NetworkX Cheat Sheet

by gonz95alo via cheatography.com/152363/cs/32835/

The basics

Import library	`import networx as nx`
Import matplotlib for graph drawing	`import matplo tli b.p yplot as plt`
Initialize (Di)graphs	`G = nx.Gra ph(), G = nx.DiG raph()`
Create a (Di)graph from another graph	`G = nx.Gra ph(H), G = nx.DiG raph( H)`
Copy existing graphs	`G2 = G1.copy()`

Unweighted node and edge creation/deletion

Add nodes	`G.add_ node(1)` `G.add_ nod es_ fro m(['a', 'b'])`
Add (directed) edges	`G.add_ edge(1, 2)` `G.add_ edg es_ fro m([(1, 'a'), (1, 'c')] )`
Delete nodes	`G.remo ve_ node(1)` `G.remo ve_ nod es_ fro m(['a', 'b'])`
Delete edges	`G.remo ve_ edge(1, 2)` `G.remo ve_ edg es_ fro m([(1, 'a'), (1, ' c')])`

If G is directed, then these take into account edge direction.

Weights and attributes

Add nodes with attribute	`G.add_ nod es_ fro m([1, 2], color = 'blue'`
Change node attributes	`G.node s[1 ][' color'] = 'red'`
Add edges with attribute	`G.add_ edg es_ fro m([ (1,2), (2,4)], weigh`
Change edge attributes	`G.edge s[1 ,2] ['w eight'] = 2`
Set attributes from dictionary	`nx.set _no de_ att rib utes(G, {1: {'city': d'}})`

Attributes can have any name (color, timestamp, weight, etc). However for edge weights they should have 'weight', as many functions for weighted graphs assume it.

Basic graph properties

Number of nodes	`N = len(G.n od es())`
Number of edges	`E = len(G.e dg es())`
Node-d-egree dictionary	`degreedict = G.degrees`
Direct-edness	`G.is_d ire cted(), G.is_u ndi rec ted()`
Planarity	`G.is_p lanar()`
Diameter, radius	`d = nx.dia met er(G), r = nx.rad ius(G)`
Average shortest path length	`aspl = nx.ave rag e_s hor tes t_p ath _le ngth(G)`

NetworkX Cheat Sheet

by gonz95alo via cheatography.com/152363/cs/32835/

Iterate over a graph

Iterate over nodes/-edges	`nodelist = [n for n in G.nodes()]` `edgelist = [n for n in G.edges()]`
Iterate over node/edge attributes	`node_attrs = [n for n in G.node s(d ata =True )]` `edge_attrs = [n for n in G.edge s(d ata =Tru e)]`
Iterate over in/out edges	`in_list = [e for e in G.in_e dges()]` `out_list = [e for e in G.out_ edg es()]`
Iterate over node's neighbors	`neighs = [n for n in G.neig hbo rs( node)]`
Iterate over succes-sors/pred-ecessors	`succ = [n for n in G.succ ess ors (node)]` `pred = [n for n in G.pred ece sso rs( node)]`

Connectivity properties

Connectedness	`G.is_c onn ected()`
	`G.is_w eak ly_ con nec ted(),`
	`G.is_s tro ngl y_c onn ected()`

Contains node or edge?	`G.has_ nod e(n ode), G.has_ edg e(* edge)`

Connectivity properties (cont)

List of connected components	`cc = nx.con nec ted _co mpo nen ts(G)`
	`wcc = nx.wea kly _co nne cte d_c omp one`
	`scc = nx.str ong ly_ con nec ted _co mpo`

Largest (simply) connected component	`largest_cc = max(nx.co nne cte d_c omp on =len)` `LCC = G.subg rap h(l arg est _cc ).c opy`

Some matricial representations

Adjacency matrix (standard)	`A = nx.to_ num py_ arr ay(G, dtype= int)`
Adjacency matrix (sparse)	`A = nx.to_ sci py_ spa rse _ar ray( G)`
Adjacency matrix eigenvalues	`eigs = nx.adj ace ncy _sp ect rum(G )`
Graph from adjacency matrix	`G = nx.fro m_n ump y_a rray(A)` `G = nx.fro m_s cip y_s par se_ arr ay(A)`
Incidence matrix (sparse)	`I = nx.inc ide nce _ma trix(G)`
Laplacian matrix (sparse)	`L = nx.lap lac ian _ma trix(G)`

Cheatography

NetworkX Cheat Sheet

by gonz95alo via cheatography.com/152363/cs/32835/

Some matricial representations (cont)

Laplacian matrix eigenvalues	`eigs = nx.lap lac ian _sp ect r um(G)`
Google matrix (standard)	`Goog = nx.goo gle _ma trix(G)`

Sparse matrices/arrays are memory efficient. See the Scipy page on them for their methods, or convert them to numpy arrays with `spars e _ar ray.to dense()`

Graph plotting

Plotting command, standard options	`nx.draw(G, pos=pos, with_l abe ls= True, ax=axis )`
Node positioning options (I)	`pos = [nx.ci rcu lar _la you t(G), nx.pla nar _la you t(G), nx.ran dom _la you t(G)]`
Node positioning options (II)	`pos = [nx.sh ell _la you t(G), nx.spr ing _la you t(G)]`
Node colors and sizes in nx.draw()	`node_color = colorlist` `node_size = sizelist`
Edge colors and widths in nx.draw()	`edge_color = colorlist` `width = widthlist`
Node fine-tuning	`nx.dra w_n etw ork x_n odes(G, other options)`
Edge fine-tuning	`nx.dra w_n etw ork x_e dges(G, other options)`

Centrality measures

Degree	`C = nx.deg ree _ce ntr ali ty(G)`
Betweenness	`C = nx.bet wee nes s_c ent ral ity(G)`
Closeness	`C = nx.clo sen ess _ce ntr ali ty(G)`
Eigenvector	`C = nx.eig env ect or_ cen tra lit y_n umpy`
Katz	`C = nx.kat z_c ent ral ity _nu mpy(G, alpha a=1.0)`
PageRank	`C = nx.pag era nk(G, alpha=0.1, person ali None)`
HITS	`C = nx.hits(G)`

All measures return a dictionary `{node: value}`. They do not take into account weights, one needs to provide an extra argument `weigh t ='w eight'` in order to consider them.

Undirected graph generators

Path graph	`G = nx.pat h_g raph(N)`
Cycle graph	`G = nx.cyc le_ gra ph(N)`
Star graph	`G = nx.sta r_g raph(N)`
Complete graph	`G = nx.com ple te_ gra ph(N)`
Erdös-Renyi (random)	`G = nx.erd os_ ren yi_ gra ph(N, p )`
Barabasi-Albert (scale-free)	`G = nx.bar aba si_ alb ert _gr aph (N, m)`
Watts-Strogatz (small world)	`G = nx.wat ts_ str oga tz_ gra ph( N, p)`

By **gonz95alo**
cheatography.com/gonz95alo/

Published 7th September, 2022.
Last updated 23rd October, 2022.