Khai Thác Dữ Liệu Đồ Thị

PHÁT SINH ĐỒ THỊ

Giảng viên: Lê Ngọc Thành

Email: lnthanh@fit.hcmus.edu.vn

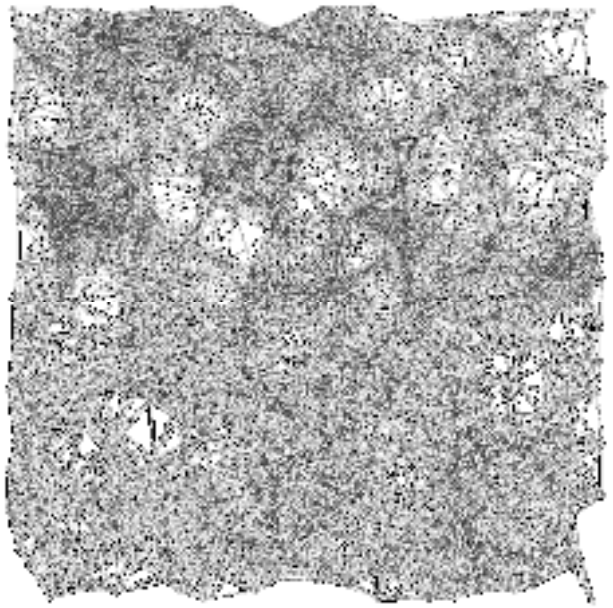
fit@hcmus

Nội dung

- **Phát sinh đồ thị**
- Các phân phối xác suất
- NetworkX để phát sinh đồ thị
  - Tạo và vẽ đồ thị đơn giản
  - Các hàm thực thi trên đồ thị
  - Phát sinh đồ thị tổng hợp

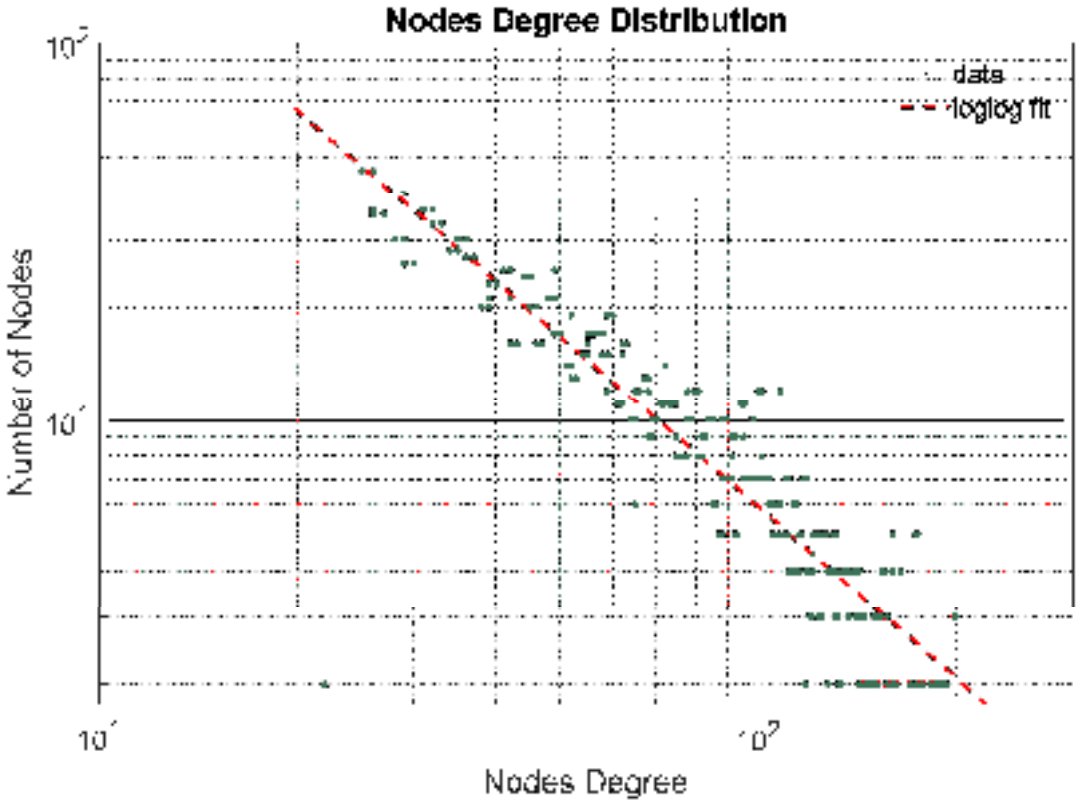
Phát sinh đồ thị

- Mục tiêu phát sinh đồ thị (graph generator) nhằm tạo ra các đồ thị tổng hợp phục vụ cho quá trình nghiên cứu giả lập.
- Yêu cầu đồ thị được phát sinh:
  - Càng giống với thực tế càng tốt



Đồ thị tổng hợp

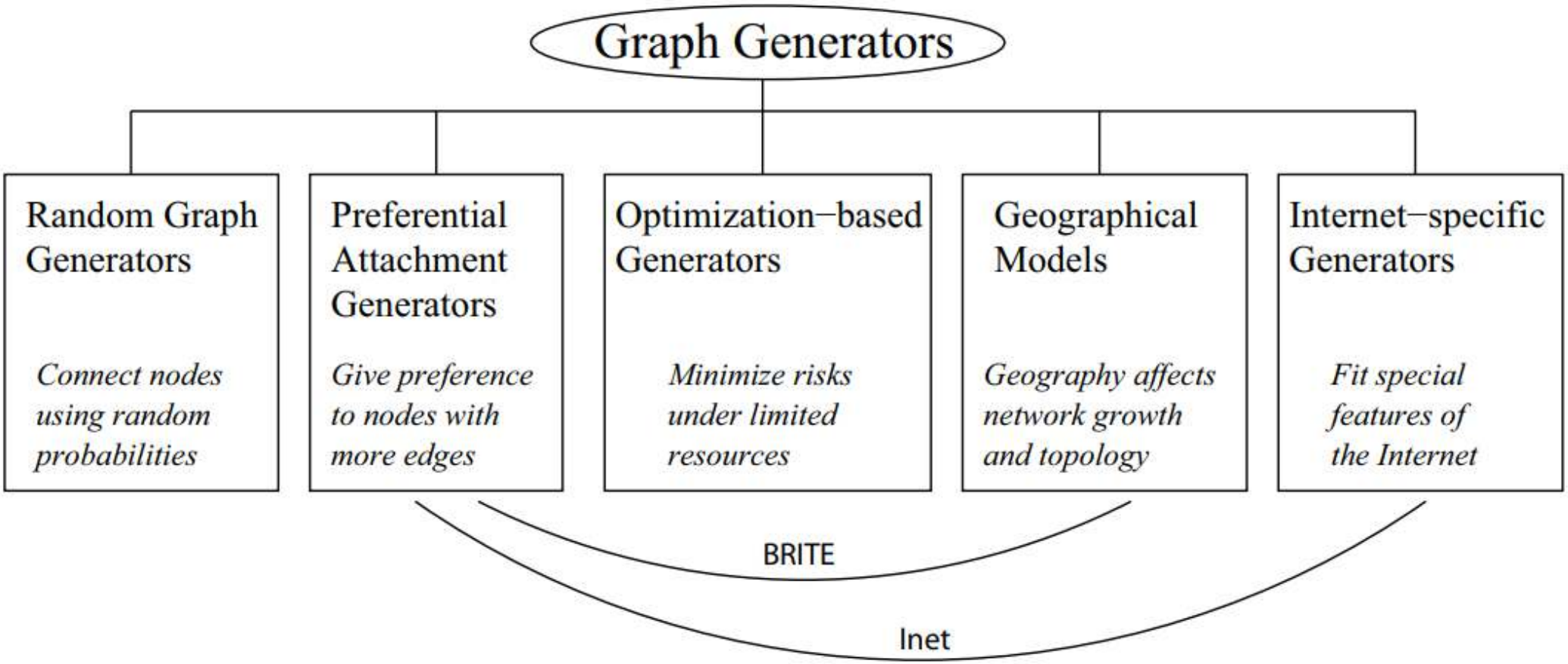
- Đồ thị tổng hợp (synthetic graph) như thế nào để giống với thực tế?
  - Kích thước đủ lớn
  - Tuân theo các luật mẫu



Các loại đồ thị tổng hợp

- Đồ thị tổng hợp được chia làm 5 loại:
  - Mô hình đồ thị ngẫu nhiên (random graph model)
  - Mô hình gắn kết ưu tiên (preferential attachment model)
  - Mô hình dựa trên tối ưu (optimization-based model)
  - Mô hình địa lý (geographical model)
  - Mô hình dành riêng cho Internet (Internet-specific model)

Các loại đồ thị tổng hợp



Nội dung

- Phát sinh đồ thị
- **Các phân phối xác suất**
- NetworkX để phát sinh đồ thị
  - Tạo và vẽ đồ thị đơn giản
  - Các hàm thực thi trên đồ thị
  - Phát sinh đồ thị tổng hợp

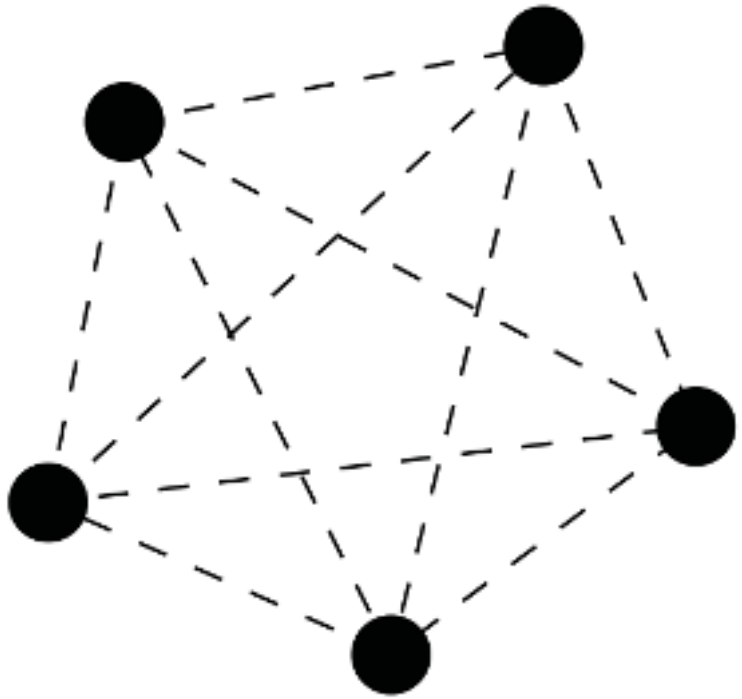
Một số ký hiệu liên quan

Symbol	Description
$\mathcal{G}$	A graph with $(\mathcal{V}, \mathcal{E})$ set of nodes and edges
$\mathcal{V}$	Set of nodes for graph $\mathcal{G}$
$\mathcal{E}$	Set of edges for graph $\mathcal{G}$
N	Number of nodes, or $\lvert\mathcal{V}\rvert$
E	Number of edges, or $\lvert\mathcal{E}\rvert$
$e_{i,j}$	Edge between node i and node j
$w_{i,j}$	Weight on edge $e_{i,j}$
w_i	Weight of node i (sum of weights of incident edges)
$\mathbf{A}$	0-1 Adjacency matrix of the unweighted graph
$\mathbf{A}_w$	Real-value adjacency matrix of the weighted graph
$a_{i,j}$	Entry in matrix $\mathbf{A}$
λ_1	Principal eigenvalue of unweighted graph
$\lambda_{1,w}$	Principal eigenvalue of weighted graph
γ	Power-law exponent: $y(x) \propto x^{-\gamma}$

Symbol	Description
k	Random variable: degree of a node
$< k >$	Average degree of nodes in the graph
CC	Clustering coefficient of the graph
$CC(k)$	Clustering coefficient of degree-k nodes
γ	Power-law exponent: $y(x) \propto x^{-\gamma}$

Đặc điểm của quá trình phát sinh đồ thị

- Hầu hết đồ thị được phát sinh bằng cách:
  - Chọn các đỉnh bất kỳ thông qua một số hàm phân phối xác suất ngẫu nhiên
  - Nối chúng với các cạnh



Các phân phối xác suất

- Trong phương pháp đồ thị ngẫu nhiên, xác xuất để một đỉnh có bậc k:
  - Phương pháp Erdös-Rensyi:

$$p_k = \binom{N}{k} p^k (1-p)^{N-k} \approx \frac{z^k e^{-z}}{k!} \quad \text{with } z = p(N-1)$$

  - Phân phối bậc PLRG:

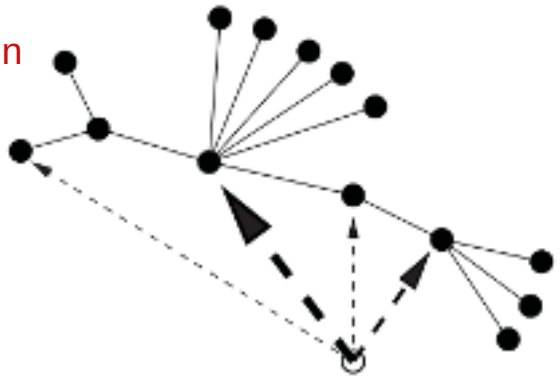
$$p_k \propto k^{-\beta}$$

Các phân phối xác suất

- Trong phương pháp đồ thị gắn kết ưu tiên, xác xuất để một đỉnh có bậc k:
  - Phương pháp Barabási và Albert:

$$P(\text{edge to existing vertex } v) = \frac{k(v) + k_0}{\sum_i (k(i) + k_0)}$$

=> Ưu tiên kết nối với đỉnh có bậc cao
=> Xếp vào nhóm kết nối dựa trên sự ưu tiên



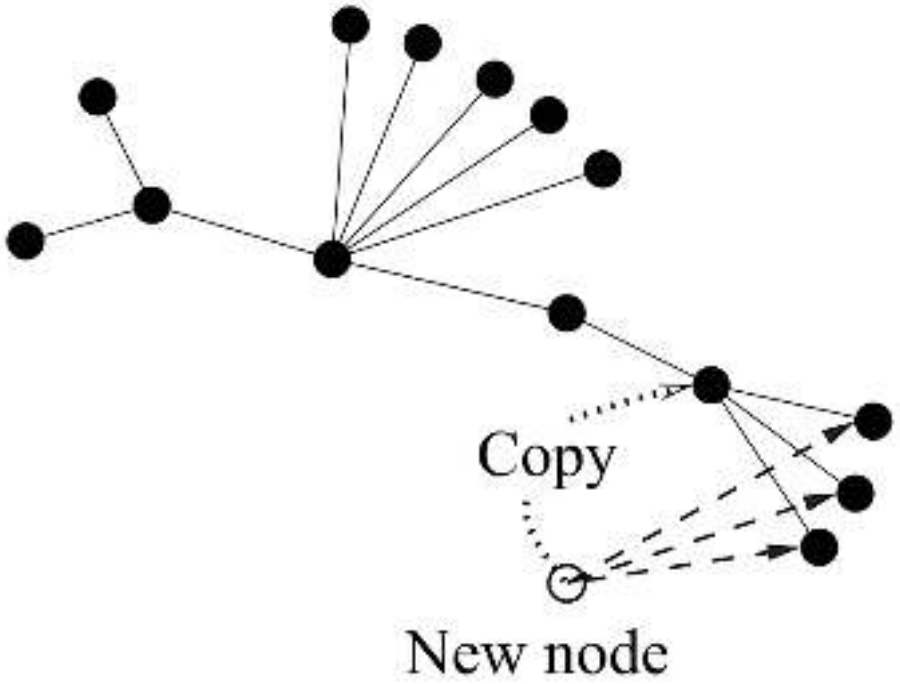
Phương pháp gắn kết đồ thị

- Các bước tiến hành:
  - Phát sinh 4 số ngẫu nhiên $m(n,n)$, $m(n,e)$, $m(e,n)$ và $m(e,e)$ theo một phân phối xác suất
  - Một đỉnh được thêm vào đồ thị sau mỗi vòng lặp
  - $m(n,n)$ cạnh được thêm từ đỉnh mới đến chính nó (khuyên)
  - $m(n,e)$ cạnh được thêm từ đỉnh mới đến ngẫu nhiên các đỉnh đã tồn tại theo quy tắc đỉnh càng có bậc trong càng cao thì càng có xác suất ưu tiên được chọn cao hơn (ưu tiên)
  - $m(e,n)$ cạnh được thêm từ các đỉnh đang tồn tại đến đỉnh mới, các đỉnh tồn tại được chọn ngẫu nhiên với xác suất dựa trên bậc ngoài của chúng.
  - $m(e,e)$ cạnh được thêm giữa các đỉnh đang tồn tại, cũng dựa trên bậc trong và bậc ngoài.
- Đỉnh nào không có bậc trong hay bậc ngoài thì bỏ đi.

Các phân phối xác suất

- Một số biến thể của phương pháp gắn kết ưu tiên:
  - Các đỉnh mới có thể chọn sao chép các cạnh của một đỉnh đang tồn tại.

- Copy số đỉnh, tạo ra bậc giống
- Copy những đối tượng đc nối

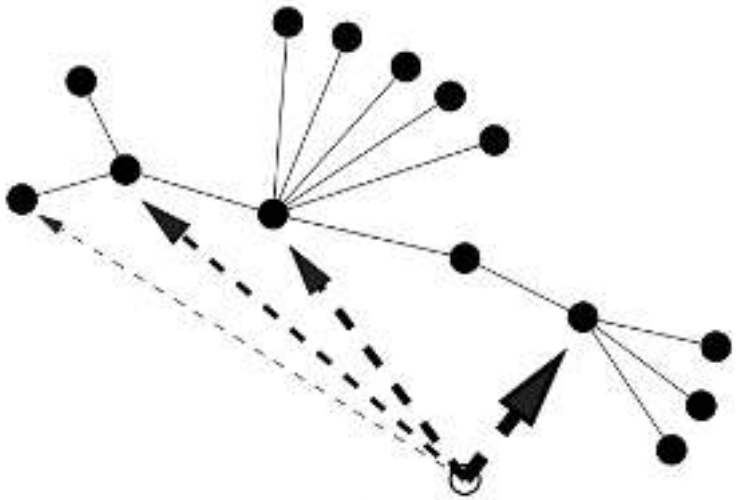


Các phân phối xác suất

- Trong phương pháp đồ thị địa lý, xác xuất để tạo một cạnh
  - Phương pháp Waxman:

$$P(u, v) = \beta \exp \frac{-d(u, v)}{L\alpha}$$

Với $d(u, v)$ là khoảng cách Euclid giữa hai đỉnh.



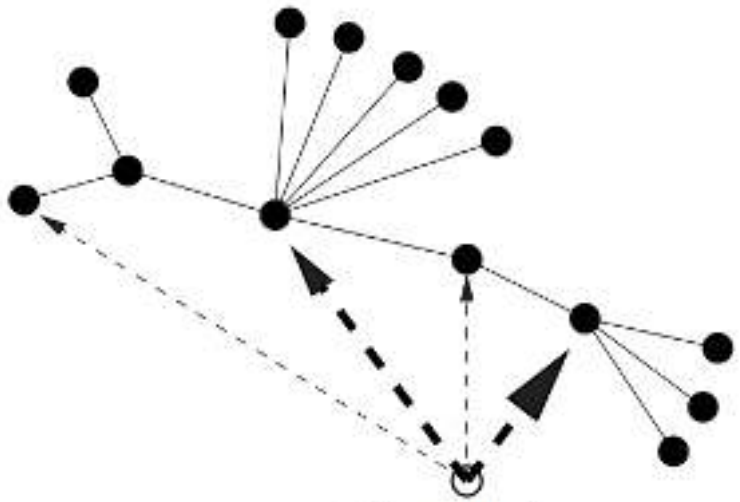
=> ưu tiên cho những node gần

Các phân phối xác suất

- Trong phương pháp đồ thị địa lý, xác xuất để tạo một cạnh
  - Phương pháp heuristic:

$$\alpha.d_{ij} + h_j \quad (j < i)$$

Với d_{ij} là khoảng cách giữa hai đỉnh, h_j là độ đo heuristic thể hiện tính trung tâm của đỉnh j



Nội dung

- Phát sinh đồ thị
- Các phân phối xác suất
- **NetworkX để phát sinh đồ thị**
  - Tạo và vẽ đồ thị đơn giản
  - Các hàm thực thi trên đồ thị
  - Phát sinh đồ thị tổng hợp

NetworkX

- NetworkX là một gói Python để tạo, thao tác và nghiên cứu cấu trúc, tính động và chức năng của các mạng phức tạp.
- Cài đặt:

  $ pip install networkx

  Hoặc

  $ pip install networkx[all]

Cách sử dụng cơ bản

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g.add_node("spam")
>>> g.add_edge(1,2)
>>> print(g.nodes())
[1, 2, 'spam']
>>> print(g.edges())
[(1, 2)]
```

Các loại đồ thị được hỗ trợ bởi NetworkX

- Graph: đơn đồ thị vô hướng (cho phép có khuyên)
- DiGraph: đơn đồ thị có hướng (cho phép có khuyên)
- MultiGraph: đa đồ thị vô hướng với các cạnh song song
- MultiDiGraph: đa đồ thị có hướng với các cạnh song song

```
>>> g = nx.Graph()
>>> d = nx.DiGraph()
>>> m = nx.MultiGraph()
>>> h = nx.MultiDiGraph()
```

Có thể chuyển từ đồ thị có hướng sang vô hướng và ngược lại:
g.to_undirected()
g.to_directed()

Thêm đỉnh

- add_nodes_from(): thêm đỉnh vào đồ thị với đối số là bất kỳ tập hợp có thứ tự hay bất kỳ đối tượng nào.

```
>>> g = nx.Graph()
>>> g.add_node('a')
>>> g.add_nodes_from( ['b','c','d'])
>>> g.add_nodes_from('xyz')
>>> h = nx.path_graph(5)
>>> g.add_nodes_from(h)
>>> g.nodes()
[0,1,'c','b',4,'d',2,3,5,'x','y','z']
```

Thêm cạnh

- add_edge(): thêm cạnh vào đồ thị
  - Nếu đỉnh không tồn tại sẽ tự động thêm đỉnh
- add_edges_from(): thêm các cạnh từ tập hợp

```
>>> g = nx.Graph( [('a','b'),('b','c'),('c'
    ,'a')] )
>>> g.add_edge('a', 'd')
>>> g.add_edges_from([('d', 'c'), ('d', 'b')
    ])
```

Thêm thuộc tính cho đỉnh

- Trong quá trình thêm đỉnh, ta có thể thêm thuộc tính cho các đỉnh.

```
>>> g = nx.Graph()
>>> g.add_node(1,name='Obrian')
>>> g.add_nodes_from([2],name='Quintana'])
>>> g.nodes[1]['name']
'Obrian'
```

Thêm thuộc tính cho cạnh

- Tương tự thêm thuộc tính cho đỉnh, ta có thể thêm thuộc tính cho cạnh

```
>>> g.add_edge(1, 2, w=4.7 )
>>> g.add_edges_from([(3,4),(4,5)], w =3.0)
>>> g.add_edges_from([(1,2,{'val':2.0})])
# adds third value in tuple as 'weight' attr
>>> g.add_weighted_edges_from([(6,7,3.0)])
>>> g.get_edge_data(3,4)
{'w' : 3.0}
>>> g.add_edge(5,6)
>>> g[5][6]
{}
```

Xóa các phần tử

- Có thể xóa các nút và cạnh khỏi đồ thị theo cách tương tự như thêm vào.

```
>>> G.remove_node(2)
>>> G.remove_nodes_from("spam")
>>> list(G.nodes)
[1, 3, 'spam']
>>> G.remove_edge(1, 3)
```

Xem một số đặc trưng

- Có thể xem một số đặc trưng

```
>>> list(G.nodes)
[1, 2, 3, 'spam', 's', 'p', 'a', 'm']
>>> list(G.edges)
[(1, 2), (1, 3), (3, 'm')]
>>> list(G.adj[1])  # or list(G.neighbors(1))
[2, 3]
>>> G.degree[1]  # the number of edges incident to 1
2
```

Đọc đồ thị từ file

- NetworkX còn có chức năng load dữ liệu đồ thị từ tập tin.

```
>>> file = 'wiki.txt'
>>> wiki = nx.read_adjlist(file, delimiter
    ='\t', create_using=nx.DiGraph())
```

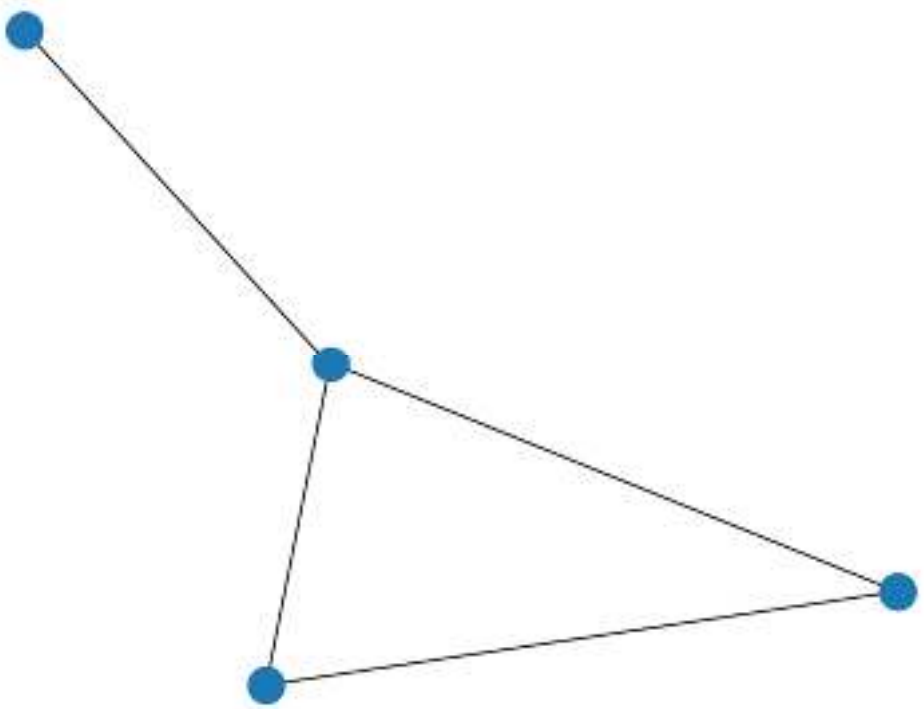
Vẽ đồ thị

- Ta sử dụng thư viện Matplotlib để vẽ đồ thị từ networkx
  - Xác định backend để vẽ lên tập tin hoặc trình diễn trực tiếp

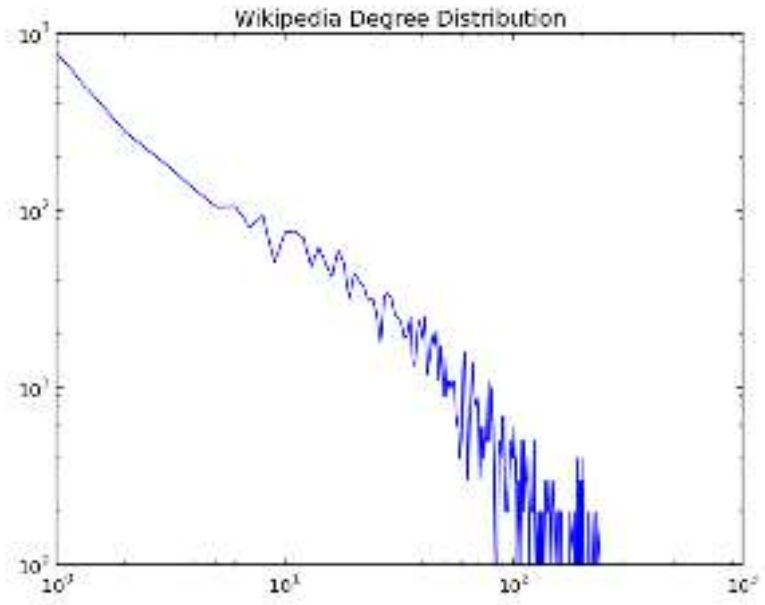
```
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
```

Ví dụ vẽ đồ thị

```
G = nx.Graph()
G.add_edges_from([(1,2), (2,3), (1,3),
    (1,4)])
nx.draw(G)
plt.savefig("simple_graph.png")
```



Vẽ phân phối bậc




```
degs = {}
for n in wiki.nodes():
  deg = wiki.degree(n)
  if deg not in degs:
    degs[deg] = 0
  degs[deg] += 1
items = sorted(degs.items())
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([k for (k,v) in items], [v for (k,
   v) in items])
ax.set_xscale('log')
ax.set_yscale('log')
plt.title("Wikipedia Degree Distribution")
fig.savefig("degree_distribution.png")
```

Các thuật toán trên đồ thị

- Nhiều thuật toán trên đồ thị đã được NetworkX thực hiện (networkx.algorithms)

```
>>> import networkx as nx
>>> help(nx.algorithms)
```

- bipartite
- block
- boundary
- centrality (package)
- clique
- cluster
- components (package)
- core
- cycles
- dag
- distance measures
- flow (package)
- isolates
- isomorphism (package)
- link_analysis (package)
- matching
- mixing
- mst
- operators
- shortest_paths (package)
- smetric

Các thuật toán trên đồ thị

- Ví dụ

shortest path

```
nx.shortest_path(G,s,t)
```

clustering

```
nx.average_clustering(G)
```

diameter

```
nx.diameter(G)
```

Các thuật toán trên đồ thị

- Ví dụ

As subgraphs

```
nx.connected_component_subgraphs(G)
```

Operations on Graph

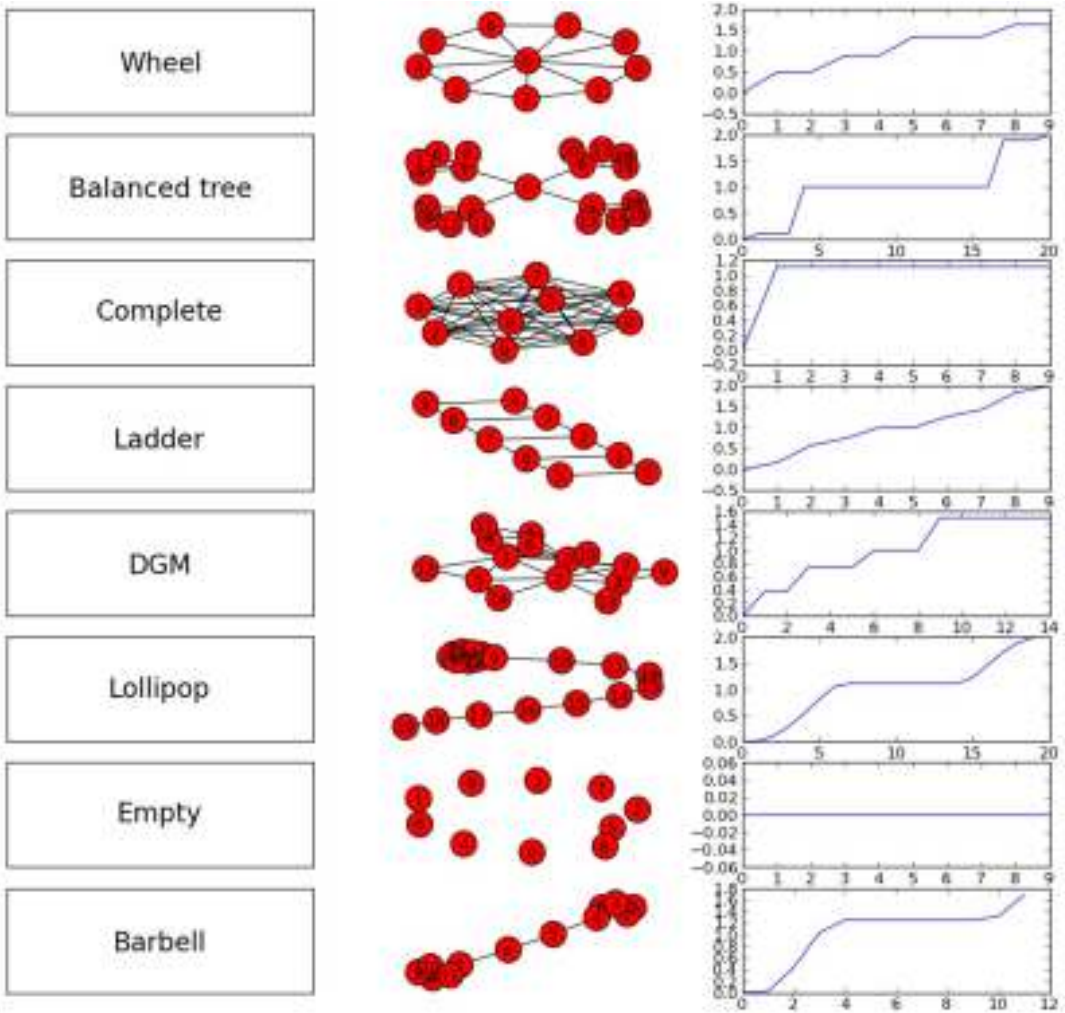
```
nx.union(G,H), intersection(G,H),
    complement(G)
```

k-cores

```
nx.find_cores(G)
```

Phát sinh đồ thị

- Bộ phát sinh đồ thị của NetworkX được đặt trong module networkx.generators



Phát sinh đồ thị đơn

Complete Graph

```
nx.complete_graph(5)
```

Chain

```
nx.path_graph(5)
```

Bipartite

```
nx.complete_bipartite_graph(n1, n2)
```

Arbitrary Dimensional Lattice (nodes are tuples of ints)

```
nx.grid_graph([10,10,10,10]) # 4D, 100^4
    nodes
```

Phát sinh đồ thị ngẫu nhiên

Preferential Attachment

```
nx.barabasi_albert_graph(n, m)
```

$G_{n,p}$

```
nx.gnp_random_graph(n,p)
```

```
nx.gnm_random_graph(n, m)
```

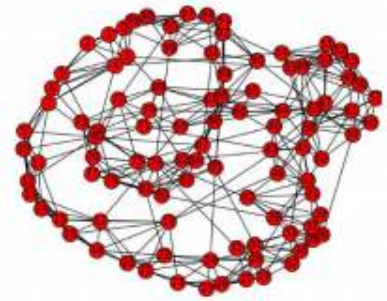
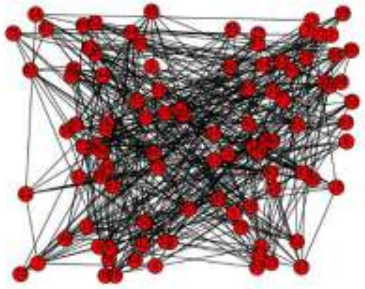
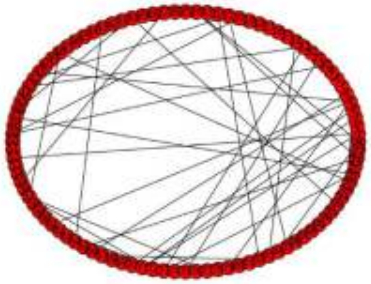
```
nx.watts_strogatz_graph(n, k, p}
```

Phát sinh đồ thị

```
# small famous graphs
>>> petersen = nx.petersen_graph()
>>> tutte = nx.tutte_graph()
>>> maze = nx.sedgewick_maze_graph()
>>> tet = nx.tetrahedral_graph()
```

Vẽ đồ thị được phát sinh

```
>>> import pylab as plt #import Matplotlib plotting interface
>>> g = nx.watts_strogatz_graph(100, 8, 0.1)
>>> nx.draw(g)
>>> nx.draw_random(g)
>>> nx.draw_circular(g)
>>> nx.draw_spectral(g)
>>> plt.savefig('graph.png')
```

Vẽ phân phối bậc cho đồ thị phát sinh

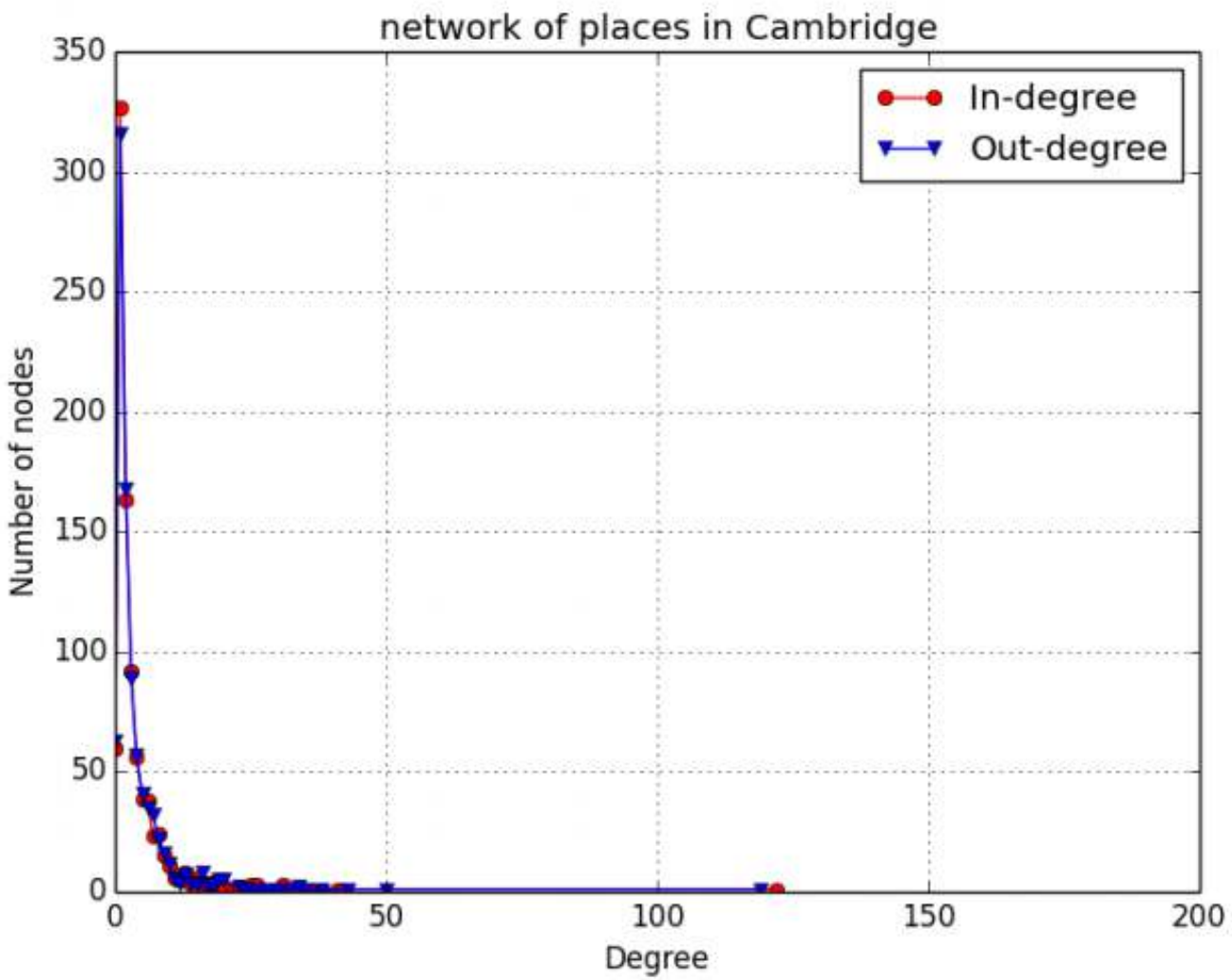
- Calculate in (and out) degrees of a directed graph

```
in_degrees = cam_net.in_degree() # dictionary node:degree
in_values = sorted(set(in_degrees.values()))
in_hist = [in_degrees.values().count(x) for x in in_values]
```

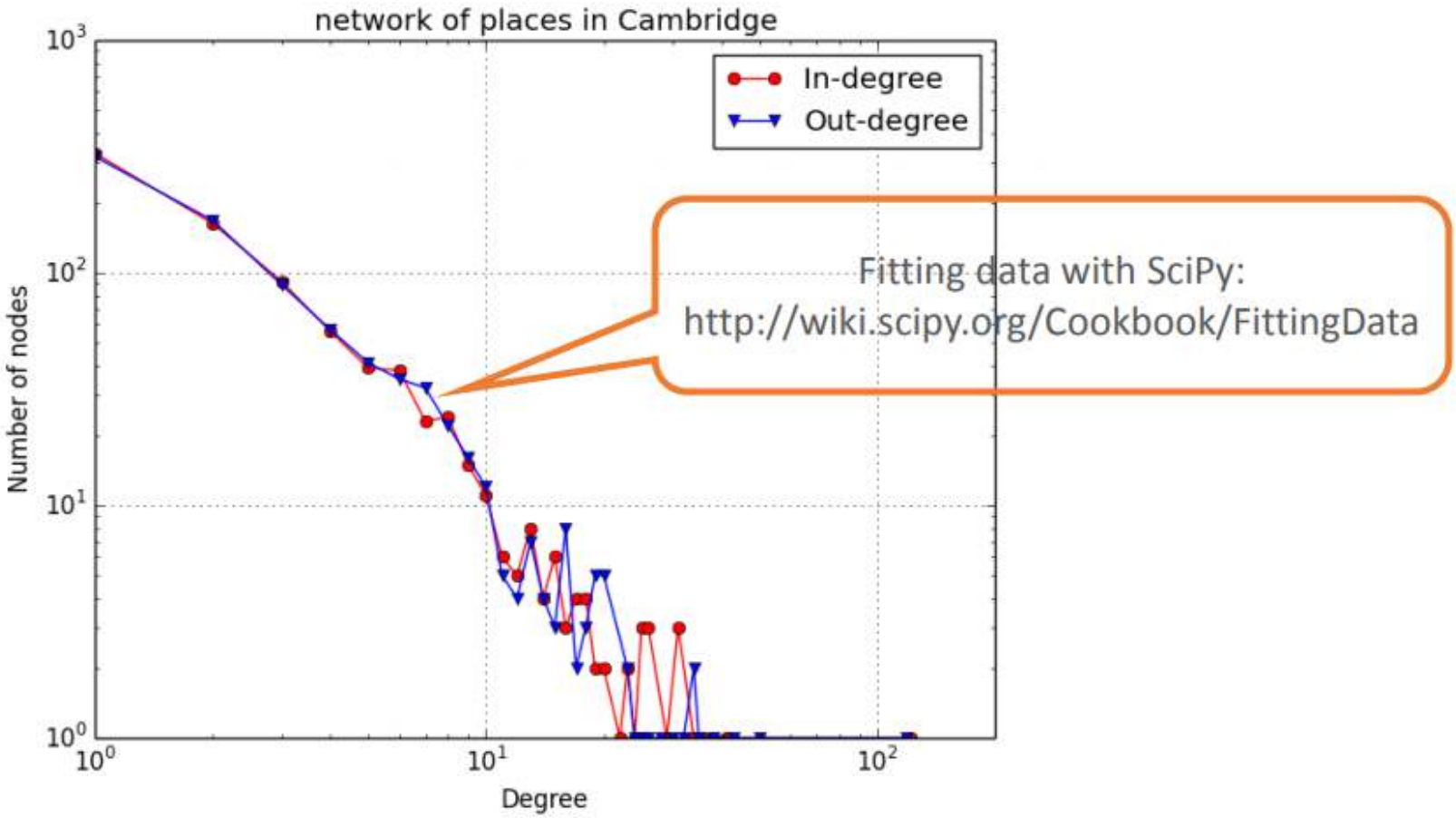
- Then use matplotlib (pylab) to plot the degree distribution

```
plt.figure() # you need to first do 'import pylab as plt'
plt.grid(True)
plt.plot(in_values, in_hist, 'ro-') # in-degree
plt.plot(out_values, out_hist, 'bv-') # out-degree
plt.legend(['In-degree', 'Out-degree'])
plt.xlabel('Degree')
plt.ylabel('Number of nodes')
plt.title('network of places in Cambridge')
plt.xlim([0, 2*10**2])
plt.savefig('./output/cam_net_degree_distribution.pdf')
plt.close()
```

Vẽ phân phối bậc cho đồ thị phát sinh



Vẽ phân phối bậc cho đồ thị phát sinh



Change scale of the x and y axes by replacing

```
plt.plot(in_values,in_hist,'ro-')
```

with

```
plt.loglog(in_values,in_hist,'ro-')
```

Tài liệu tham khảo

- Chakrabarti, D. and Faloutsos, C., 2012. Graph mining: laws, tools, and case studies. Synthesis Lectures on Data Mining and Knowledge Discovery, 7(1), pp.1-207.
- https://www.cl.cam.ac.uk/teaching/1314/L109/tutorial.pdf
- https://www.slideshare.net/shankyme/nx-tutorial-basics