Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Khoa Công nghệ Thông tin

---o0o---

Lab 2: Community Detection

Môn: Khai thác dữ liệu đồ thị

Giảng viên lý thuyết: - Lê Ngọc Thành

Giảng viên thực hành:

- **Phạm Trọng Nghĩa**

Sinh Viên:

- **Nguyễn Trung Hiếu - 19127403**

Table of Contents

I. Thuật toán phát hiện cộng đồng.

1. Thuật toán Girven-Newman.

a. Giới thiệu.

Michelle Girvan và Mark Newman đề xuất ra một thuật toán phát hiện cộng đồng đáp ứng các mục tiêu trong đó có 3 tính chất:

- Là phương pháp chia nhỏ, trong đó cạnh được loại bỏ dần dần ra đồ thị.

- Các cạnh loại bỏ trong mỗi bước được xác định qua độ đo edge centrality.

- Sau khi loại bỏ, các cạnh còn lại được tính lại.

Do bản chất của edge centrality là một cạnh phân chia làm cho các bên cụm muốn đi qua cụm khác thì đi qua các cạnh có edge centrality cao.

Thuật toán cải tiến từ Girven-Newman là thuật toán CONGA nhằm mục đích giải quyết vấn đề chồng chéo cộng đồng.

b. Ý tưởng thuật toán.

Bước 1: tính toán edge betweeness centrality của toàn bộ cạnh trong đồ thị.

Bước 2: Sắp xếp để lấy ra cạnh có độ đo cao nhất.

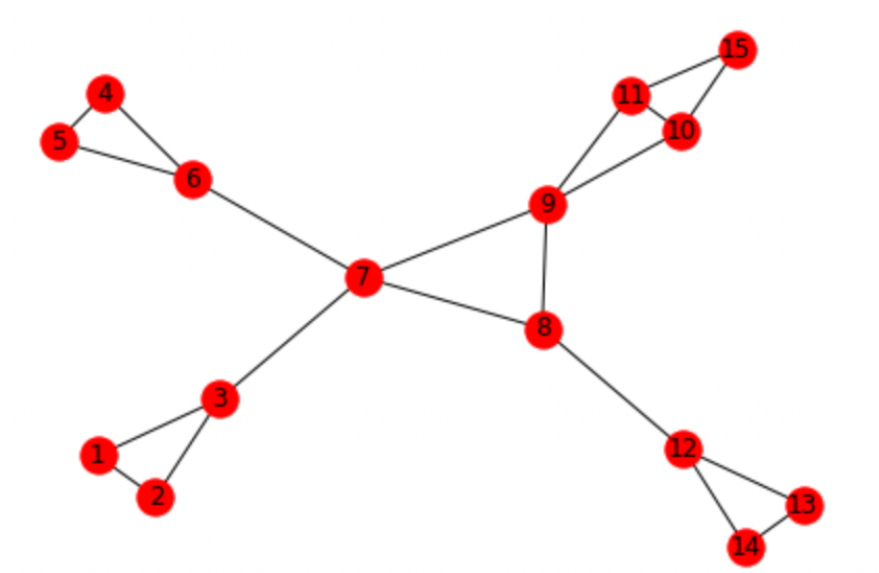
Bước 3: Xóa cạnh có độ đo cao nhất ra khỏi đồ thị.

Bước 4: tính toán lại edge betweeness centrality của toàn bộ cạnh trong đồ thị.

Bước 5: lặp lại các từ bước 2 đến 5 cho đến khi độ đo của các cạnh bằng nhau hoặc có thể chạy n lần do người dùng cài đặt.

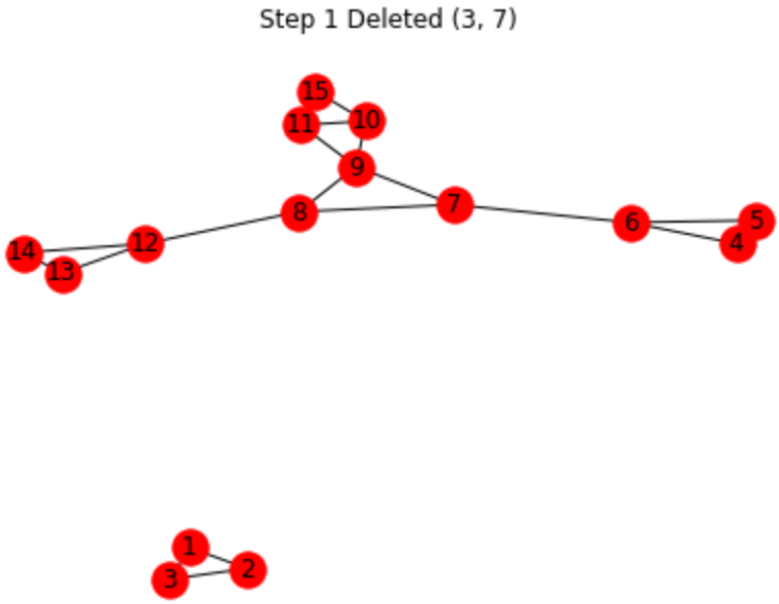
c. Ứng dụng trên đồ thị nhỏ.

Đồ thị ban đầu.

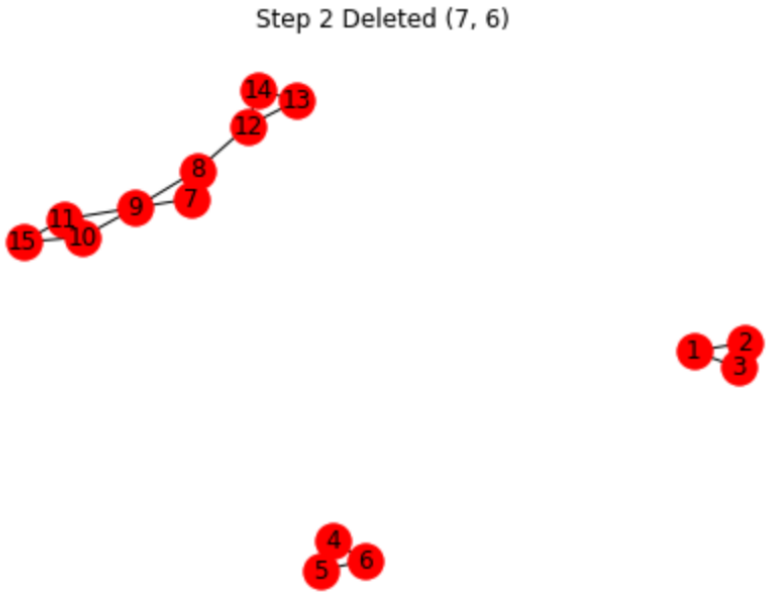


Chạy thuật toán.

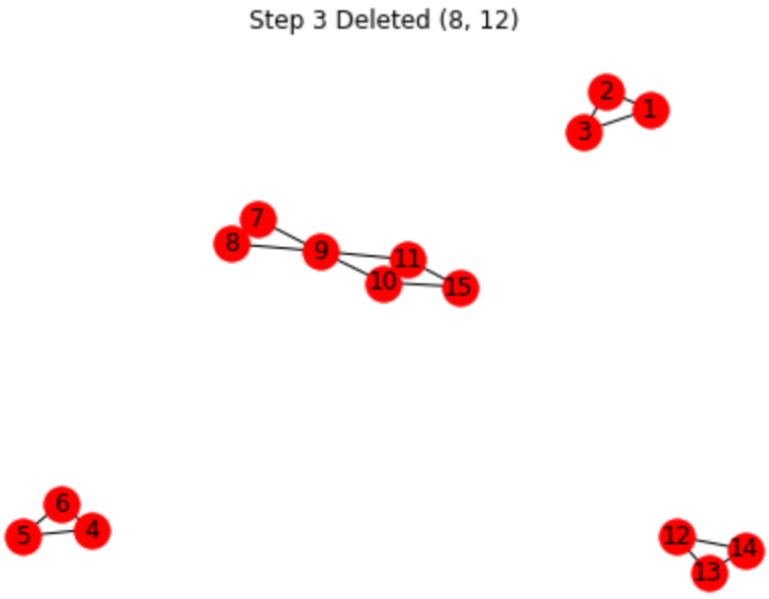
Để dễ dàng trực quan biểu đồ, ta có thể nhìn thấy trong hình có 5 nhóm cộng đồng. Bây giờ ta ứng dụng thuật toán vào sơ đồ và xem cách giải quyết của thuật toán.



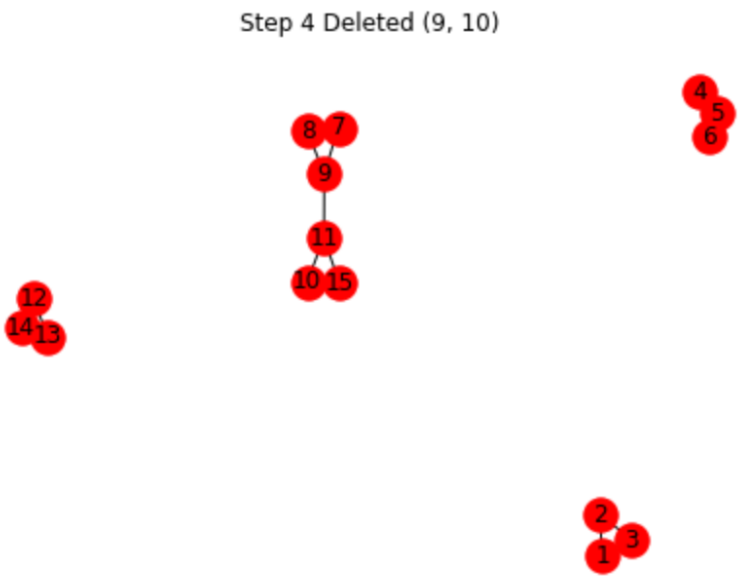
Ở bước 1 ta xóa cạnh (3,7) vì có edge betweeness cao nhất.



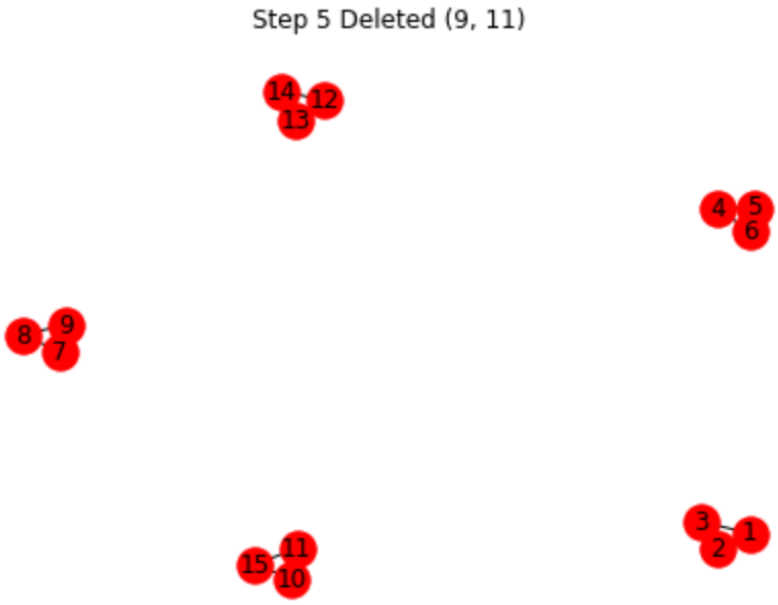
Bước 2 xóa cạnh 7,6 và tạo ra được 3 cộng đồng.



Bước 3 xóa cạnh (8,12) và tạo thành 4 cộng đồng.



Bước 4 xóa cạnh (9,10) và vẫn còn 4 cộng đồng và tiếp tục chạy tiếp.



Bước 5 là bước cuối cùng và tạo thành 5 cộng đồng và ngừng thuật toán.

d. Ứng dụng trên đồ thị lớn.

2. Thuật toán Fluid Communities.

a. Giới thiệu.

Là thuật toán học không giám sát. Tuy nhiên, với nhược điểm là ta cần phải xác định số lượng cộng đồng trước khi chạy thuật toán này. Ý tưởng dựa trên việc, node đang xét sẽ thuộc về cộng đồng mà có tổng trọng số node hàng xóm của nó kết nối tới nó là lớn nhất.

b. Ý tưởng thuật toán.

Bước 1: Gán ngẫu nhiên các điểm khác nhau trên đồ thị dựa trên số lượng cộng đồng đã được định nghĩa trước, và trọng số khởi tạo là 1 với các group.

Bước 2: Xét một điểm bất kỳ trong đồ thị và xem xét các node hàng xóm và tính trọng số của k lớp. Chọn trọng số cao nhất thuộc group nào thì điểm đang xét thuộc về group đó. Đối với trường hợp trọng số bằng nhau thì ta chọn ngẫu nhiên.

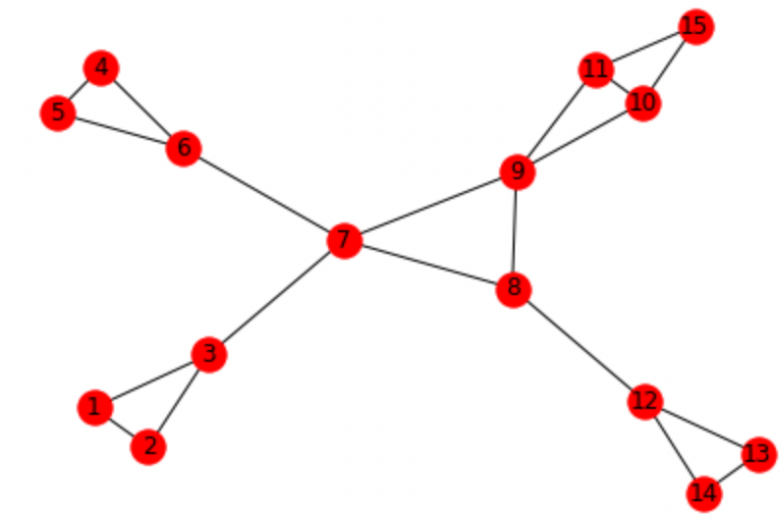
Bước 3: Sau khi kết thúc hết các đỉnh cần xét và các node đều thuộc về một group. Ta lặp lại thuật toán một lần nữa và chạy lại toàn bộ đỉnh.

Bước 4: lặp lại cho đến khi không còn sự thay đổi về việc các node nhảy qua lại các group.

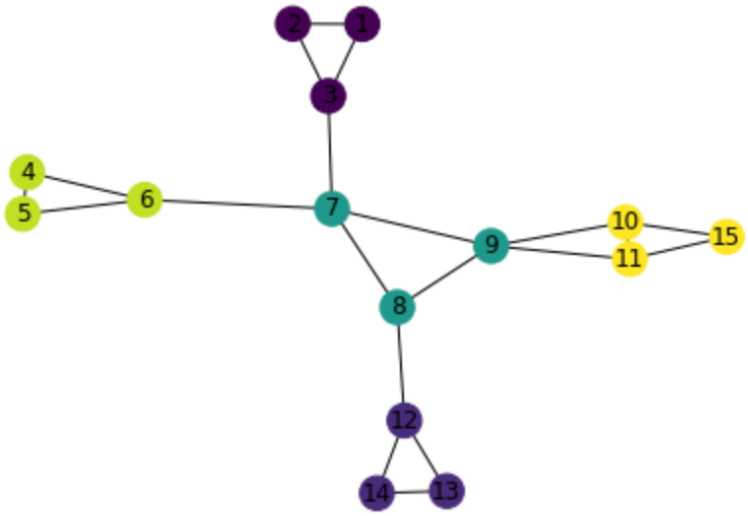
c. Ứng dụng trên đồ thị nhỏ.

Lưu ý: Do thuật toán set random các điểm được chọn ban đầu. Với mỗi lần chạy thì kết quả sẽ chạy khác nhau.

Đồ thị ban đầu.

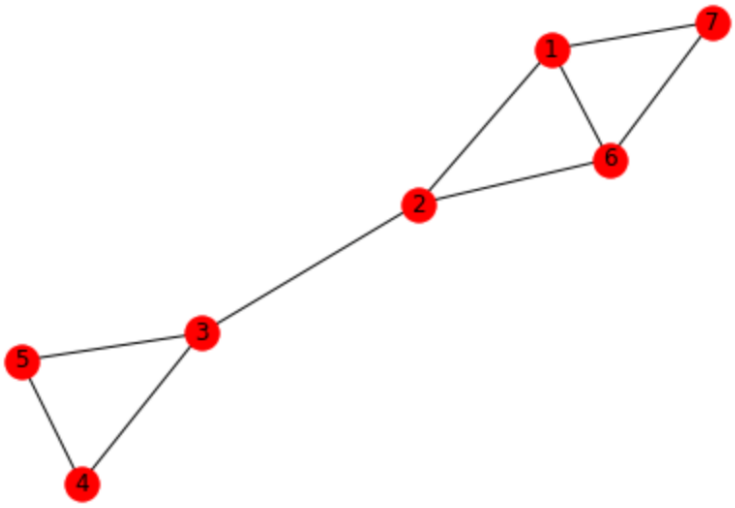


Đồ thị sau khi chạy thuật toán.

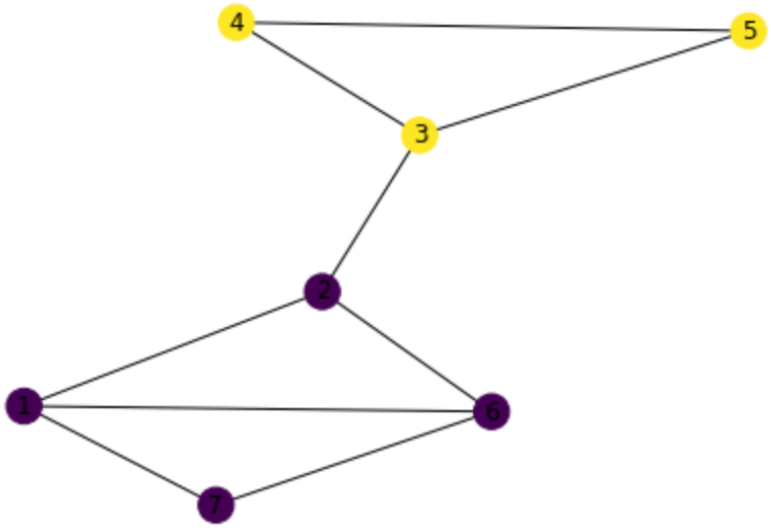



```
Group: [[4, 6, 5], [13, 12, 14], [9, 7, 8], [10, 11, 15], [2, 3, 1]]
Weight: [0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
Node được lấy ngẫu nhiên ban đầu: [[7], [11], [10], [14], [4]]
```

Đồ thị thứ hai.



Đồ thị sau khi chạy thuật toán.




```
Group: [[1, 2, 6, 7], [3, 5, 4]]
Weight: [0.25, 0.3333333333333333]
Node được lấy ngẫu nhiên ban đầu: [[6], [2]]
```

d. Ứng dụng trên đồ thị thế giới thật.

Bộ dữ liệu Facebook

```
Take 0.46785902976989746 to run asyn fluidc algorithm
```

Mất 0.46 giấy để chạy bộ dữ liệu này.

Bộ dữ liệu LastFM.

```
Take 0.4736039638519287s to run asyn fluidc algorithm
```

Mất 0.47s giấy để chạy bộ dữ liệu này.

e. Cách sử dụng thuật toán đã viết trong bài.

Input đầu vào:

Dòng đầu là số lượng đỉnh n (1 <= n <= 100) và số lượng cộng đồng (k >= 1).

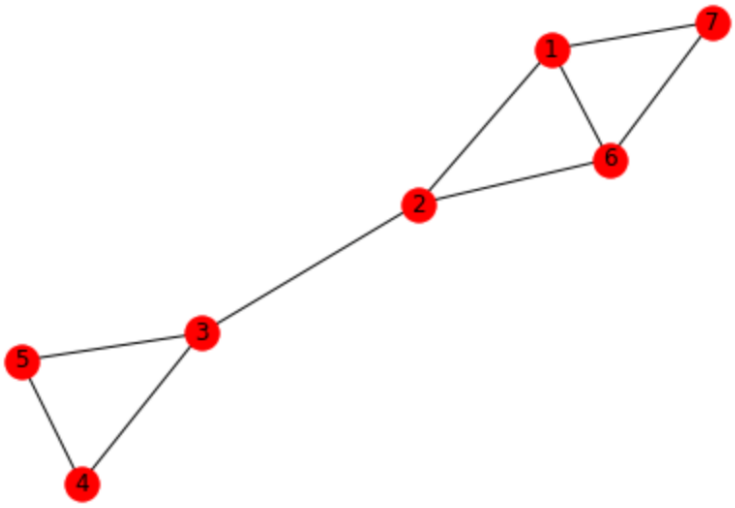
Các dòng tiếp theo là một ma trận kề (Edge list) nối cạnh u với v. Câu lệnh dừng lại với u,v là 0.

Lưu ý: do jupyter notebook chỉ nhận input vào 1 dòng nên ta cần phải đưa toàn bộ input về cùng 1 dòng.

Example: được sử dụng trong bài.

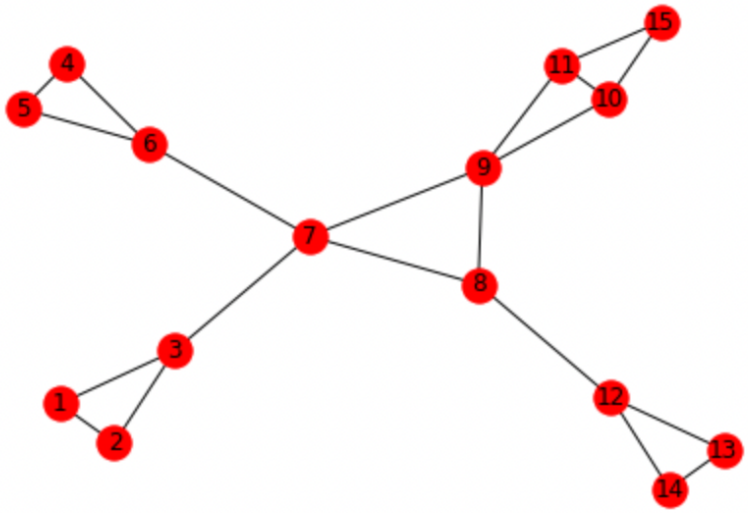
```
7 2 1 2 1 6 1 7 2 3 2 6 3 2 3 4 3 5 4 5 6 7 0 0
```

Có đồ thị là:



15 5 1 2 1 3 2 3 3 7 4 6 4 5 5 6 6 7 7 8 8 9 8 12 9 10 9 11 10 11 12 13 13 14 12 14 8 9 7 9 15 11 15 10 0 0

Có đồ thị là:



II. Mô tả đồ thị thực tế.

1. Bộ dữ liệu Facebook.

a. Link tham khảo.

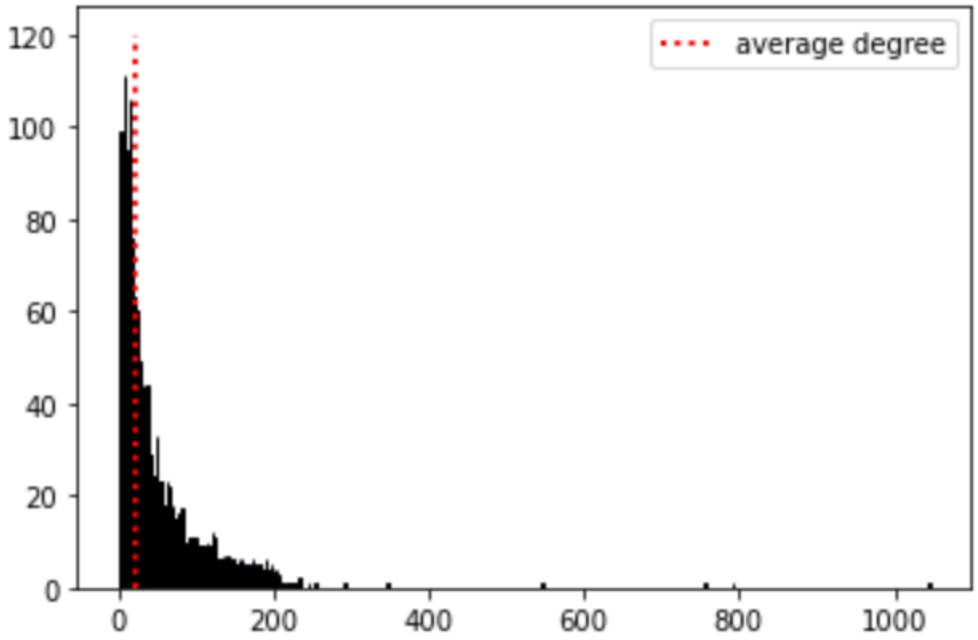
https://raw.githubusercontent.com/alexaverbuch/shortestpath_bench/master/data/raw/facebook_combined.txt

b. Thông tin mô tả cơ bản.

Dữ liệu với mỗi dòng là mỗi user có kết bạn với user khác. Sau đây là mô tả cơ bản của đồ thị

```
Graph with 4039 nodes and 88234 edges
Average Degree 21.84550631344392
```

Đồ thị có 4039 đỉnh và 88234 cạnh. Với bậc trung bình của đồ thị là 21.85. Dưới đây là phân bố bậc của đồ thị.



Bậc phân bố của đồ thị cho thấy có nhiều node chiếm bậc thấp khiến cho bậc trung bình bị giảm xuống đáng kể. Trung bình một người có 28 người bạn.

Đường kính của đồ thị cho thấy độ trải dài của đồ thị như thế nào.

```
In [157]: nx.diameter(G)
Out[157]: 8
```

Với đường kính của đồ thị là 8 cho ta thấy đồ thị này có độ trải dài khá rộng.

Bởi vì bậc trung bình của đồ thị cũng không cho ta thấy được insight nào quan trọng, vì vậy ta sẽ xét tiếp có tầm khoảng bao nhiêu phần trăm node sẽ nằm gần bậc trung bình.

```
Percentage of node near average degree: 31.2206%
```

Với khoảng 31.2% nằm gần bậc trung bình +- 10 degree. Điều này cũng cho thấy sự chênh lệch khá lớn giữa các đỉnh có bậc nhỏ chiếm rất nhiều và node có bậc cao chiếm ít.

Tiếp theo ta xem xét các node center, là các node có độ lệch tâm là nhỏ nhất và xem các node center có liên hệ và nằm trong 100 node có bậc cao nhất không.

```
In [159]: center = nx.center(G)
          center
Out[159]: ['567']

In [160]: any(item in center for item in sorted([n for n, d in G.degree()], rev
Out[160]: False
```

Đồ thị chỉ có 1 node center là node 567 và node này không có liên hệ với 100 node có bậc cao nhất.

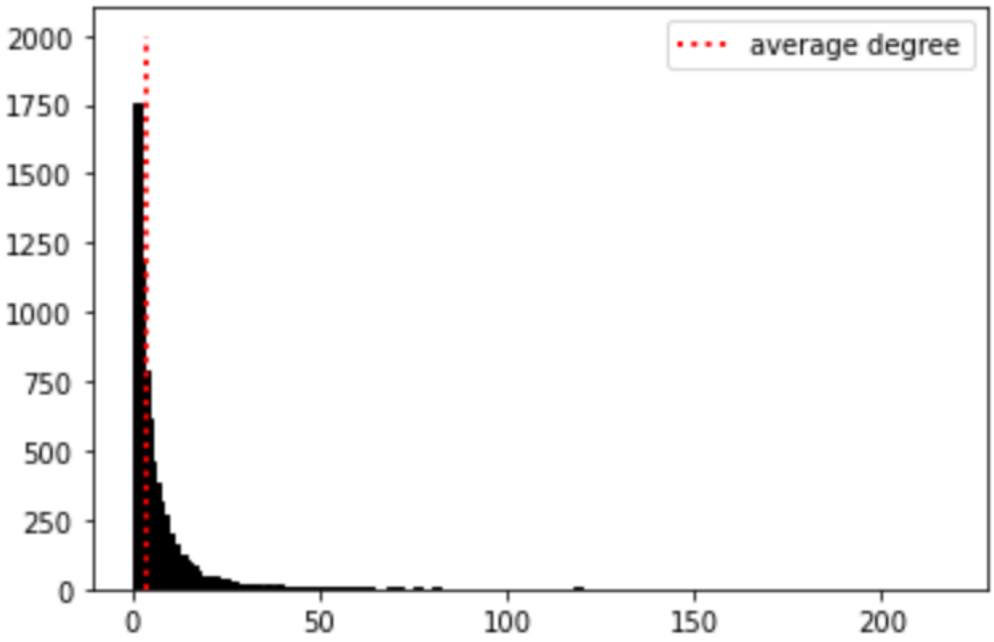
2. Bộ dữ liệu lastFM.

a. Link tham khảo.

https://snap.stanford.edu/data/feather-lastfm-social.html.

b. Thông tin mô tả cơ bản.

Bộ dữ liệu được lấy từ API của mạng xã hội âm nhạc LastFM của người Châu Á và được lấy và tháng 3 năm 2020. Bộ dữ liệu cho thấy mối quan hệ giữa các artist và follower.
Bộ dữ liệu bao gồm 7624 đỉnh và 27806 cạnh với bậc trung bình là 3.64. Thể hiện đúng bản chất của mô tả dữ liệu bộ dữ liệu này là mối quan hệ của những người artist và follower. Tiếp theo là phân phối bậc của bộ dữ liệu này.




```
In [22]: nx.diameter(G)
Out[22]: 15
```

Đường kính là 15 cho thấy độ trải rộng của bộ dữ liệu này là rất lớn.

```
Percentage of node near average degree: 86.4113%
```

Tầm có 86% đỉnh nằm xung quanh bậc trung bình với mức chênh lệch là 10.

```
In [25]: center = nx.center(G)
         center
Out[25]: [5454, 2892, 5646, 7199, 7030, 4309, 1281, 1444, 4119]

In [26]: any(item in center for item in sorted([n for n, d in G.de
Out[26]: False
```

Với các điểm trung tâm nhưng lại không có điểm nào nằm trong top 100 điểm có bậc cao nhất. Điều này đồng nghĩa với việc các node trung tâm của đồ thị không có mối liên hệ gì với các node có bậc cao.

III. Các thước đo lường của đồ thị (metrics measure).

1. Các định nghĩa về một số các giá trị đo lường.

Betweenness centrality: độ đo tầm ảnh hưởng của một node trong graph. Với giá trị càng cao thì mức ảnh hưởng càng cao. Ví dụ các node khác ở cluster này muốn liên lạc với đường đi ngắn nhất với cluster khác thì phải thông qua node có betweenness cao này

Closeness: tính toán về khoảng cách ngắn nhất của một node đến các node còn lại mà nó có kết nối. Closeness càng cao cho thấy node đó nổi tiếng và liên lạc được với nhiều node khác nhau.

Eigenvector: Centrality của 1 node là tổ hợp tuyến tính của các centrality lân cận. Engenvalue càng cao cho thấy các nhóm lân cận của node đó sẽ cao theo.

Edge betweeness centrality: với mỗi cặp node trong network thì có bao nhiêu cặp kết nối lại với nhau ngắn nhất mà phải thông qua cạnh đó. Với cạnh có giá trị càng cao thì cho thấy tần số nó xuất hiện trong đường đi ngắn nhất từ cụm này tới cụm khác nhiều hơn.

IV. Tham khảo.

https://neo4j.com/docs/graph-data-science/current/algorithms/label-propagation/

https://www.youtube.com/watch?v=Gkk5zx9hA_k

https://github.com/gephi/gephi/wiki/Datasets

https://www.clear.rice.edu/comp140/labs/04/

https://snap.stanford.edu/data/