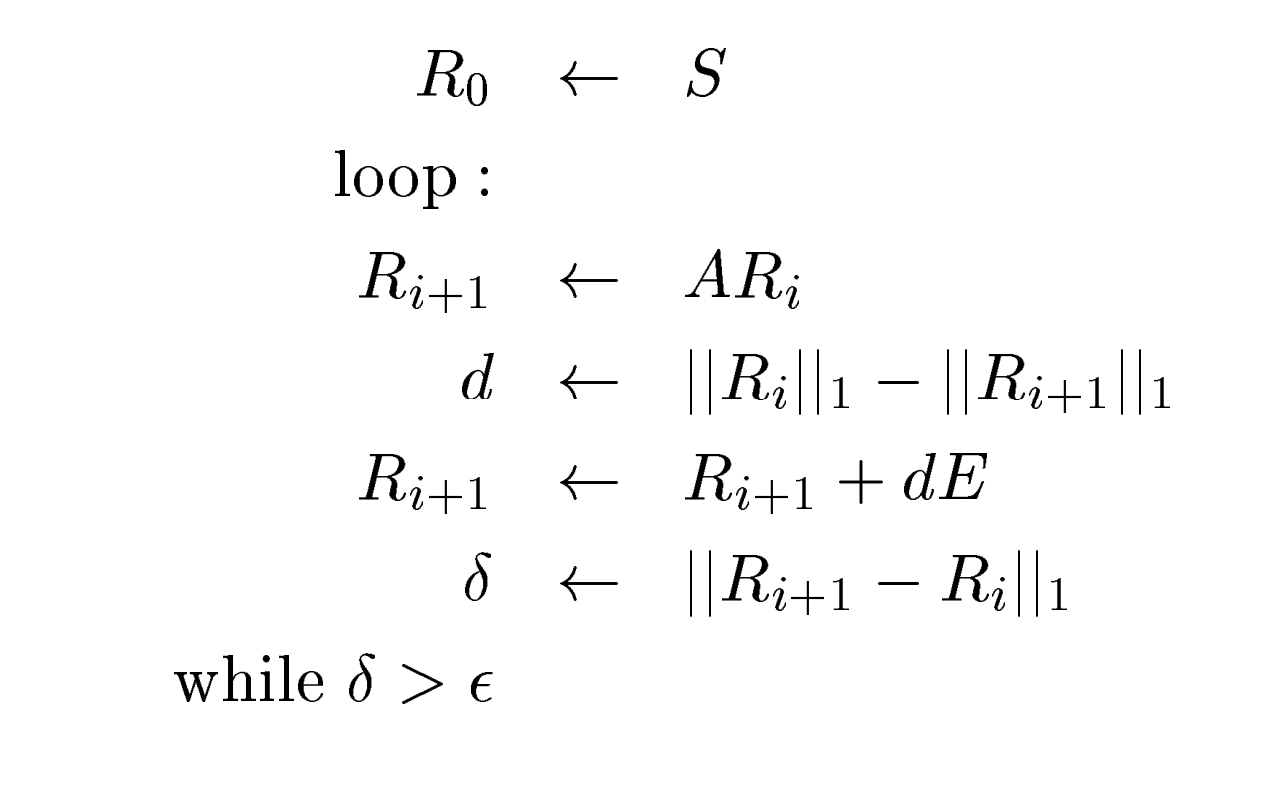
.

.

node-998,adjacent nodes->[77-998[77->998], 802-998[802->998]]

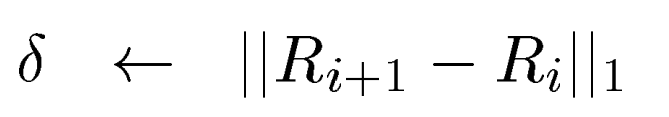
node-999,adjacent nodes->[410-999[999->410], 530-999[999->530]]

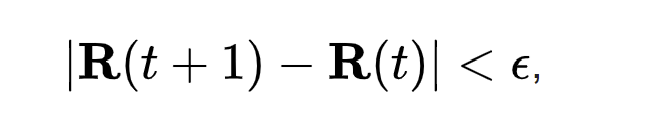
**Power Iterative method to compute the page rank**:



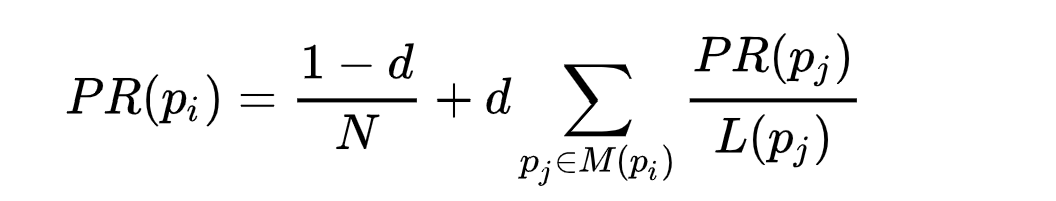
∈ - precision factor

δ – norm distance

**Convergence property** – page rank iteratively computed until norm distance  is reaching the closer value to have the precision. We can control the page rank precision with the *precision* value we given. We assumed it to be from anywhere between 10e-5 to 10e-10.

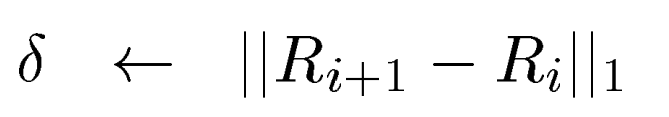
In other words, the computation ends for each page when some small .

**Basic formula of page ranking:**



ranks.add(((1 - df) / graph.getNodeCount()) + df \* sum);

danglingRank = computeDanglingRank(danglingRank, outDegree, node.getNumber("page\_rank"));



normDistance += Math.*abs*((ranks.get(i) + danglingRank) - graph.getNode(i).getNumber("page\_rank"));

**Output ranks in each iteration:**

----------------------------------------

ranks:[1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 0.03878636363636363, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 0.03878636363636363, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 0.03878636363636363, 1.5000000000000001E-4, 0.03878636363636363, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 0.03878636363636363, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4, 1.5000000000000001E-4……………….

**Finalized Output ranks:**

-------------------------------

-----------------

node-0, rank-0.01283871

node-1, rank-0.00984298

node-2, rank-0.00164389

node-3, rank-0.00284136

node-4, rank-0.01890197

node-5, rank-0.00763147

node-6, rank-0.00631208

node-7, rank-0.00077098

node-8, rank-0.00109337

node-9, rank-0.0050875

node-10, rank-0.00647814

node-11, rank-0.00150253

node-12, rank-0.00286248

node-13, rank-0.00565531

node-14, rank-0.00615851

node-15, rank-0.00529753

node-16, rank-0.00168791

node-17, rank-0.00244589

node-18, rank-0.00204916

node-19, rank-0.00082889

node-20, rank-0.00144425

node-21, rank-0.00028885

node-22, rank-0.00156048

node-23, rank-0.00164696

node-24, rank-0.00078489

node-25, rank-0.00380484

node-26, rank-0.00172894

ode-911, rank-0.00091144

node-912, rank-0.0006144

node-913, rank-0.00051075

node-914, rank-0.00028885

node-915, rank-0.00093851

node-916, rank-0.00037069

node-917, rank-0.00053792

node-918, rank-0.00061034

node-919, rank-0.00045466

node-920, rank-0.00050863

node-921, rank-0.0004012

node-922, rank-0.00045354

node-923, rank-0.00032392

node-924, rank-0.00028885

node-925, rank-0.00028885

node-926, rank-0.00096983

node-927, rank-0.00041587

node-928, rank-0.00050629

node-929, rank-0.00041592

node-930, rank-0.00028885

node-931, rank-0.00037069

node-932, rank-0.0006876

node-933, rank-0.00120185

node-934, rank-0.00028885

node-935, rank-0.00053924

node-936, rank-0.0005033

node-937, rank-0.00028885

node-938, rank-0.00080055

node-939, rank-0.00037069

node-940, rank-0.00079987

node-941, rank-0.0012423

node-942, rank-0.00028885

node-943, rank-0.00028885

node-944, rank-0.00028885

node-945, rank-0.00057967

node-971, rank-0.00068986

node-972, rank-0.00070612

node-973, rank-0.00041849

node-974, rank-0.00086118

node-975, rank-0.00028885

node-976, rank-0.00028885

node-977, rank-0.00028885

node-978, rank-0.00060587

node-979, rank-0.00049729

node-980, rank-0.00028885

node-981, rank-0.00051075

node-982, rank-0.00028885

node-983, rank-0.00186956

node-984, rank-0.00096163

node-985, rank-0.00028885

node-986, rank-0.0004012

.

.

.

.

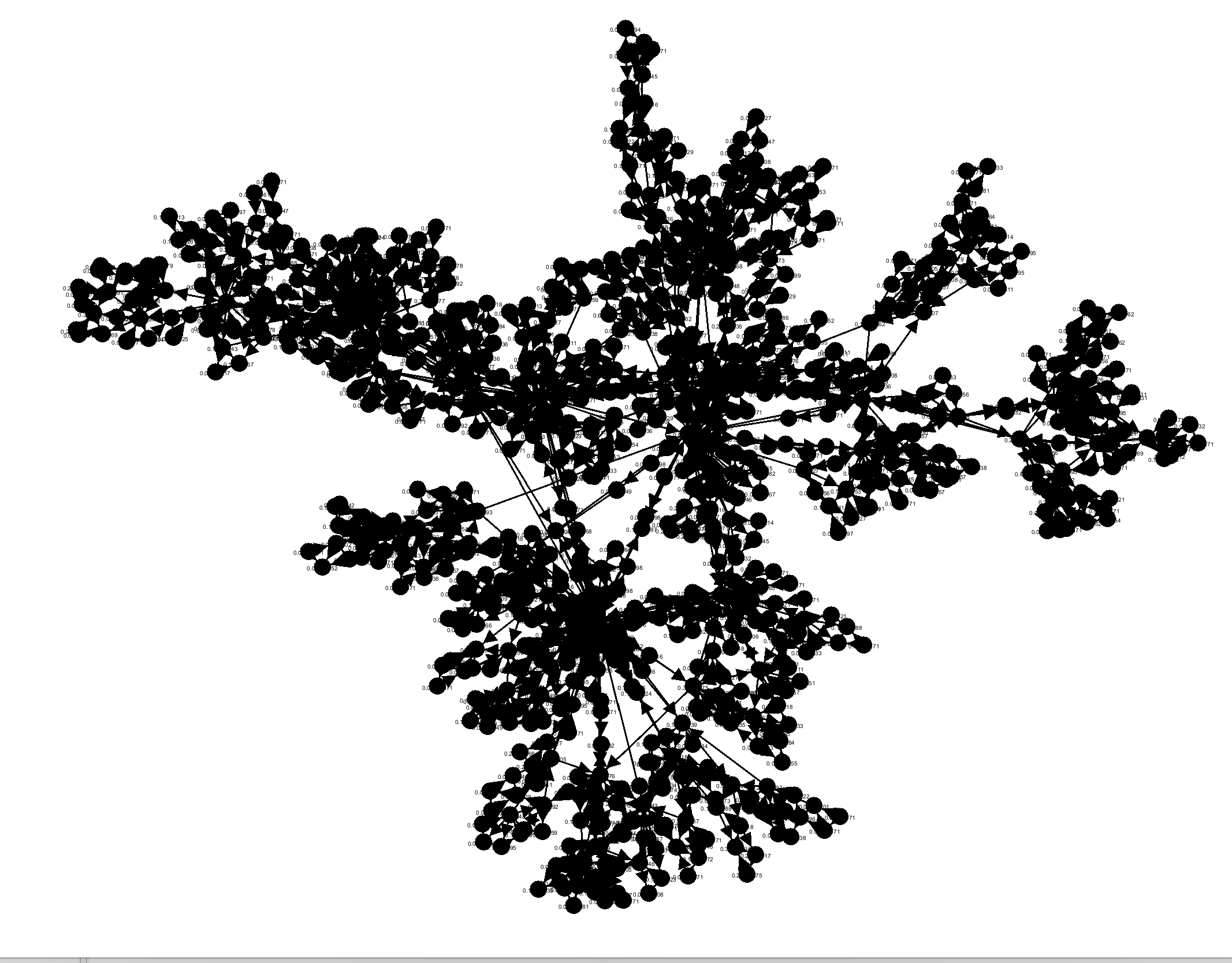
.

node-996, rank-0.00111102

node-997, rank-0.00053747

node-998, rank-0.00055887

node-999, rank-0.0004855



Libraries used for test data generation and visualization of graph:

References:

[PageRank - Wikipedia](https://en.wikipedia.org/wiki/PageRank#:~:text=Various%20studies%20have%20tested%20different,will%20be%20set%20around%200.85.)