

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA HỆ THỐNG THÔNG TIN**

-----○○○○-----



**MÔN HỌC: MẠNG XÃ HỘI**

**Đề tài: Phân tích dữ liệu mạng xã hội**

**GVHD :** ThS. Nguyễn Thị Minh Phụng

**Lớp:** IS353.M11.HTCL

**Sinh viên thực hiện:**

Đoàn Thục Quyên - 18521320

Thành phố Hồ Chí Minh, 2020

## This image shows a full page of primary-ruled notebook paper. It features multiple sets of horizontal lines across the entire page. Each set consists of three lines: a solid top line, a dashed middle line, and a solid bottom line. The lines are evenly spaced and extend from the left margin to the right edge of the page. There are no margins, text, or other markings present.

<b>I. GIỚI THIỆU .....</b>	<b>5</b>
<b>II. XÁC ĐỊNH BÀI TOÁN.....</b>	<b>6</b>
<b>III. DỮ LIỆU.....</b>	<b>6</b>
1. Nguồn dữ liệu: .....	6
2. Nội dung dữ liệu .....	6
<b>IV. CÁC KIẾN THỨC LIÊN QUAN .....</b>	<b>6</b>
1. GIẢI THÍCH CÁC ĐỘ ĐO CẦN THIẾT TRONG ĐỒ THỊ .....	6
1.1. Degree Centrality.....	7
1.2. Closeness Centrality .....	7
1.3. Harmonic Centrality .....	7
1.4. Betweenness Centrality .....	7
1.5. Eigenvector Centrality .....	8
1.6. Katz centrality.....	8
1.7. PageRank centrality .....	9
2. GIẢI THÍCH CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG .....	10
2.1. Louvain.....	10
2.2. Girvan Newman.....	11
2.3. Kmeans .....	12
<b>V. XỬ LÝ DỮ LIỆU .....</b>	<b>15</b>
1. TIỀN XỬ LÝ DỮ LIỆU.....	15
2. CHUYỂN DATAFRAME THÀNH ĐỒ THỊ .....	16
2.1. Đồ thị 2 phía .....	16
2.2. Đồ thị 1 phía .....	18
<b>VI. CÀI ĐẶT THUẬT TOÁN VÀ KẾT QUẢ.....</b>	<b>18</b>
1. CÁC ĐỘ ĐO .....	18
1.1. Degree Centrality.....	19
1.2. Closeness Centrality .....	20
1.3. Harmonic Centrality .....	21
1.4. Eigenvector Centrality .....	21
1.5. Pagerank.....	22
1.6. Betweenness Centrality .....	23

<b>2. CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG.....</b>	<b>24</b>
<b>2.1. Louvain.....</b>	<b>24</b>
a. Python.....	24
b. Bằng Gephi.....	28
c. So sánh.....	35
d. Nhận xét.....	36
<b>2.2. K-means.....</b>	<b>36</b>
<b>2.3. Grivan Newman.....</b>	<b>42</b>
a. Python.....	42
b. Gephi.....	46
c. So sánh.....	53
d. Nhận xét.....	54
<b>VII. Kết luận.....</b>	<b>54</b>

## I. GIỚI THIỆU

Kể từ tháng 12/2019, thế giới đã chứng kiến sự hoành hành của đại dịch COVID. Nó đã cướp đi sinh mạng của hàng triệu người trên thế giới. Trong khi hầu hết các hoạt động kinh doanh của ngành F&B cũng như các hoạt động thương mại bị trì trệ do nhu cầu chi tiêu trong dịch của người dân là không cao, việc đến các cửa hàng tiện lợi hay trung tâm thương mại trong khi xã hội đang thực hiện giãn cách xã hội là một hành động mạo hiểm, đánh đổi bằng sức khỏe của bản thân và gia đình. Bất chấp những khó khăn trên, hoạt động thương mại điện tử, cũng như kinh doanh online nói chung vẫn đạt được những bước tăng trưởng nhảy vọt. Để thấy rõ sự tăng trưởng này, chúng ta chỉ cần nhìn vào Amazon – một trong những siêu tập đoàn hoạt động trong ngành thương mại điện tử, lợi nhuận Quý II năm 2021 của Amazon đã đạt con số 7.8 tỷ USD – tăng 48% so với cùng kỳ năm trước. Qua đó ta có thể thấy tiềm năng tăng trưởng của ngành này vẫn còn rất cao. Để có thể tồn tại trong ngành này, công nghệ là một thứ không thể thiếu, tồn tại đồng thời song song với quá trình phát triển của doanh nghiệp. Một công ty truyền thống phải mất hàng chục triệu đồng mỗi tháng để thuê mặt bằng, nhưng lại sở hữu con số doanh thu chỉ bằng khoảng 1/10 so với một công ty sử dụng giao dịch trực tuyến làm nền tảng cốt lõi, công ty ấy lại không phải tốn chi phí cho mặt bằng – thứ quyết định phần lớn đến giá cả hàng hóa nếu công ty muốn có lợi nhuận. Từ đây, ta có thể thấy rằng công nghệ quyết định thị phần của bạn trong miếng bánh béo bở mang tên thị trường. Để công nghệ trở nên hữu ích và thông minh hơn, những hệ thống AI dự đoán nhu cầu mua sắm của khách hàng, xác định tệp khách hàng mà công ty cần nhắm tới, sẽ giúp công ty giảm tối đa chi phí cho việc marketing, tiếp cận khách hàng. Giúp khách hàng dễ dàng lựa chọn được sản phẩm mà họ cần. Trong khuôn khổ đề án này, em sẽ nghiên cứu, phân tích một bộ dữ liệu bán hàng tạp hóa, qua đó hiểu được nhu cầu mua sắm của khách hàng.

## II. XÁC ĐỊNH BÀI TOÁN

**Input:** Tập dữ liệu ban đầu trên nguồn dữ liệu Kaggle đã qua tiền xử lý dữ liệu

**Output:** Đưa ra độ đo, đưa ra cộng đồng phục vụ cho việc phân tích mạng xã hội “Groceries Market Basket”

## III. DỮ LIỆU

### 1. Nguồn dữ liệu:

- Nguồn: <https://www.kaggle.com/irfanasrullah/groceries>
- File cần tải: groceries - groceries.csv

### 2. Nội dung dữ liệu

Đây là tập dữ liệu hàng tạp hóa với danh sách các mặt hàng mà khách hàng đã mua. Tập dữ liệu chứa 9835 giao dịch của khách hàng mua các mặt hàng của hàng tạp hóa và 169 sản phẩm duy nhất của cửa hàng.

Dữ liệu file “groceries - groceries.csv” gồm:

- 9835 dòng dữ liệu
- 33 cột thuộc tính

Item(s)	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	...	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29	Item 30
8	meat	turkey	citrus fruit	tropical fruit	herbs	other vegetables	frozen meals	shopping bags	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	frankfurter	processed cheese	domestic eggs	white bread	brown bread	ketchup	soda	hygiene articles	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	onions	soda	chewing gum	chocolate	newspapers	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	seasonal products	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	citrus fruit	tropical fruit	whole milk	curd	dessert	mayonnaise	domestic eggs	margarine	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Giải thích file dữ liệu:

- Mỗi dòng là một chi tiết hóa đơn, chứa các mặt hàng khách hàng đã mua và số lượng mặt hàng.
- Cột Item(s): thể hiện tổng mặt hàng trong hóa đơn đó.
- Các cột Item 1, Item 2, ... Item 32: chứa tên mặt hàng khách hàng đã mua. Có thể hiểu như sau, nếu cột thuộc tính Item(s) có giá trị bằng 8 thì cột Item 1, Item 2, ... Item 8 khác rỗng và có chứa tên mặt hàng có trong hóa đơn.

## IV. CÁC KIẾN THỨC LIÊN QUAN

### 1. GIẢI THÍCH CÁC ĐỘ ĐO CẦN THIẾT TRONG ĐỒ THỊ

Trong lý thuyết đồ thị và phân tích mạng xã hội, các chỉ số đo độ chỉ ra được vị trí trong bảng xếp hạng, mức độ quang trọng, ảnh hưởng cũng như vai trò của các node trong mạng.

### 1.1.Degree Centrality

Degree centrality là một trong những số liệu chúng ta có thể sử dụng để đánh giá mức độ quan trọng của một node. Degree centrality được định nghĩa bằng số lượng node lân cận của nó chia cho số lượng node lân cận nhiều nhất chúng ta có thể có.

$$\text{Degree Centrality}(x) = \frac{\text{Số lượng node lân cận của node } x}{\text{Tất cả node lân cận node } x \text{ có thể có}}$$

Nếu đồ thị có hướng, thì được chia làm hai độ đo degree centrality là indegree và outdegree. Indegree là số lượng cạnh nối trực tiếp tới node và outdegree là số lượng cạnh mà node nối đến các node khác. Trong trường hợp có chiều thì degree centrality bằng tổng của indegree và outdegree.

Giá trị của chỉ số này nằm trong khoảng từ 0.00 đến 1.00 và giá trị càng gần tới 1.00 thì tính trung tâm trực tiếp của node đó càng lớn, tức là càng nằm ở vị trí trung tâm của mạng.

### 1.2.Closeness Centrality

Trong đồ thị có kết nối (connected graph), độ đo closeness centrality của một node là trung bình cộng đường đi ngắn nhất giữa node đó và các node khác trong mạng. Vì vậy, một node càng gần trung tâm thì nó càng gần các node khác. Closeness centrality được tính bằng cách nghịch đảo của tổng khoảng cách từ node x đến tất cả các node khác. Nghĩa là các node càng gần trung tâm thì có tổng khoảng cách đến các node khác càng nhỏ thì độ đo closeness centrality càng lớn.

$$\text{Closeness Centrality}(x) = \frac{1}{\sum_y d(y, x)}$$

Trong đó,  $d(y, x)$  là khoảng cách/đường đi ngắn nhất từ node x đến node y.

Nhưng độ đo này chỉ hoạt động đối với những mạng kết nối chặt chẽ với nhau (strongly connected). Nếu mạng không kết nối chặt chẽ thì ta dùng độ đo Harmonic centrality.

### 1.3.Harmonic Centrality

Trong đồ thị không cần thiết phải kết nối, Harmonic Centrality bằng tổng nghịch đảo khoảng cách đến các node khác.

$$\text{Harmonic Centrality}(x) = \sum_{x \neq y} \frac{1}{d(y, x)}$$

Trong đó,  $\frac{1}{d(y, x)} = 0$  khi không có đường nối từ node x đến node y.

Harmonic được đề xuất bởi Marchiori và Latora vào năm 2000, được hoàn thiện bởi Dekker và năm 2005, từng được đặt tên là "valued centrality" bởi Rochat vào năm 2009.

### 1.4.Betweenness Centrality

Betweenness Centrality (đây không phải là edge betweenness) là độ đo centrality của đồ thị dựa vào đường đi ngắn nhất. Với mỗi cặp đỉnh trong đồ thị liên thông, tồn tại ít nhất một đường đi ngắn nhất giữa các đỉnh sao cho số cạnh mà nằm trên đường đi qua (đối với đồ thị không có trọng số) hoặc tổng trọng số của các cạnh (đối với đồ thị có trọng số) là nhỏ nhất. Độ đo betweenness centrality cho mỗi đỉnh là số lượng đường đi ngắn nhất đi qua đỉnh đó.

Betweenness centrality được xem như là thước đo chung về tính tập trung: nó được áp dụng rộng rãi cho một số vấn đề lý thuyết mạng, bao gồm những vấn đề liên quan đến mạng xã hội, sinh học, giao thông, ... Freeman là người đầu tiên đưa ra định nghĩa chính thức về betweenness centrality. Trong quan niệm của ông, những đỉnh mà có xác suất cao được chọn làm điểm trên đường đi ngắn nhất giữa 2 đỉnh được chọn ngẫu nhiên thì sẽ có độ đo betweenness cao.

Có thể tính bằng công thức:

$$\text{Betweenness Centrality}(x) = \sum_{y,z \neq x, \sigma_{yz} \neq 0} \frac{\sigma_{yz}(x)}{\sigma_{yz}}$$

Trong đó:

- $\sigma_{yz}$  : số lượng đường đi ngắn nhất đi từ y đến z
- $\sigma_{yz}(x)$ : số lượng đường đi ngắn nhất đi từ y đến z có đi qua node x.

### 1.5.Eigenvector Centrality

Eigenvector centrality là độ đo mức độ ảnh hưởng của một node trong một mạng. Mỗi node sẽ được gán một điểm số tương đối bằng nhau

Một điểm số tương đối sẽ được gán cho tất cả các node trong mạng. Sau đó, nó sẽ cho điểm tín nhiệm cho những node lân cận.

$$c_{eig}(x) = \frac{1}{\lambda} \sum_{y \rightarrow x} e_{eig}(y)$$

$\lambda$ : eigenvalue của ma trận liên hệ A (là hằng số)

$c_{eig}$ : eigenvector của ma trận liên hệ A

Eigencentality là eigenvector tương ứng với eigenvalue ( $\lambda$ ) của A:  $AX = \lambda X$

Chỉ áp dụng với đồ thị liên thông mạnh

### 1.6.Katz centrality

Độ đo ảnh hưởng bằng cách tính tổng số lần đi qua giữa một cặp node.

$$c_{katz}(x) = \beta \sum_{k=0}^{\infty} \sum_{x \rightarrow y} \alpha^k \left( A^k \right)_{xy}$$

Total number of walks  
of length k between x, y



$\alpha < 1$  : Những đường đi dài có trọng số nhỏ

$\alpha$  là hệ số suy giảm trong khoảng  $(0, \frac{1}{\lambda})$ , trong đó  $\lambda$  là eigenvalue lớn nhất của A

$\beta$  là cấp cho một số node có nhiều đặc quyền hơn

Những đường dài có trọng số nhỏ hơn những đường ngắn

$(A^k)_{xy}$ : Tổng số lượng đường đi có độ dài k giữa 2 node x và y

Phù hợp với đồ thị vòng có hướng (directed acyclic graphs)

### 1.7. PageRank centrality

PageRank hay Ranking viết tắt là PR tạm dịch là thứ hạng trang. Khi nói đến PageRank người ta thường nghĩ đến ngay Google PageRank. Đó là một hệ thống xếp hạng trang Web của các máy tìm kiếm nhằm sắp xếp thứ tự ưu tiên đường dẫn URL trong trang kết quả tìm kiếm.

PageRank được phát triển tại đại học Stanford bởi Larry Page (cũng bởi vậy mà có tên PageRank) và sau đó bởi Sergey Brin như một phần dự án công cụ tìm kiếm mới.

Theo Google một cách tóm lược thì PageRank chỉ được đánh giá từ hệ thống liên kết đường dẫn. Trang của bạn càng nhận nhiều liên kết ( phải là dofollow ) trở đến thì mức độ quan trọng trang của bạn càng tăng.

Công thức PageRank có dạng như sau:

$$PR(p_i) = \frac{1-d}{n} + d \sum_{p_j \in M(i)} \frac{PR(p_j)}{L(j)}$$

Trong đó:

- d: hằng số Google quy định. Thông thường,  $d = 0.85$ .

- $PR(p_j)$ : PageRank của các đỉnh đi vào đỉnh  $i$ .
- $L(j)$ : số link out của các đỉnh đi vào đỉnh  $i$ .

## 2. GIẢI THÍCH CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG

Bài toán phát hiện cộng đồng tập trung vào việc từ một đồ thị mạng xã hội, tìm ra những cụm, nhóm cộng đồng có mối liên hệ chặt chẽ với nhau. Qua trực quan có thể dễ dàng tìm ra những nhóm cộng đồng có độ tập trung cao

### 2.1.Louvain

Độ đo đơn thể  $M$  (Modularity  $M$ ) được sử dụng để đánh giá chất lượng thuật toán phát hiện cộng đồng, độ đo đơn thể  $M$  có giá trị càng lớn thể hiện độ chính xác của thuật toán càng cao, chất lượng việc phát hiện cộng đồng được đánh giá là tốt.

Thuật toán Louvain là thuật toán đầu tiên được đề xuất nhằm tối ưu hóa độ đo đơn thể. Ý tưởng của kĩ thuật như sau, ban đầu mỗi nút thuộc về một mô đun riêng biệt (chính nó), sau đó chúng được hợp nhất dựa trên mức độ tăng các mô đun và quá trình này lặp đi lặp lại cho đến khi thu được giá trị mô đun lớn nhất.

Ta có hiểu thuật toán gồm 2 giai đoạn chính: Tối ưu hóa mô-đun và Tổng hợp cộng đồng

#### Tối ưu hóa mô đun:

Louvain sẽ sắp xếp mỗi nút vào cộng đồng của chỉ gồm chính nó. Sau đó, từng nút một sẽ chuyển sang các cộng đồng khác nhau  $C$  cho đến khi  $\Delta M$  không có sự thay đổi đáng kể.

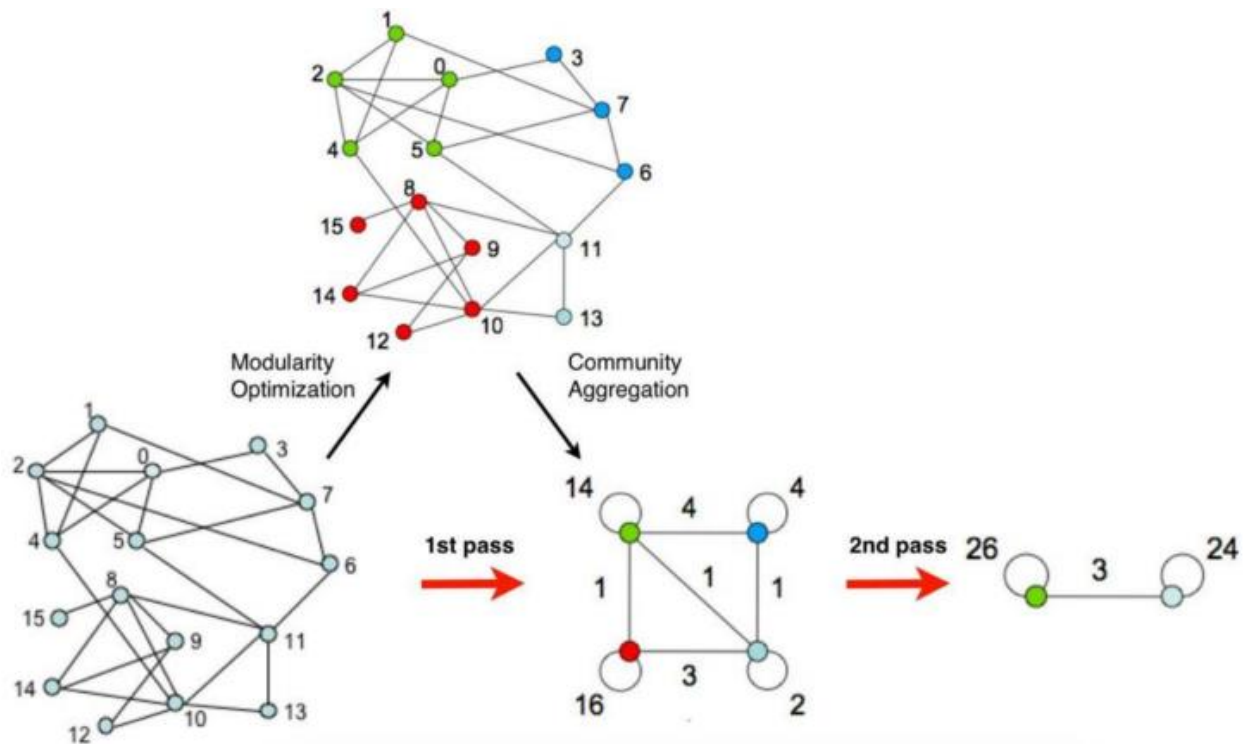
$$\Delta M = \left[ \frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left( \frac{\Sigma_{tot} + k_i}{2m} \right)^2 \right] - \left[ \frac{\Sigma_{in}}{2m} - \left( \frac{\Sigma_{tot}}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right]$$

Trong đó:

- $\Sigma_{in}$  là tổng trọng số của các liên kết bên trong cộng đồng  $C$ ,
- $\Sigma_{tot}$  tổng trọng số của tất cả các liên kết tới các nút trong  $C$ ,
- $k_i$  tổng trọng số của tất cả các liên kết ngẫu nhiên trong nút  $i$ ,
- $k_{i,in}$  là tổng trọng số của các liên kết từ nút  $i$  đến các nút trong cộng đồng  $C$
- $m$  là tổng trọng số của tất cả các cạnh trong đồ thị.

#### Tổng hợp cộng đồng:

Sau khi kết thúc bước đầu tiên, tất cả các nút thuộc cùng một cộng đồng được hợp nhất thành một nút khổng lồ duy nhất. Các liên kết kết nối các nút khổng lồ là tổng các liên kết trước đó các nút ở cộng này kết nối với các nút ở cộng đồng khác. Bước này cũng tạo ra các liên kết đến chính nó là tổng của tất cả các liên kết bên trong một cộng đồng trước khi được thu gọn vào một nút.



Tóm tắt thuật toán:

**Bước 1.** Đầu tiên, mỗi nút trong mạng được gán vào một cộng đồng khác nhau, vì vậy trong lần đầu phân vùng, số lượng cộng đồng bằng số nút của mạng.

**Bước 2.** Đối với mỗi nút, xem xét các nút lân cận của nó và đánh giá giá trị, tính mô đun sau khi loại bỏ một nút khỏi cộng đồng của nút đó và đặt nút đó vào một trong những cộng đồng lân cận của nó. Nếu  $\Delta M$  dương, nút vẫn ở trong cộng đồng ban đầu của nó, ngược lại, nút được đặt trong cộng đồng cập nhật.

**Bước 3.** Lặp lại Bước 2 đến khi cộng đồng của tất cả các nút không còn thay đổi.

**Bước 4.** Xây dựng một đồ thị mới và mỗi nút đại diện cho một cộng đồng – kết quả phân vùng của Bước 3. Thực hiện lại Bước 2 và Bước 3 liên tục cho đến khi đạt được giá trị mô đun lớn nhất

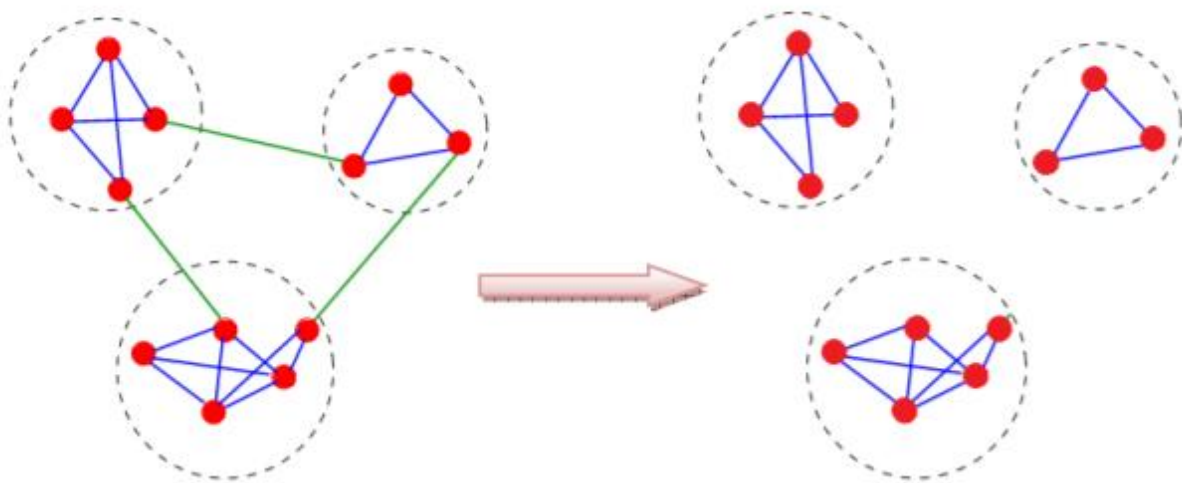
## 2.2. Girvan Newman

Thuật toán Girvan-Newman phương pháp phát hiện cộng đồng bằng thuật toán phân. Thay vì cố gắng để xây dựng một biện pháp tìm cạnh trung tâm của cộng đồng, chúng ta đi tìm những cạnh ít trung tâm nhất, cạnh đó được gọi tên là cạnh giữa cộng đồng(edge betweenness). Thuật toán này dựa trên quan niệm cho rằng khi các cộng đồng được gắn kết với nhau thì đường đi giữa cộng đồng này đến cộng đồng khác sẽ đi qua các cạnh nối giữa các cộng đồng với tần suất cao. Mục đích chính của thuật toán là tìm những cạnh nối đó và loại bỏ dần dần các cạnh nối từ đồ thị ban đầu. Khi đó, các cộng đồng trong mạng sẽ bị ngắt kết nối với nhau, ta có thể xác định được cách phân vùng đồ thị thành các phần nhỏ riêng rẽ.

Thuật toán lần đầu tiên được đề xuất bởi Freeman. Theo Freeman, các cạnh được coi là cạnh có số lượng con đường ngắn nhất giữa các cặp đỉnh khác nhau chạy qua nó. Cạnh nối có ảnh hưởng rất lớn đến dòng chảy của thông tin giữa các nút khác, đặc biệt là trong trường hợp thông tin lưu truyền trong mạng chủ yếu theo con đường ngắn nhất.

Để tìm các cạnh trong mạng nối hai đỉnh thuộc hai cộng đồng khác nhau - là các cạnh có độ trung gian cao, và xác định độ đo trung gian này bằng cách tính số đường đi ngắn nhất giữa các cặp đỉnh mà có qua nó. Với đồ thị có trọng số, độ đo trung gian của cạnh có trọng số đơn giản được tính bằng độ đo trung gian của cạnh không có trọng số chia cho trọng số của cạnh đó. Nếu một mạng lưới bao gồm các cộng đồng chỉ được liên kết yếu bằng một nhóm cạnh, thì tất cả các đường đi ngắn nhất giữa các cộng đồng khác nhau sẽ phải đi dọc theo một trong số ít các cạnh thuộc nhóm cạnh đó. Vì vậy, các cạnh kết nối các cộng đồng sẽ là cạnh có độ đo trung gian cao. Bằng cách loại bỏ các cạnh, thuật toán Girvan-Newman tách được thành các nhóm riêng biệt. Thuật toán được thực hiện theo các bước sau:

1. Tính độ đo trung gian (edge betweenness) cho tất cả các cạnh trong mạng.
2. Hủy bỏ các cạnh có độ trung gian cao nhất.
3. Tính lại độ trung gian cho tất cả các cạnh bị ảnh hưởng theo các cạnh đã loại bỏ.
4. Lặp lại từ bước 2 cho đến khi không còn các cạnh trung gian.



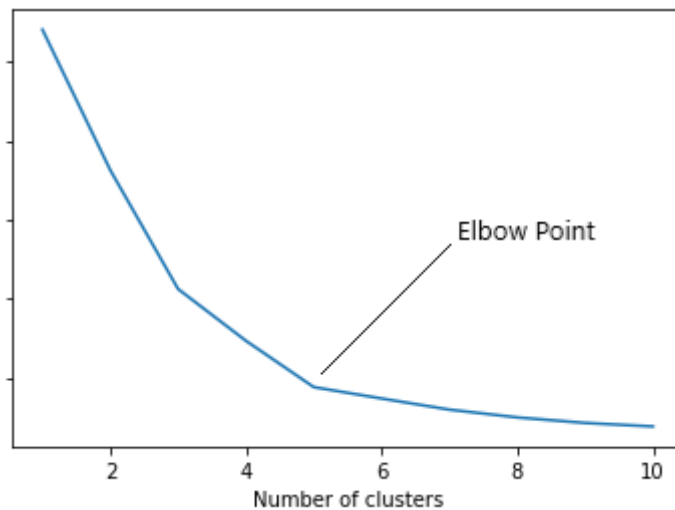
Hình 1: Ví dụ về phát hiện cộng đồng sử dụng Girvan-Newman

## 2.3.Kmeans

Khi thực hiện gom cụm bằng Kmeans, chúng ta cần xác định được k cụm. Nhưng câu hỏi đặt ra là số lượng cụm bao nhiêu là hợp lí, chúng ta có thể đo lường được độ chính xác của nó hay không. Câu trả lời cho câu hỏi này là chúng ta có 2 phương pháp:

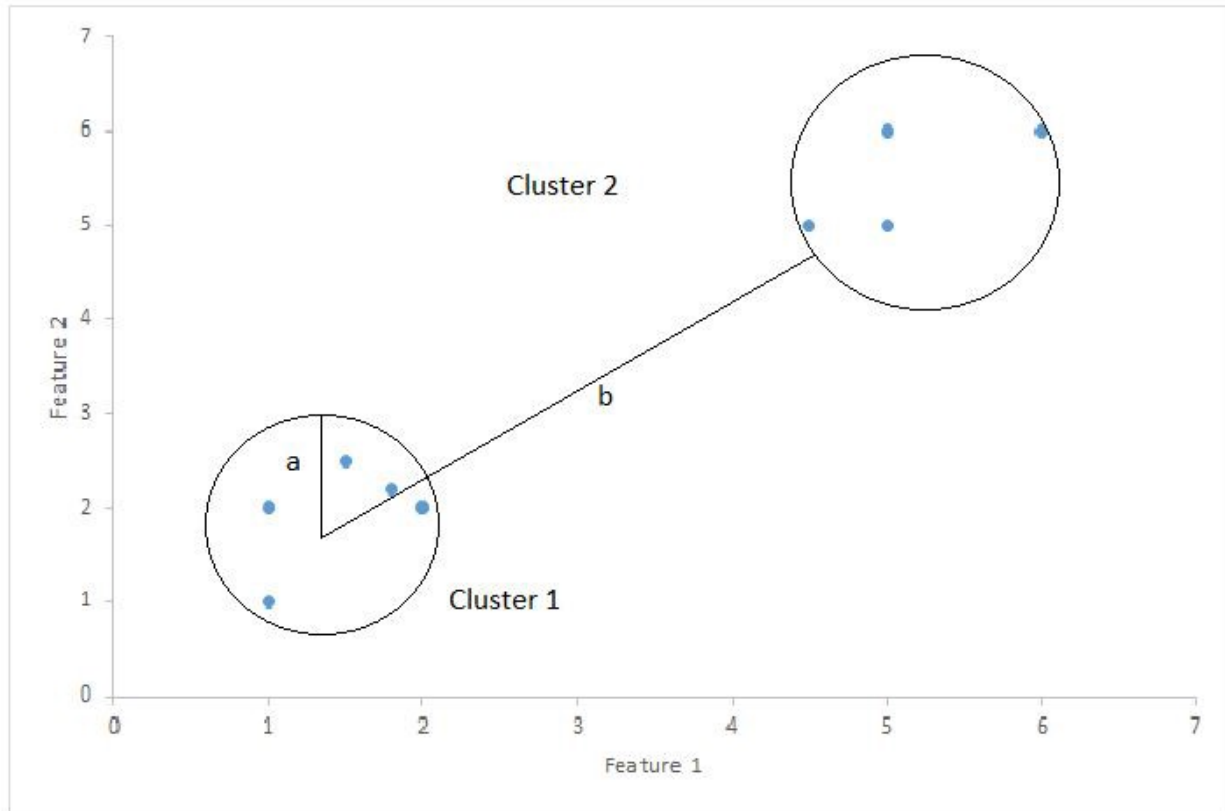
- Phương pháp khuỷu tay (Elbow)
- Phương pháp tính trung bình hệ số Silhouette

**Phương pháp Elbow** là một phương pháp thực nghiệm để tìm số lượng cụm tối ưu cho một tập dữ liệu. Trong phương pháp này, chúng ta chọn một phạm vi các giá trị ứng viên của k (số lượng cụm) để vẽ biểu đồ đường và chọn phần khuỷu của đường cong làm số lượng cụm sẽ sử dụng.



**Hệ số Silhouette** là số liệu được tính toán để đo mức độ tốt của kỹ thuật phân cụm. Giá trị của nó nằm trong khoảng từ -1 đến 1. Khi hệ số Silhouette càng gần:

- “1”: có nghĩa là các cụm càng cách xa nhau và phân biệt rõ ràng.
- “0”: có nghĩa là các cụm càng ít sự khác biệt hoặc chúng ta có thể hiểu khoảng cách giữa các cụm là không đáng kể.
- “-1”: có nghĩa là các cụm bị gán sai cách.



Cách tính:

$$\text{Silhouette Score} = (b-a)/\max(a,b)$$

Trong đó:

- + a: Khoảng cách trung bình trong cụm, tức là khoảng cách giữa mỗi điểm trong cụm
- + b: Khoảng cách trung bình giữa các cụm tức là khoảng cách trung bình giữa tất cả các cụm với nhau.

Cách thực hiện tính hệ số Silhouette trong python:

```
from sklearn.metrics import silhouette_samples, silhouette_score
silhouette_avg = silhouette_score(X, cluster_labels)
```

**Thuật toán K-means clustering (phân cụm K-means)** là thuật toán Unsupervised learning. Trong thuật toán K-means clustering, mục đích của chúng ta là làm thế nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

Ý tưởng đơn giản nhất về cluster (cụm) là tập hợp các điểm ở gần nhau trong một không gian nào đó (không gian này có thể có 2 hoặc nhiều hơn 2 chiều).

Tóm tắt thuật toán

**Đầu vào:** Dữ liệu X và số lượng cluster cần tìm k.

**Đầu ra:** Các vector tâm cụm (center) M và label cho từng điểm dữ liệu Y.

- 2.4. Chọn k điểm bất kỳ làm các center ban đầu.
- 2.5. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.
- 2.6. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
- 2.7. Cập nhật lại các center cho từng cluster bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
- 2.8. Quay lại bước 2.

## V. XỬ LÝ DỮ LIỆU

### 1. TIỀN XỬ LÝ DỮ LIỆU

**Thêm cột Order ID cho tập dữ liệu bằng file Excel:**

- Thêm vào bên trái một cột mới, đặt tên là “OrderID”
- Thêm các giá trị từ O0001 đến O9835 theo thứ tự từ trên xuống

**Lấy các dữ liệu cần phân tích đưa vào các biến trên Jupyter notenook:**

- Lấy 200 dòng ngẫu nhiên trong tập dữ liệu, đặt lại các giá trị index
- Xuất ra file 200 dòng để phục vụ cho việc backup
- Với mỗi dòng, dùng vòng lặp đi qua các cột để lấy tên các mặt hàng và thêm với một cái danh sách mặt hàng.
- Xóa các giá trị trùng trong danh sách đó, ta được danh sách các mặt hàng(item)
- Lấy cột 0 của tập dữ liệu ban đầu, ta được danh sách các mã đơn hàng(order)
- Sau cùng kiểm tra số lượng mặt hàng và đơn hàng

```
10 item_list = get_item(df_order)
11 item = item_list.drop_duplicates().reset_index()['Item']
12
13 print("Số lượng sản phẩm: ", item.nunique())
14 print("Số lượng hóa đơn:", order.nunique())
15
```

✓ 1.8s

Số lượng sản phẩm: 131

Số lượng hóa đơn: 200

Dữ liệu của ta sẽ phân tích trên 200 hóa đơn và 131 loại mặt hàng khác nhau.

## 2. CHUYỂN DATAFRAME THÀNH ĐỒ THỊ

### 2.1. Đồ thị 2 phía

- Đồ thị này nhận 2 loại node: các mã đơn hàng và các mặt hàng trong hóa đơn. Cạnh của chúng là mối quan hệ giữa việc một đơn hàng có mua một mặt hàng.
- Để trực quan hóa đồ thị:
  - + Thêm danh sách các node: danh sách các node mã đơn hàng nằm bên trái và danh sách các mặt hàng nằm bên phải.
  - + Thêm cạnh vào đồ thị

```
import networkx as nx
B = nx.Graph()

B.add_nodes_from(item, bipartite=0)
B.add_nodes_from(order, bipartite=1)

nlink = 0
for index in df_order.index:
    nitem = df_order['Item(s)'][index]
    nlink = nlink + nitem
    for i in range(nitem):
        B.add_edge(df_order.iloc[index, 0], df_order.iloc[index, 2+i],
weight=1)

print("Số lượng cạnh của đồ thị 2 phía: ", nlink)
```

#### Kết quả:

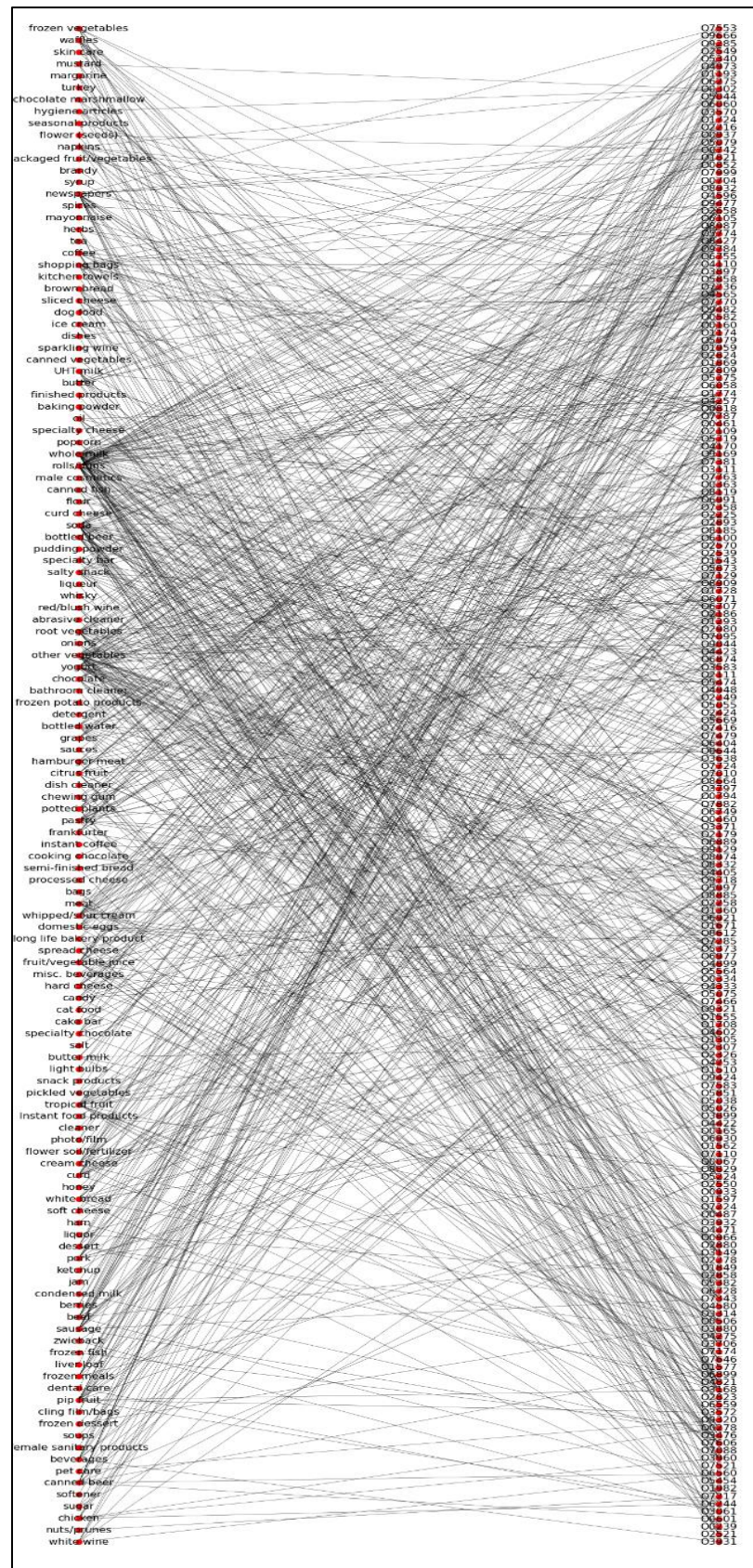
Số lượng cạnh của đồ thị 2 phía: 944

Đồ thị này gồm:

- + 200 node hóa đơn
- + 131 mặt hàng
- + 944 cạnh giữa chúng. Nếu hóa đơn có 3 sản phẩm thì sẽ có 3 cạnh nối đến 3 sản phẩm đó.

- Ta thu được đồ thị 2 phía như sau:





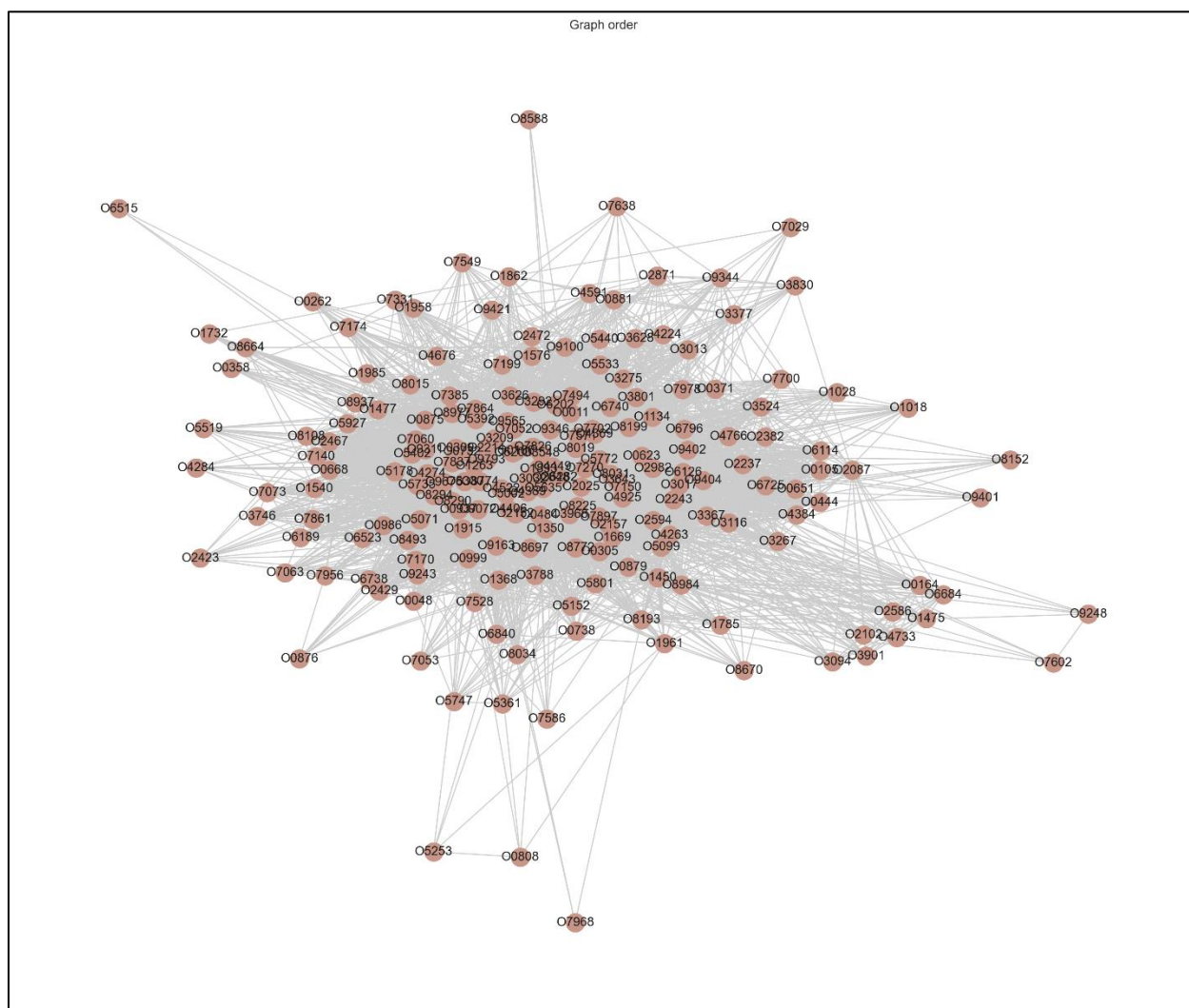
Nhìn vào đồ thị, ta có thể thấy rõ các mặt hàng được mua nhiều bởi các hóa đơn ( các mặt hàng có nhiều cạnh liên kết tới) và hóa đơn nào mua nhiều sản phẩm ( có nhiều cạnh liên kết đến các mặt hàng)

## 2.2. Đồ thị 1 phía

- Từ đồ thị 2 phía, ta dùng hàm `bipartite.weighted_projected_graph()` của network để tạo đồ thị 1 phía.

```
from networkx.algorithms import bipartite
G = bipartite.weighted_projected_graph(B, order)
```

- Ta có được đồ thị 1 phía gồm: 200 node và 5678 cạnh. Trong đó:
  - + Node: là mã hóa đơn
  - + Edge: Hai hóa đơn cùng một mặt hàng thì sẽ hình thành 1 cạnh giữa 2 hóa đơn đó.



## VI. CÀI ĐẶT THUẬT TOÁN VÀ KẾT QUẢ

### 1. CÁC ĐỘ ĐO

### 1.1.Degree Centrality

Jupyter notebook	Gephi																																																										
<p>➤ Cài đặt</p> <pre>degree = nx.degree(G)</pre> <p>➤ Kết quả</p> <table> <thead> <tr> <th>OrderID</th><th>degree</th></tr> </thead> <tbody> <tr><td>O4449</td><td>147</td></tr> <tr><td>O6282</td><td>140</td></tr> <tr><td>O0186</td><td>136</td></tr> <tr><td>O5635</td><td>135</td></tr> <tr><td>O3032</td><td>135</td></tr> <tr><td>O1263</td><td>133</td></tr> <tr><td>O7270</td><td>124</td></tr> <tr><td>O2848</td><td>118</td></tr> <tr><td>O4369</td><td>117</td></tr> <tr><td>O4522</td><td>116</td></tr> </tbody> </table>	OrderID	degree	O4449	147	O6282	140	O0186	136	O5635	135	O3032	135	O1263	133	O7270	124	O2848	118	O4369	117	O4522	116	<p>➤ Kết quả</p> <table> <thead> <tr> <th>Id</th><th>Degree</th></tr> </thead> <tbody> <tr><td>O4449</td><td>147</td></tr> <tr><td>O6282</td><td>140</td></tr> <tr><td>O0186</td><td>136</td></tr> <tr><td>O5635</td><td>135</td></tr> <tr><td>O3032</td><td>135</td></tr> <tr><td>O1263</td><td>133</td></tr> <tr><td>O7270</td><td>124</td></tr> <tr><td>O2848</td><td>118</td></tr> <tr><td>O4369</td><td>117</td></tr> <tr><td>O4522</td><td>116</td></tr> <tr><td>O5002</td><td>115</td></tr> <tr><td>O2214</td><td>111</td></tr> <tr><td>O3548</td><td>111</td></tr> <tr><td>O9793</td><td>110</td></tr> <tr><td>O4389</td><td>107</td></tr> <tr><td>O1350</td><td>106</td></tr> <tr><td>O3209</td><td>105</td></tr> </tbody> </table>	Id	Degree	O4449	147	O6282	140	O0186	136	O5635	135	O3032	135	O1263	133	O7270	124	O2848	118	O4369	117	O4522	116	O5002	115	O2214	111	O3548	111	O9793	110	O4389	107	O1350	106	O3209	105
OrderID	degree																																																										
O4449	147																																																										
O6282	140																																																										
O0186	136																																																										
O5635	135																																																										
O3032	135																																																										
O1263	133																																																										
O7270	124																																																										
O2848	118																																																										
O4369	117																																																										
O4522	116																																																										
Id	Degree																																																										
O4449	147																																																										
O6282	140																																																										
O0186	136																																																										
O5635	135																																																										
O3032	135																																																										
O1263	133																																																										
O7270	124																																																										
O2848	118																																																										
O4369	117																																																										
O4522	116																																																										
O5002	115																																																										
O2214	111																																																										
O3548	111																																																										
O9793	110																																																										
O4389	107																																																										
O1350	106																																																										
O3209	105																																																										

#### Nhận xét:

Degree chỉ định mức độ quan trọng dựa trên số lượng liên kết trực tiếp tới nút. Node có mã hóa đơn O4449 có degree 147 chứng tỏ có 147 liên kết tới các hóa đơn khác. Node này có degree cao nhất, nghĩa là một trong những node có kết nối mạnh trong mạng. Ta có thể suy ra những mặt hàng của hóa đơn này cũng được mua nhiều ở các hóa đơn khác, hay hóa đơn này mua hầu hết các mặt hàng mà hóa đơn khác mua.

Bảng sau thống kê các món hàng của hóa đơn O4449 và tổng số lượng các món hàng đó của tất cả 200 nodes. Trong đó, có 12/20 sản phẩm của hóa đơn là các sản phẩm nằm trong top 30 sản phẩm bán chạy của cửa hàng tạp hóa

frankfurter	13
turkey	6
citrus fruit	18
whole milk	50
curd	15
frozen potato products	4
rolls/buns	37
white bread	11
pastry	16
semi-finished bread	3
pickled vegetables	4
sweet spreads	1
bottled water	19
soda	39
fruit/vegetable juice	11
bottled beer	17
white wine	9
prosecco	1
chocolate	15
candy	9

## 1.2.Closeness Centrality

Jupyter notebook	Gephi
<p>➤ Cài đặt</p> <pre><code>closeness = nx.closeness centrality(G)</code></pre> <p>➤ Kết quả</p>	<p>➤ Kết quả</p>



OrderID	closeness_centrality	Id	Closeness Centrality
O4449	0.7928286853	O4449	0.792829
O6282	0.7713178295	O6282	0.771318
O0186	0.7595419847	O0186	0.759542
O5635	0.7566539924	O5635	0.756654
O3032	0.7566539924	O3032	0.756654
O1263	0.7509433962	O1263	0.750943
O7270	0.7262773723	O7270	0.726277
O2848	0.7107142857	O2848	0.710714
O4522	0.7056737589	O4369	0.705674
O4369	0.7056737589	O4522	0.705674

#### Nhận xét:

Những hóa đơn có độ closeness cao, thì có tầm ảnh hưởng lớn trong toàn mạng. Mà những hóa đơn càng gần nhau thì càng có nhiều những mặt hàng giống nhau, trong đó những hóa đơn có độ do closeness cao nhất sẽ có các mặt hàng giống với nhiều hóa đơn khác trong mạng.

### 1.3. Harmonic Centrality

Jupyter notebook	Gephi																																												
<p>➤ Cài đặt</p> <pre>sorted(nx.harmonic_centrality(G).items(),       key=lambda x:x[1], reverse=True)</pre> <p>➤ Kết quả</p> <table> <tr> <th>OrderID</th><th>harmonic_centrality</th></tr> <tr><td>O4449</td><td>173.0000000000</td></tr> <tr><td>O6282</td><td>169.5000000000</td></tr> <tr><td>O0186</td><td>167.5000000000</td></tr> <tr><td>O5635</td><td>167.0000000000</td></tr> <tr><td>O3032</td><td>167.0000000000</td></tr> <tr><td>O1263</td><td>166.0000000000</td></tr> <tr><td>O7270</td><td>161.5000000000</td></tr> <tr><td>O2848</td><td>158.5000000000</td></tr> <tr><td>O4369</td><td>157.8333333333</td></tr> <tr><td>O4522</td><td>157.5000000000</td></tr> </table>	OrderID	harmonic_centrality	O4449	173.0000000000	O6282	169.5000000000	O0186	167.5000000000	O5635	167.0000000000	O3032	167.0000000000	O1263	166.0000000000	O7270	161.5000000000	O2848	158.5000000000	O4369	157.8333333333	O4522	157.5000000000	<p>➤ Kết quả</p> <table> <tr> <th>Id</th><th>Harmonic</th></tr> <tr><td>O4449</td><td>0.869347</td></tr> <tr><td>O6282</td><td>0.851759</td></tr> <tr><td>O0186</td><td>0.841709</td></tr> <tr><td>O5635</td><td>0.839196</td></tr> <tr><td>O3032</td><td>0.839196</td></tr> <tr><td>O1263</td><td>0.834171</td></tr> <tr><td>O7270</td><td>0.811558</td></tr> <tr><td>O2848</td><td>0.796482</td></tr> <tr><td>O4369</td><td>0.793132</td></tr> <tr><td>O4522</td><td>0.791457</td></tr> </table>	Id	Harmonic	O4449	0.869347	O6282	0.851759	O0186	0.841709	O5635	0.839196	O3032	0.839196	O1263	0.834171	O7270	0.811558	O2848	0.796482	O4369	0.793132	O4522	0.791457
OrderID	harmonic_centrality																																												
O4449	173.0000000000																																												
O6282	169.5000000000																																												
O0186	167.5000000000																																												
O5635	167.0000000000																																												
O3032	167.0000000000																																												
O1263	166.0000000000																																												
O7270	161.5000000000																																												
O2848	158.5000000000																																												
O4369	157.8333333333																																												
O4522	157.5000000000																																												
Id	Harmonic																																												
O4449	0.869347																																												
O6282	0.851759																																												
O0186	0.841709																																												
O5635	0.839196																																												
O3032	0.839196																																												
O1263	0.834171																																												
O7270	0.811558																																												
O2848	0.796482																																												
O4369	0.793132																																												
O4522	0.791457																																												

**Nhận xét:** tương tự như của Closeness Centrality.

### 1.4. Eigenvector Centrality

Jupyter notebook	Gephi																																												
<p>➤ Cài đặt</p> <pre>sorted(nx.eigenvector_centrality(G).items(),       key=lambda x:x[1], reverse=True)</pre> <p>➤ Kết quả</p> <table> <thead> <tr> <th>OrderID</th><th>eigenvector_centrality</th></tr> </thead> <tbody> <tr><td>O4449</td><td>0.1379939757</td></tr> <tr><td>O5635</td><td>0.1375687508</td></tr> <tr><td>O3032</td><td>0.1355738328</td></tr> <tr><td>O1263</td><td>0.1307544089</td></tr> <tr><td>O6282</td><td>0.1300681576</td></tr> <tr><td>O7270</td><td>0.1285119034</td></tr> <tr><td>O0186</td><td>0.1283726203</td></tr> <tr><td>O2848</td><td>0.1251526050</td></tr> <tr><td>O4522</td><td>0.1229054138</td></tr> <tr><td>O4369</td><td>0.1204853585</td></tr> </tbody> </table>	OrderID	eigenvector_centrality	O4449	0.1379939757	O5635	0.1375687508	O3032	0.1355738328	O1263	0.1307544089	O6282	0.1300681576	O7270	0.1285119034	O0186	0.1283726203	O2848	0.1251526050	O4522	0.1229054138	O4369	0.1204853585	<p>➤ Kết quả</p> <table> <thead> <tr> <th>Id</th><th>Eigenvector ...</th></tr> </thead> <tbody> <tr><td>O4449</td><td>1.0</td></tr> <tr><td>O5635</td><td>0.994534</td></tr> <tr><td>O3032</td><td>0.98008</td></tr> <tr><td>O1263</td><td>0.944938</td></tr> <tr><td>O6282</td><td>0.942965</td></tr> <tr><td>O0186</td><td>0.929531</td></tr> <tr><td>O7270</td><td>0.928948</td></tr> <tr><td>O2848</td><td>0.904029</td></tr> <tr><td>O4522</td><td>0.887279</td></tr> <tr><td>O4369</td><td>0.871284</td></tr> </tbody> </table>	Id	Eigenvector ...	O4449	1.0	O5635	0.994534	O3032	0.98008	O1263	0.944938	O6282	0.942965	O0186	0.929531	O7270	0.928948	O2848	0.904029	O4522	0.887279	O4369	0.871284
OrderID	eigenvector_centrality																																												
O4449	0.1379939757																																												
O5635	0.1375687508																																												
O3032	0.1355738328																																												
O1263	0.1307544089																																												
O6282	0.1300681576																																												
O7270	0.1285119034																																												
O0186	0.1283726203																																												
O2848	0.1251526050																																												
O4522	0.1229054138																																												
O4369	0.1204853585																																												
Id	Eigenvector ...																																												
O4449	1.0																																												
O5635	0.994534																																												
O3032	0.98008																																												
O1263	0.944938																																												
O6282	0.942965																																												
O0186	0.929531																																												
O7270	0.928948																																												
O2848	0.904029																																												
O4522	0.887279																																												
O4369	0.871284																																												

**Nhận xét:** Bằng cách tính toán các kết nối mở rộng của một nút, EigenCentrality có thể xác định các hóa đơn có ảnh hưởng trên toàn mạng chứ không chỉ những nút được kết nối trực tiếp với nó. Về ý nghĩa, nó đo lường ảnh hưởng của một hóa đơn dựa trên số lượng liên kết mà nó có với các hóa đơn khác trong mạng để tìm ra hóa đơn có các mặt hàng phổ biến trong các hóa đơn khác.

## 1.5. Pagerank

Jupyter notebook	Gephi
<p>➤ Cài đặt</p> <pre>sorted(list(nx.degree(G)), key=lambda       x:x[1], reverse=True)</pre> <p>➤ Kết quả</p>	<p>➤ Kết quả</p>

OrderID	pagerank
4	O0186 0.0179121843
65	O3032 0.0164814646
93	O4449 0.0161644179
28	O1263 0.0159605256
113	O5635 0.0152110140
124	O6282 0.0145987977
50	O2214 0.0120615572
144	O7270 0.0120427498
94	O4522 0.0118666115
199	O9793 0.0116686033
60	O2848 0.0116317323
100	O5002 0.0111130130
89	O4369 0.0108601167
91	O4389 0.0107324427
68	O3209 0.0105825205
29	O1350 0.0104162732
122	O6200 0.0099820721
165	O8019 0.0095322564
75	O3548 0.0094119751
173	O8290 0.0093326839

Id	PageRank
O4449	0.012137
O6282	0.011839
O0186	0.011282
O1263	0.011027
O3032	0.010825
O5635	0.010662
O7270	0.009867
O4369	0.009534
O5002	0.009421
O2848	0.009363
O4522	0.009296
O1350	0.009188
O9793	0.009114
O9402	0.008938
O3548	0.008915
O2214	0.008825
O4389	0.008616

**Nhận xét:**

## 1.6. Betweenness Centrality

Jupyter notebook	Gephi																																												
<p>➤ Cài đặt</p> <pre>betweenness = nx.betweenness centrality(G)</pre> <p>➤ Kết quả</p> <table> <tr> <th>OrderID</th><th>betweenness centrality</th></tr> <tr><td>O6282</td><td>0.0412943186</td></tr> <tr><td>O4449</td><td>0.0330548682</td></tr> <tr><td>O0186</td><td>0.0269571846</td></tr> <tr><td>O1350</td><td>0.0232821654</td></tr> <tr><td>O1263</td><td>0.0227998195</td></tr> <tr><td>O9402</td><td>0.0175789136</td></tr> <tr><td>O8019</td><td>0.0171694752</td></tr> <tr><td>O3032</td><td>0.0166365143</td></tr> <tr><td>O4369</td><td>0.0141601174</td></tr> <tr><td>O4263</td><td>0.0135636317</td></tr> </table>	OrderID	betweenness centrality	O6282	0.0412943186	O4449	0.0330548682	O0186	0.0269571846	O1350	0.0232821654	O1263	0.0227998195	O9402	0.0175789136	O8019	0.0171694752	O3032	0.0166365143	O4369	0.0141601174	O4263	0.0135636317	<p>➤ Kết quả</p> <table> <tr> <th>Id</th><th>Betweenness Centrality</th></tr> <tr><td>O6282</td><td>813.539371</td></tr> <tr><td>O4449</td><td>651.213959</td></tr> <tr><td>O0186</td><td>531.083493</td></tr> <tr><td>O1350</td><td>458.68194</td></tr> <tr><td>O1263</td><td>449.179245</td></tr> <tr><td>O9402</td><td>346.322177</td></tr> <tr><td>O8019</td><td>338.255832</td></tr> <tr><td>O3032</td><td>327.755968</td></tr> <tr><td>O4369</td><td>278.968472</td></tr> <tr><td>O4263</td><td>267.217108</td></tr> </table>	Id	Betweenness Centrality	O6282	813.539371	O4449	651.213959	O0186	531.083493	O1350	458.68194	O1263	449.179245	O9402	346.322177	O8019	338.255832	O3032	327.755968	O4369	278.968472	O4263	267.217108
OrderID	betweenness centrality																																												
O6282	0.0412943186																																												
O4449	0.0330548682																																												
O0186	0.0269571846																																												
O1350	0.0232821654																																												
O1263	0.0227998195																																												
O9402	0.0175789136																																												
O8019	0.0171694752																																												
O3032	0.0166365143																																												
O4369	0.0141601174																																												
O4263	0.0135636317																																												
Id	Betweenness Centrality																																												
O6282	813.539371																																												
O4449	651.213959																																												
O0186	531.083493																																												
O1350	458.68194																																												
O1263	449.179245																																												
O9402	346.322177																																												
O8019	338.255832																																												
O3032	327.755968																																												
O4369	278.968472																																												
O4263	267.217108																																												

## 2. CÁC THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG

### 2.1.Louvain

#### a. Python

##### ➤ Cài đặt thuật toán và sử dụng thuật toán

```
# Cài đặt các package liên quan
!pip install python-louvain
import matplotlib.cm as cm
import matplotlib
import community as community_louvain

# Tìm các phần tốt nhất
partition = community_louvain.best_partition(G)

# Vẽ biểu đồ
plt.figure(figsize=(20,17))
pos = nx.spring_layout(G)

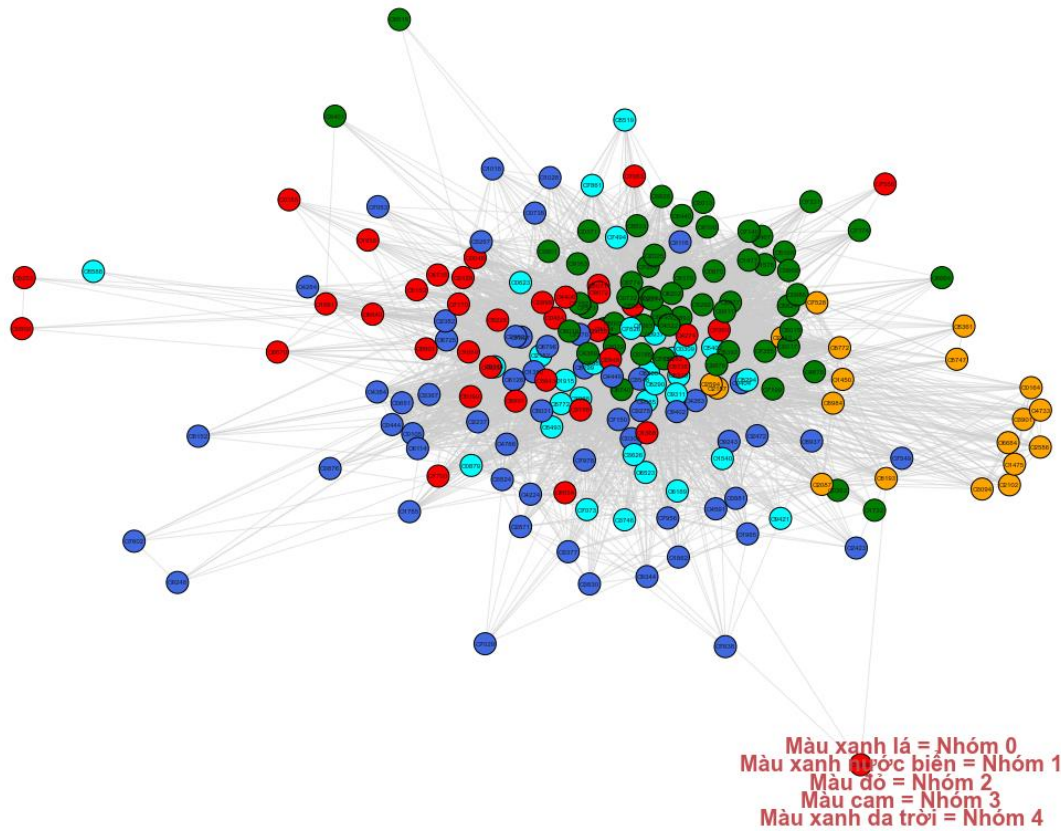
# Tô màu node theo từng partition
cmap = cm.get_cmap('plasma', max(partition.values()) + 1)
nx.draw_networkx_edges(G, pos, alpha=0.5, edge_color="#CCCCCC")
for node, color in partition.items():
    nx.draw_networkx_nodes(G, pos, [node], node_size=300, cmap=cmap,
        node_color=[cmap.colors[color]])

nx.draw_networkx_labels(G, pos)
plt.show()
```

##### ➤ Kết quả sau khi chạy thuật toán



## Thuật toán Louvain



Kết quả ta thu 5 cụm.

```
1 for i in range(max(partition.values())+1):
2     print("Nhóm ", i)
3     x = nodeInCluster(i)
4     print(x)
```

✓ 0.3s Python

Nhóm 0  
['03292', '08774', '05927', '00986', '05178', '05533', '07174', '07571', '07702', '07864', '07199', '05440', '08664', '02025', '01893', '01134', '01576', '02467', '05002', '09793', '03209', '07385', '03032', '08371', '03801', '03013', '05392', '01477', '07052', '00186', '01732', '07140', '00262', '00875', '09163', '03017', '02214', '08108', '08977', '08015', '07331', '06202', '09401', '06740', '00732', '01263', '00011', '04676', '04369', '08019', '00668', '06515', '09100', '04522', '03628', '09676']

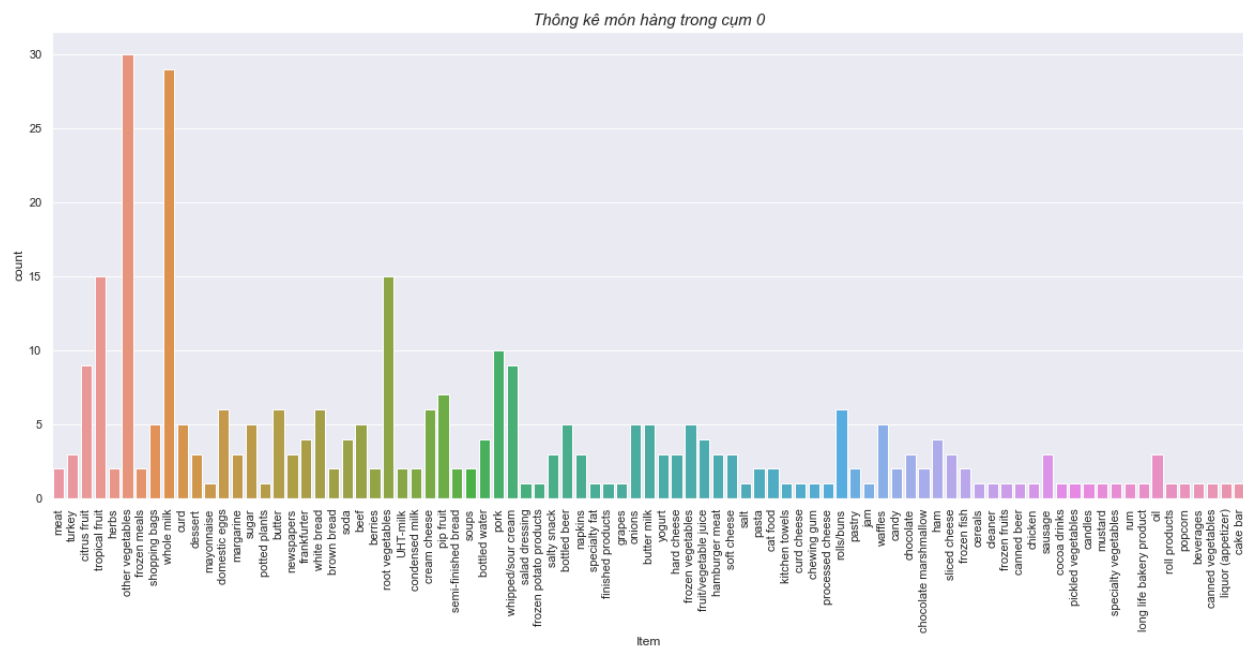
Nhóm 1  
['07897', '00305', '07053', '08937', '06725', '04384', '01985', '09404', '00444', '03830', '09243', '04284', '01350', '07150', '04263', '02237', '00651', '01785', '03548', '06126', '07549', '07602', '08199', '02382', '01862', '00881', '00105', '07029', '01028', '06796', '02982', '07978', '04766', '07956', '06114', '07638', '03116', '03267', '04224', '03377', '08152', '03367', '00876', '07270', '09402', '03524', '03275', '09344', '00738', '02472', '04591', '04449', '08031', '09248', '02423', '01018', '02871']

Nhóm 2  
['07586', '00808', '01669', '00358', '07968', '07063', '01368', '07837', '08034', '08670', '05253', '01958', '00484', '02429', '07060', '00048', '05099', '01961', '04406', '03788', '07700', '03643', '05635', '06738', '05071', '09072', '06840', '04925', '05801', '05735', '02848', '07170', '05152', '08697', '08225', '00999', '04274']

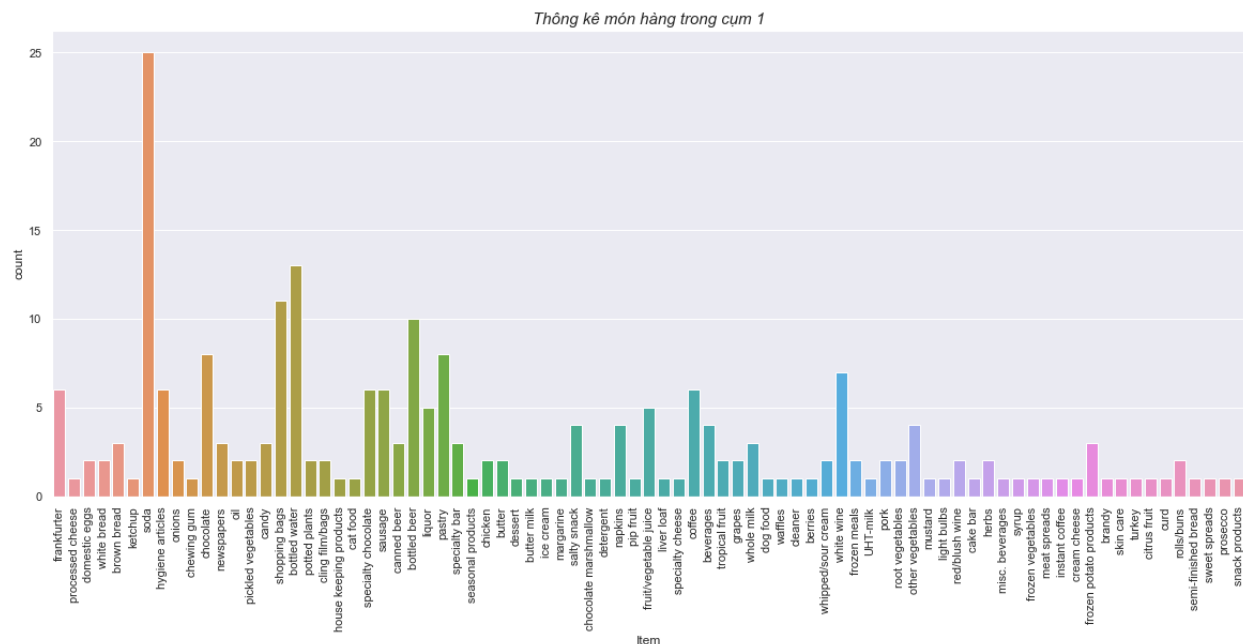
Nhóm 3  
['01450', '05361', '05747', '02157', '08984', '00164', '02102', '03094', '07528', '02087', '02586', '02243', '06684', '03901', '04733', '08193', '02594', '01475', '08772']

Nhóm 4  
['07073', '00937', '08294', '02152', '09421', '05330', '06189', '03968', '09311', '05402', '00879', '01540', '06200', '06282', '08493', '05772', '03746', '08588', '09346', '00623', '05519', '09565', '01915', '04389', '06523', '03626', '00309', '08290', '07861', '07494', '07826']

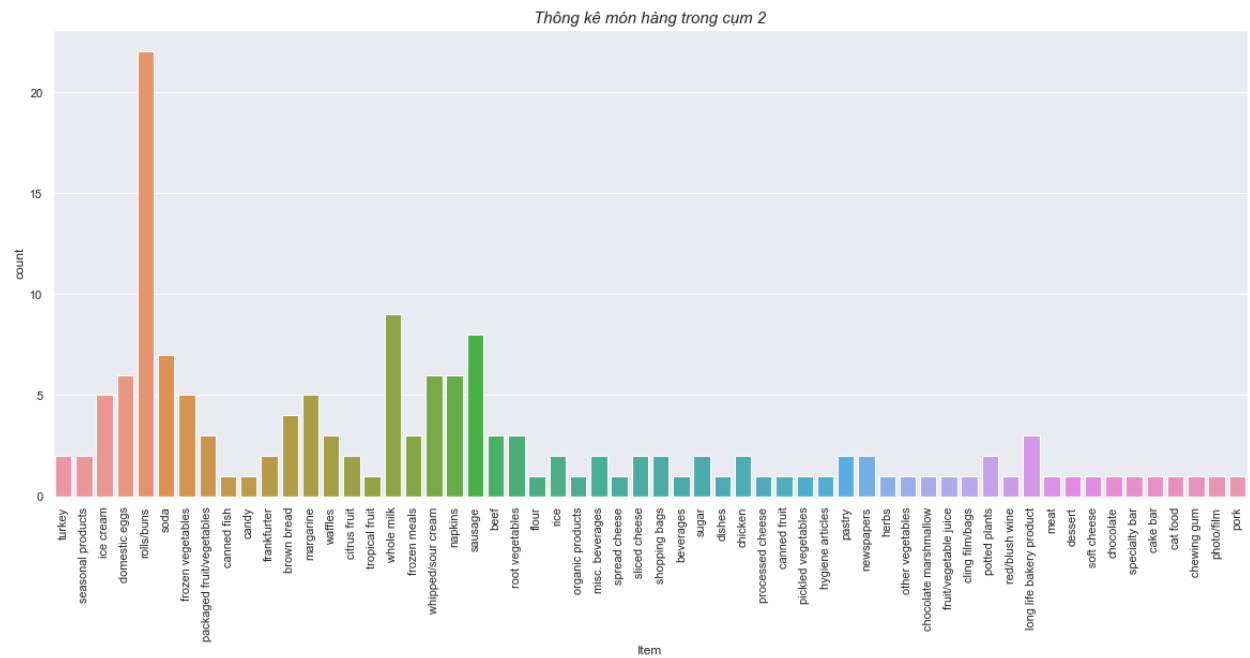
**Cụm 0:** có 55 hóa đơn. Những mặt hàng, nhiều người mua trong cụm là **other vegetables, whole milk, root vegetable, tropical fruit**



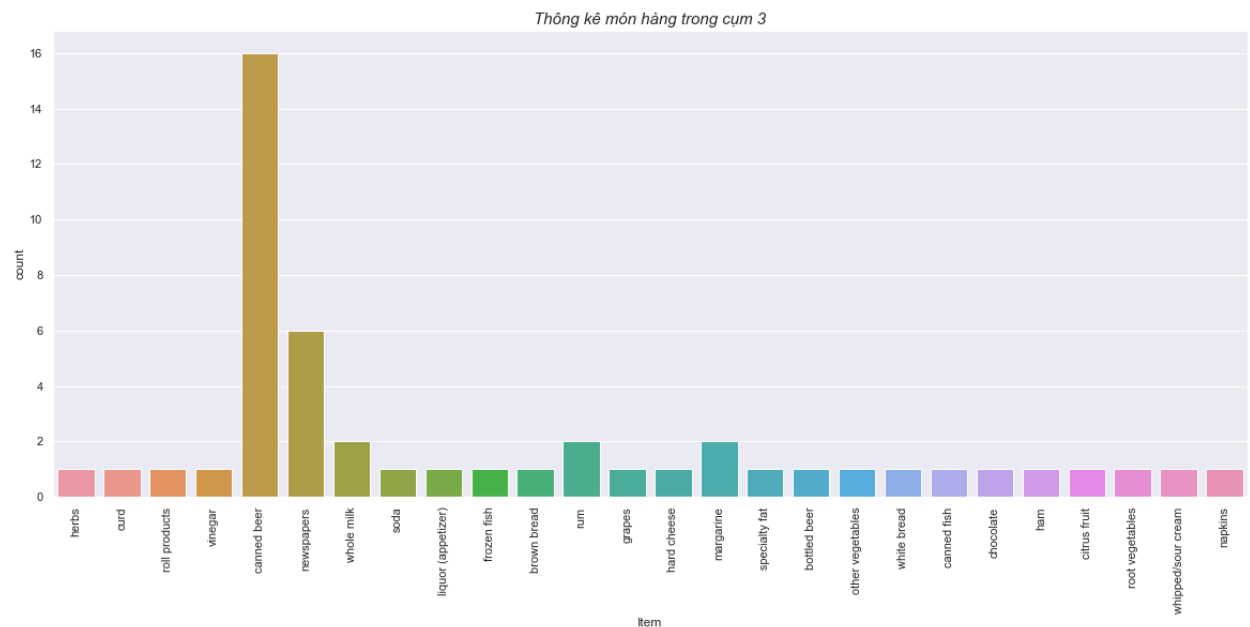
**Cụm 1:** gồm 57 hóa đơn. Các mặt hàng được mua nhiều trong hóa đơn là **soda, bottled water, shopping bags**



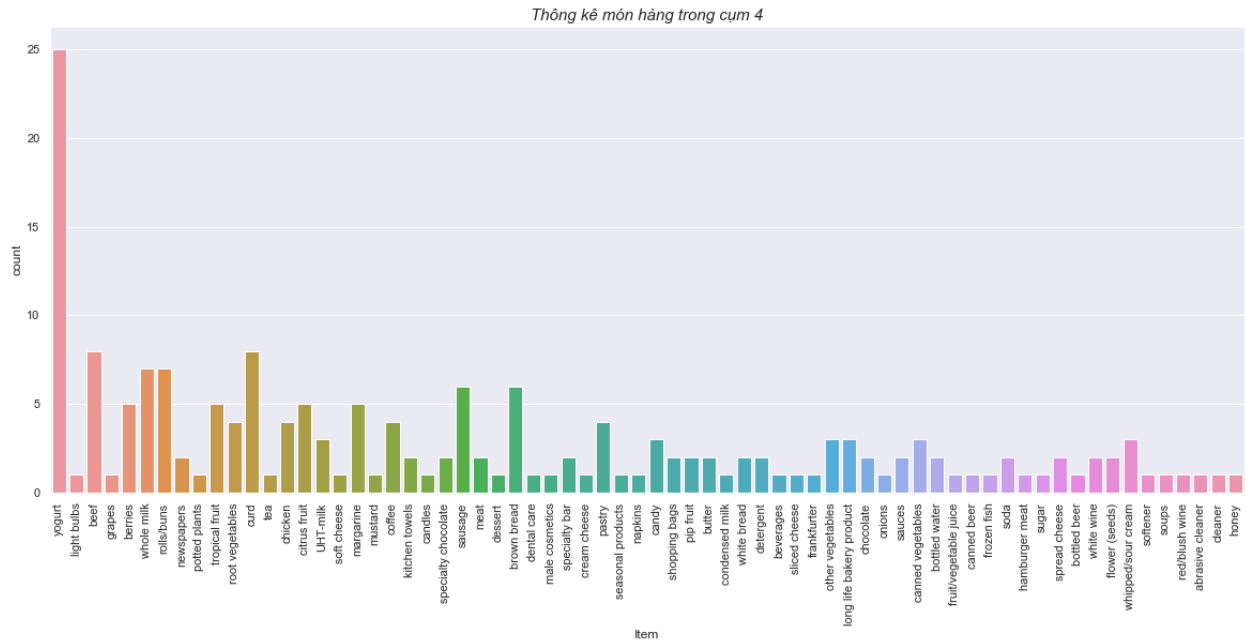
**Cụm 2:** gồm 37 hóa đơn. Trong đó, các mặt hàng được mua nhiều là **rolls/buns**



**Cụm 3:** gồm 19 hóa đơn, trong đó các mặt hàng được mua nhiều là **canned beer**



**Cụm 4:** gồm 31 hóa đơn, trong đó các mặt hàng được mua nhiều bao gồm **yogurt**




## b. Bằng Gephi

### ➤ Cài đặt và sử dụng thuật toán

- Trong ô Settings, chỗ mục Modularity chọn “Run”

Settings		
<input checked="" type="checkbox"/> <b>Network Overview</b>		
Average Degree	56.78	Run ?
Avg. Weighted Degree	77.88	Run ?
Network Diameter	3	Run ?
Graph Density	0.285	Run ?
HITS		Run ?
Modularity	0.174	Run ?
Clustering Coefficient		Run ●
PageRank		Run ?
Connected Components	1	Run ?
Girvan-Newman Clustering		Run ?

- Chọn như hình rồi click “Run”

 Modularity settings ×

**Modularity**  
Community detection algorithm.

☒ Randomize

Produce a better decomposition but increases computation time

☒ Use weights

Use edge weight

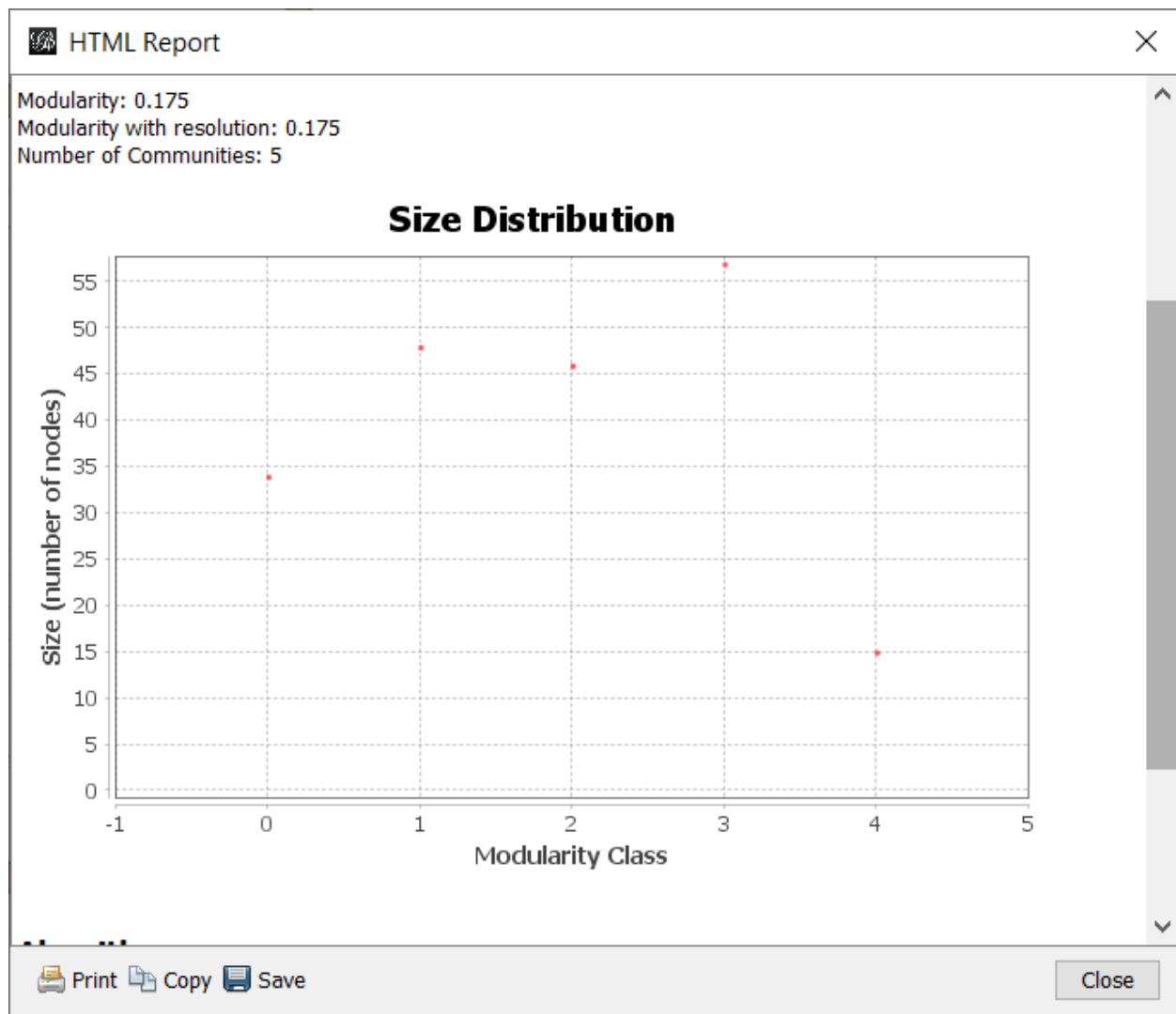
Resolution:

Lower to get more communities (smaller ones) and higher than 1.0 to get less communities (bigger ones).

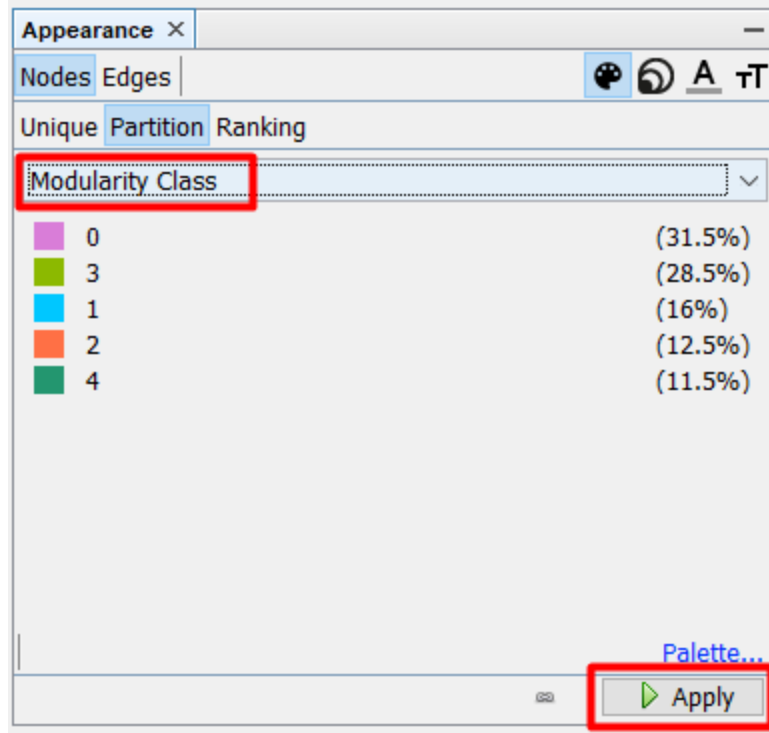
OK

Cancel

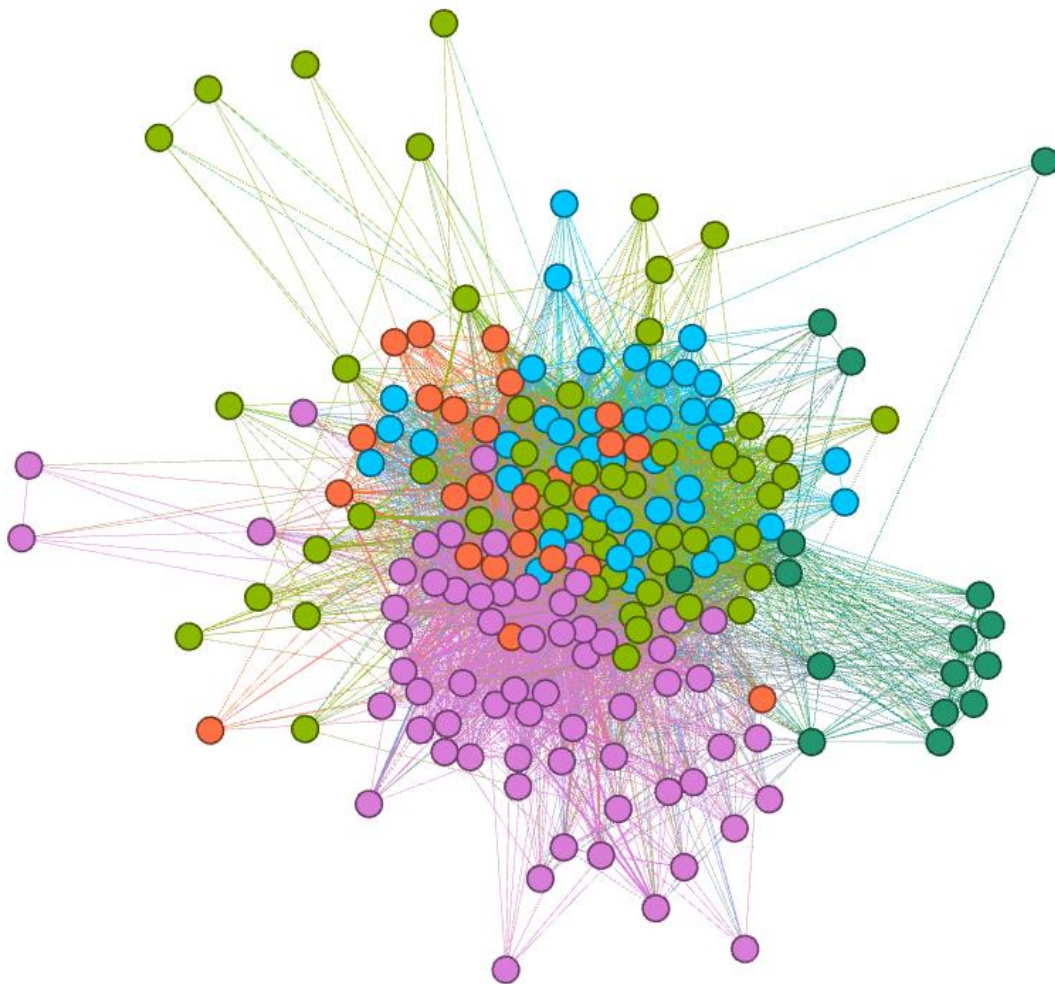
- Ta có kết quả gom cụm như hình sau.



- Trực quan hóa kết quả gom cụm lên đồ thị.



- Kết quả sau khi điều chỉnh màu



➤ **Kết quả sau khi chạy thuật toán**

Ta được:

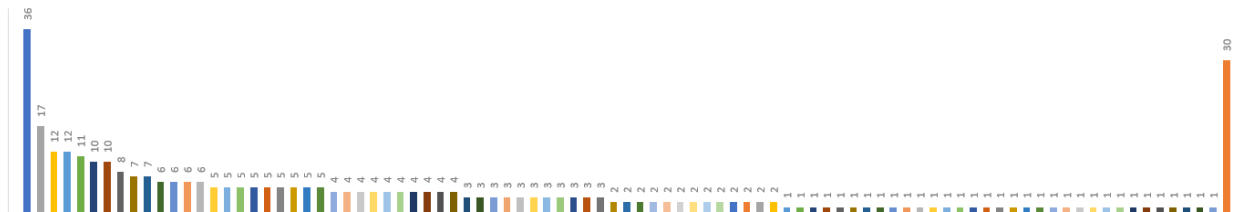
- Cụm 0 (màu hồng): có 64 node
- Cụm 1 (màu xanh da trời): có 33 node
- Cụm 2 (màu cam): có 26 node
- Cụm 3 (màu xanh chuối): có 58 node
- Cụm 4 (màu xanh rêu) : có 24 node

Kết quả gom cụm:

**Cụm 0:** gồm 64 hóa đơn, các mặt hàng được mua nhiều là **whole milk, other vegetables, tropical fruit**



■ whole milk	■ tropical fruit	■ rolls/buns	■ root vegetables	■ citrus fruit	■ pork	■ whipped/sour cream	■ shopping bags
■ pip fruit	■ frozen vegetables	■ butter	■ cream cheese	■ domestic eggs	■ waffles	■ beef	■ fruit/vegetable juice
■ butter milk	■ sugar	■ frozen meals	■ soda	■ curd	■ bottled beer	■ white bread	■ frankfurter
■ chocolate	■ bottled water	■ ham	■ dessert	■ onions	■ margarine	■ sliced cheese	■ salty snack
■ soft cheese	■ yogurt	■ sausage	■ brown bread	■ hamburger meat	■ pastry	■ turkey	■ newspapers
■ meat	■ hard cheese	■ oil	■ napkins	■ specialty bar	■ berries	■ frozen potato products	■ grapes
■ pasta	■ cleaner	■ candy	■ frozen fish	■ semi-finished bread	■ cat food	■ chocolate marshmallow	■ long life bakery product
■ soups	■ potted plants	■ chicken	■ coffee	■ misc. beverages	■ curd cheese	■ herbs	■ processed cheese
■ popcorn	■ finished products	■ beverages	■ dog food	■ specialty chocolate	■ condensed milk	■ pickled vegetables	■ liquor (appetizer)
■ UHT-milk	■ salt	■ cling film/bags	■ canned vegetables	■ chewing gum	■ rum	■ mustard	■ specialty fat
■ mayonnaise	■ roll products	■ kitchen towels	■ cereals	■ candles	■ cocoa drinks	■ cake bar	■ jam
■ salad dressing	■ specialty vegetables	■ other vegetables					

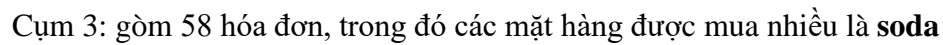


**SỐ LƯỢNG TỪNG SẢN PHẨM CỦA GROUP 1**

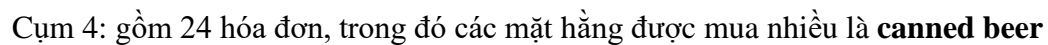


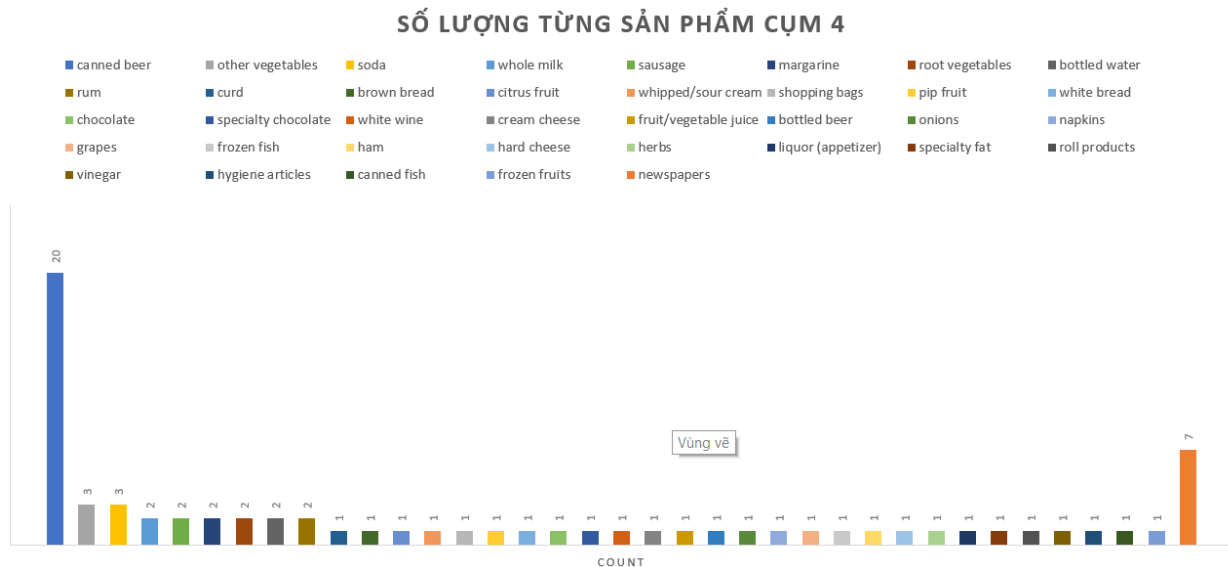
Trang | 33

■ bottled beer   ■ chocolate   ■ candy   ■ bottled water   ■ white wine   ■ specialty chocolate   ■ fruit/vegetable juice  
■ liquor   ■ root vegetables   ■ pastry   ■ UHT-milk   ■ white bread   ■ specialty bar   ■ frankfurter  
■ seasonal products   ■ turkey   ■ curd   ■ whole milk   ■ rolls/buns   ■ brown bread   ■ citrus fruit  
■ butter   ■ soda   ■ condensed milk   ■ domestic eggs   ■ frozen meals   ■ frozen potato products   ■ semi-finished bread  
■ cat food   ■ misc. beverages   ■ pickled vegetables   ■ cake bar   ■ prosecco   ■ brandy   ■ sweet spreads  
■ snack products   ■ skin care   ■ shopping bags



■ soda	■ sausage	■ napkins	■ pastry	■ whipped/sour cream	■ bottled water	■ domestic eggs
■ frankfurter	■ ice cream	■ hygiene articles	■ brown bread	■ margarine	■ root vegetables	■ beverages
■ whole milk	■ coffee	■ chicken	■ newspapers	■ potted plants	■ frozen vegetables	■ herbs
■ beef	■ other vegetables	■ shopping bags	■ red/blush wine	■ pork	■ waffles	■ salty snack
■ packaged fruit/vegetables	■ long life bakery product	■ chocolate	■ white wine	■ sugar	■ bottled beer	■ onions
■ oil	■ chocolate marshmallow	■ processed cheese	■ pickled vegetables	■ cling film/bags	■ chewing gum	■ liquor
■ rice	■ tropical fruit	■ berries	■ butter	■ white bread	■ specialty chocolate	■ detergent
■ spread cheese	■ fruit/vegetable juice	■ dessert	■ sliced cheese	■ grapes	■ mustard	■ light bulbs
■ seasonal products	■ butter milk	■ frozen meals	■ turkey	■ frozen potato products	■ cat food	■ misc. beverages
■ cake bar	■ flour	■ photo/film	■ specialty cheese	■ house keeping products	■ canned fish	■ liver loaf
■ canned fruit	■ instant coffee	■ meat spreads	■ dishes	■ ketchup	■ organic products	■ rolls/buns





### c. So sánh

Thực hiện so sánh kết quả gom cụm bằng tool Gephi và bằng code trên Jupyter.

- Cả 2 đều cho gom thành 4 cụm
- Chi tiết mỗi cụm như sau

Jupyter	Gephi
Cụm 0: other vegetables, whole milk, root vegetable, tropical fruit.	Cụm 0: whole milk, other vegetables, tropical fruit.
Cụm 1: soda, bottled water, shopping bags	Cụm 1: yogurt.
Cụm 2: rolls/buns	Cụm 2: bottled beer, shopping bags, chocolate
Cụm 3: canned beer.	Cụm 3: soda.
Cụm 4: yogurt.	Cụm 4: canned beer.

Ta thấy kết quả gom khi thực hiện trên 2 phần mềm ra kết quả có vừa có sự giống nhau và khác biệt. Cụ thể:

Giống nhau: Các mặt hàng được gom chung trong cả 2 phần mềm là:

- Mặt hàng: other vegetables, whole milk, tropical fruit
- Mặt hàng: yogurt.
- Mặt hàng: canned beer.

Khác nhau:

- Kết quả bằng Jupyter gom mặt hàng soda chung với bottled water, shopping bags và một cụm cho rolls/buns.
- Kết quả bằng Jupyter tách cụm hóa đơn mua nhiều mặt hàng soda riêng và cụm các mặt hàng bottled beer, shopping bags, chocolate đi chung với nhau.

#### d. Nhận xét

Từ việc so sánh việc gom cụm trên cả 2 phần mềm, ta có thể nhận xét với mỗi thời điểm mua hàng có 3 nhóm:

- Nhóm thứ nhất khi other vegetables thì có xu hướng mua whole milk và tropical fruit và ngược lại.
- Nhóm thứ hai có xu hướng mua yogurt
- Nhóm thứ ba có xu hướng mua canned beer

### 2.2.K-means

#### ➤ Cài đặt thuật toán và sử dụng thuật toán

##### Bước 1: Chọn số lượng cụm

- Đầu tiên, chúng ta chọn phương pháp cùi chỏ để chọn số lượng cụm

##### Phương pháp “Elbow”:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
import plotly.graph_objects as go
import numpy as np
X=df_vector.drop("OrderID",axis=1)
scaler = MinMaxScaler()
scaler.fit(X)
X=scaler.transform(X)
inertia = []
for i in range(1,11):
    kmeans = KMeans(
        n_clusters=i, init="k-means++",
        n_init=10,
        tol=1e-04, random_state=42
    )
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

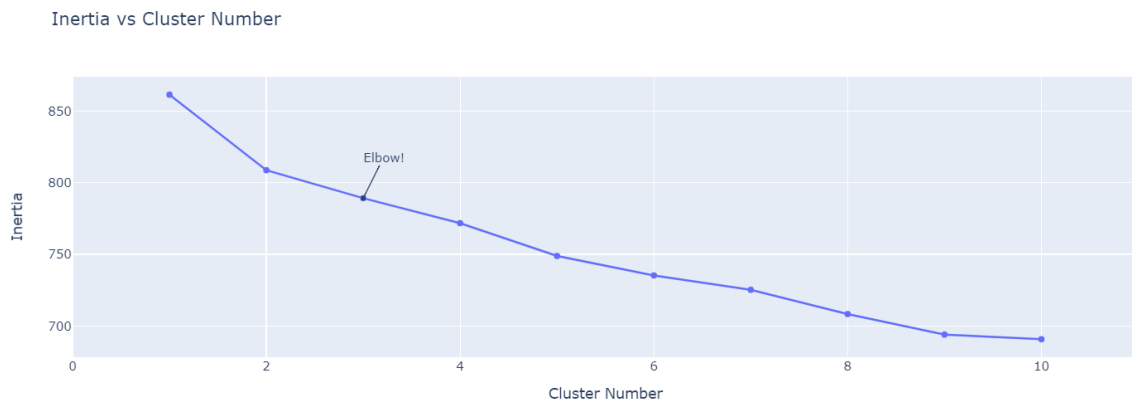
fig = go.Figure(data=go.Scatter(x=np.arange(1,11),y=inertia))
fig.update_layout( title="Inertia vs Cluster Number",
                    xaxis=dict(range=[0,11],title="Cluster Number"),
                    yaxis={'title':'Inertia'},
                    annotations=[
                        dict(
                            x=3,
                            y=inertia[2],
                            xref="x",
```

```

        yref="y",
        text="Elbow!",
        showarrow=True,
        arrowhead=7,
        ax=20,
        ay=-40
    )
]
)

```

### Kết quả:



- Vì kết quả của phương pháp cùi chỏ không rõ ràng. Câu hỏi đặt ra là chúng ta nên chọn số cụm là bao nhiêu trong khoảng từ 2 đến 9. Nên ta tính thêm hệ số Silhouette.

### Silhouette coefficient

```

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

X=df_vector.drop("OrderID",axis=1)

range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9]

for n_clusters in range_n_clusters:

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the
    # formed
    # clusters
    silhouette_avg = silhouette_score(X, cluster_labels)

```

```

print( "For n_clusters =", n_clusters, ". The average silhouette_score
is :", silhouette_avg)

# Compute the silhouette scores for each sample
sample_silhouette_values = silhouette_samples(X, cluster_labels)

y_lower = 10
for i in range(n_clusters):
    # Aggregate the silhouette scores for samples belonging to
    # cluster i, and sort them
    ith_cluster_silhouette_values =
        sample_silhouette_values[cluster_labels == i]

    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

```

#### Kết quả:

```

For n_clusters = 2 The average silhouette_score is : 0.2106902344388537
For n_clusters = 3 The average silhouette_score is : 0.09901905655401791
For n_clusters = 4 The average silhouette_score is : 0.05877333083866951
For n_clusters = 5 The average silhouette_score is : 0.10650799759632329
For n_clusters = 6 The average silhouette_score is : 0.05851082463488078
For n_clusters = 7 The average silhouette_score is : 0.07026135931919629
For n_clusters = 8 The average silhouette_score is : 0.054691947398296546
For n_clusters = 9 The average silhouette_score is : 0.07582918385505899

```

- Ta thấy số lượng cụm là 2 là có điểm silhouette cao nhất nên ta chọn số lượng cụm cần gom là 2

#### Bước 2: Gom cụm bằng thuật toán Kmeans

##### Thuật toán gom cụm Kmeans

```

import pandas as pd
from sklearn.cluster import KMeans
import numpy as np

kmeans = KMeans(n_clusters=2)
X=df_vector.drop("OrderID",axis=1)
y = kmeans.fit_predict(X)

df_vector['Cluster'] = y

# Tô màu cho từng cụm

```

```

color_map = []
for index, row in df_vector.iterrows():
    value = row['Cluster']
    if value == 0:
        color_map.append('green')
    elif value == 1:
        color_map.append('royalblue')
    else:
        color_map.append('orange')

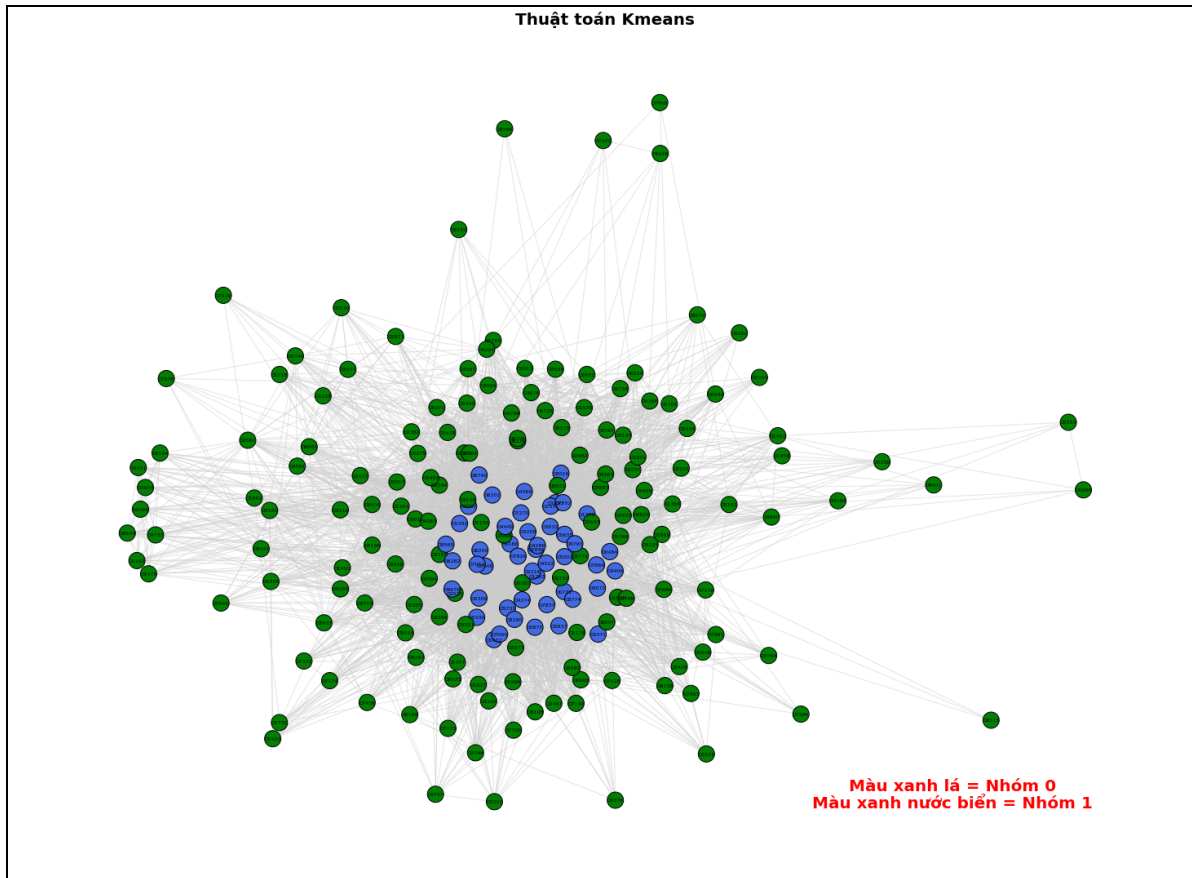
# Vẽ đồ thị
plt.subplots(figsize=(20, 15))
pos = nx.spring_layout(G)

nx.draw_networkx_edges(G, pos, alpha=0.5, edge_color="#CCCCCC")
nx.draw_networkx_nodes(G, pos, node_size=400, node_color= color_map,
edgecolors='black')
nx.draw_networkx_labels(G, pos, font_size=6)

plt.show()

```

**Kết quả:** Ta gom được 2 cụm và có kết quả gom cụm như sau. Trong đó cụm 0 được tô màu xanh lá và cụm 1 được tô màu xanh dương.



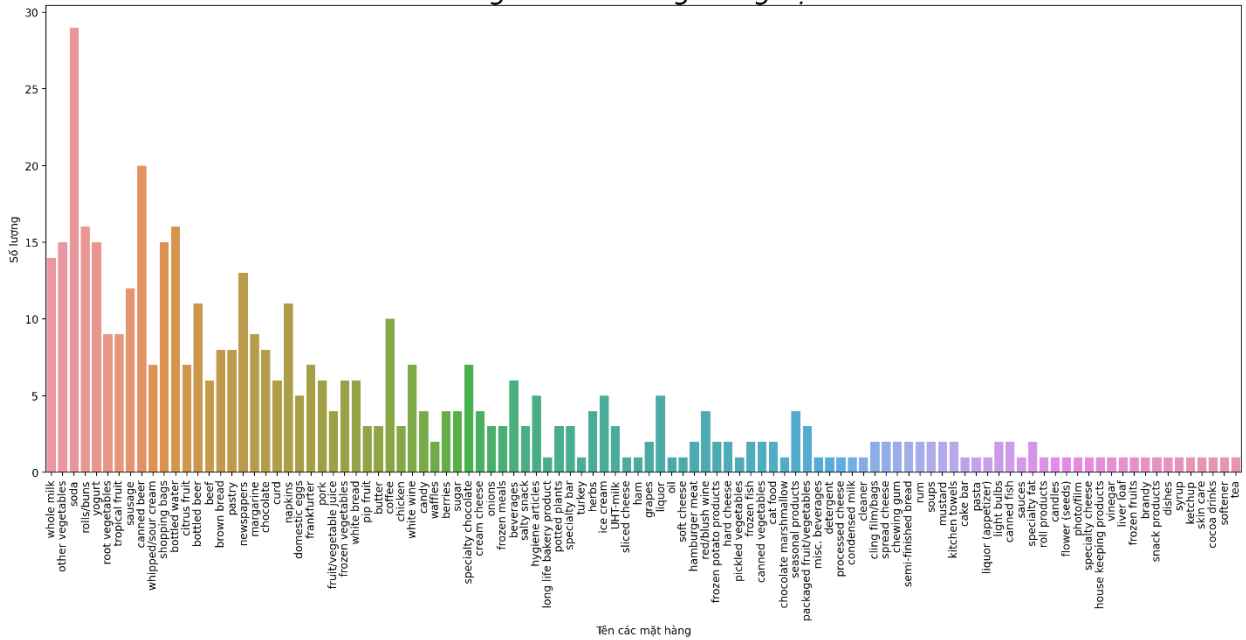
### Bước 3: Phân tích kết quả gom cụm:

- Từ 2 cụm, ta phân tích các mặt hàng có trong cụm và ta được kết quả như sau.

Cụm 0: gồm 40 hóa đơn, trong đó các mặt hàng được mua nhiều là **soda, canned beer, bottled water, rolls/buns**

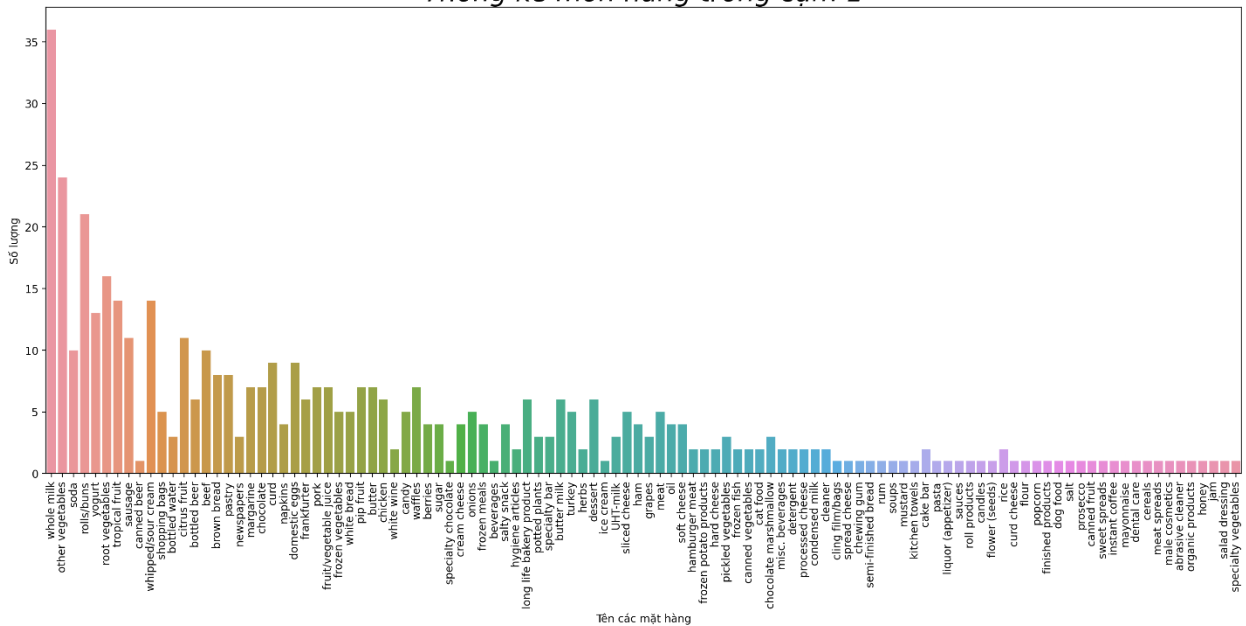


Thông kê món hàng trong cụm 0



Cụm 1: gồm 140 hóa đơn, trong đó các mặt hàng được mua nhiều là **whole milk, other vegetables, rolls/buns, root vegetables**

Thông kê món hàng trong cụm 1



#### Bước 4: Nhận xét

Ta được 2 cụm, trong đó:

- Cụm 0: có lượng hóa đơn mua soda rất cao (28 trên tổng số 40 hóa đơn). Và khách hàng ở nhóm này có xu hướng mua soda chung với canned beer, bottled water, rolls/buns và

ngược lại. Ta có thể suy được những hóa đơn này có thể phục vụ cho những mục đích phục vụ tiệc, bữa ăn nhẹ cho giới văn phòng hoặc cho những khách du lịch, ...

- **Cụm 1:** có số lượng hóa đơn mua whole milk cao nhất. Khách hàng ở nhóm này có xu hướng mua whole milk kèm với other vegetables, rolls/buns, root vegetables và ngược lại. Ta suy được những hóa đơn này có thể phục vụ cho những mục đích nấu ăn hoặc các phục vụ các bữa ăn chính/bữa ăn nhẹ.

## 2.3.Girvan Newman

### a. Python

#### ➤ Cài đặt thuật toán và sử dụng thuật toán

#### Bước 1: Tính toán gom cụm

- **Cách 1:** Dùng thư viện của network

```
from networkx.algorithms import community
communities_generator = community.girvan_newman(G)
```

Ta thu được kết quả là 3 cụm, cụ thể như hình sau

```
Cụm 0
{'07174', '01475', '09163', '07549', '02102', '07571', '08152', '07140', '09311', '00879', '01958', '07978', '06114', '04449', '02087', '01018',
'09346', '00937', '03626', '05099', '02472', '03801', '08225', '01477', '00875', '08670', '05519', '02157', '01134', '07053', '00732', '01862',
'07331', '00651', '07063', '07602', '08697', '00011', '04263', '03830', '03901', '09404', '00986', '05735', '05772', '09421', '05801', '03017',
'04224', '02152', '03032', '03013', '04389', '08493', '07060', '08290', '03267', '07052', '07150', '05002', '00881', '02237', '01985', '01368',
'00668', '00876', '04284', '02594', '07861', '07956', '00371', '04406', '05178', '06126', '02243', '00186', '03524', '08199', '00999', '08937',
'07199', '02423', '07826', '09676', '04733', '08193', '05533', '03209', '00358', '04925', '03746', '09248', '07170', '01961', '02429', '00305',
'07494', '08772', '02982', '08108', '06515', '05152', '07073', '02025', '04676', '02848', '05392', '04369', '07968', '09793', '05440', '00262',
'00105', '02467', '04766', '05635', '01540', '08034', '06200', '08294', '08977', '01350', '06684', '09100', '01450', '06523', '03788', '08015',
'03968', '04522', '01263', '05927', '06740', '03292', '05747', '01028', '08031', '07897', '05402', '00164', '06725', '04384', '07702', '01576',
'05361', '02214', '08984', '08664', '07638', '03643', '02586', '03275', '07528', '03367', '07270', '01915', '00484', '03377', '06840', '08019',
'06738', '02382', '05071', '09401', '06282', '03094', '07864', '06189', '04274', '00444', '09344', '00048', '09072', '07586', '07029', '03548',
'01893', '06796', '07700', '03628', '03116', '07385', '00623', '01732', '05330', '04591', '00309', '00738', '06202', '01669', '09243', '07837',
'09402', '09565', '02871', '01785', '08774'}

Cụm 1
{'00808', '05253'}

Cụm 2
{'08588'}
```

- **Cách 2:** Tự code thuật toán

```
def CmtyGirvanNewmanStep(G):
    init_ncomp = nx.number_connected_components(G)    #Số lượng components ban đầu
    ncomp = init_ncomp

    while (ncomp <= init_ncomp):
        bw = nx.edge_betweenness centrality(G)    #Tính edge betweenness

        #Tìm tất cả cạnh có độ đo edge betweenness cao nhất bằng nhau và xóa nó đi
        max_ = max(bw.values())
        for k, v in bw.items():
            if float(v) == max_:
                print("xóa cạnh:",k[0],",",k[1] )
                G.remove_edge(k[0],k[1])    # Xóa cạnh
        ncomp = nx.number_connected_components(G)    # Tính lại số lượng components
        x = len(set(list(bw.values())))
```

Ta cũng thu được kết quả là 3 cụm giống với cách 1.

Nhóm 0

07174, 01475, 09163, 07549, 02102, 07571, 08152, 07140, 09311, 00879, 01958, 07978, 06114, 04449, 02087, 01018, 09346, 00937, 03626, 05099, 02472, 03801, 08225, 01477, 00875, 08670, 05519, 02157, 01134, 07053, 00732, 01862, 07331, 00651, 07063, 07602, 08697, 00011, 04263, 03830, 03901, 09404, 00986, 05735, 05772, 09421, 05801, 03017, 04224, 02152, 03032, 03013, 04389, 08493, 07060, 08290, 03267, 07052, 07150, 05002, 00881, 02237, 01985, 01368, 00668, 00876, 04284, 02594, 07861, 07956, 00371, 04406, 05178, 06126, 02243, 00186, 03524, 08199, 00999, 08937, 07199, 02423, 07826, 09676, 04733, 08193, 05533, 03209, 00358, 04925, 03746, 09248, 07170, 01961, 02429, 00305, 07494, 08772, 02982, 08108, 06515, 05152, 07073, 02025, 04676, 02848, 05392, 04369, 07968, 09793, 05440, 00262, 00105, 02467, 04766, 05635, 01540, 08034, 06200, 08294, 08977, 01350, 06684, 09100, 01450, 06523, 03788, 08015, 03968, 04522, 01263, 05927, 06740, 03292, 05747, 01028, 08031, 07897, 05402, 00164, 06725, 04384, 07702, 01576, 05361, 02214, 08984, 08664, 07638, 03643, 02586, 03275, 07528, 03367, 07270, 01915, 00484, 03377, 06840, 08019, 06738, 02382, 05071, 09401, 06282, 03094, 07864, 06189, 04274, 00444, 09344, 00048, 09072, 07586, 07029, 03548, 01893, 06796, 07700, 03628, 03116, 07385, 00623, 01732, 05330, 04591, 00309, 00738, 06202, 01669, 09243, 07837, 09402, 09565, 02871, 01785, 08774,

Nhóm 1

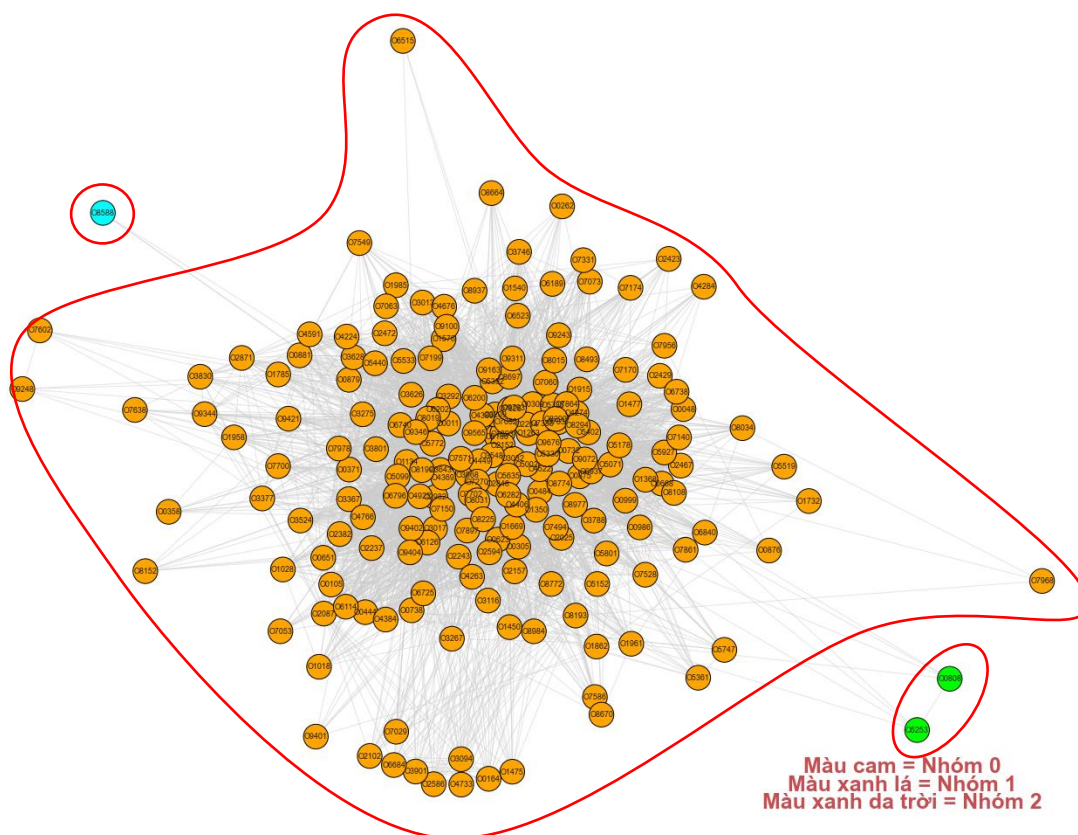
00808, 05253,

Nhóm 2

08588,

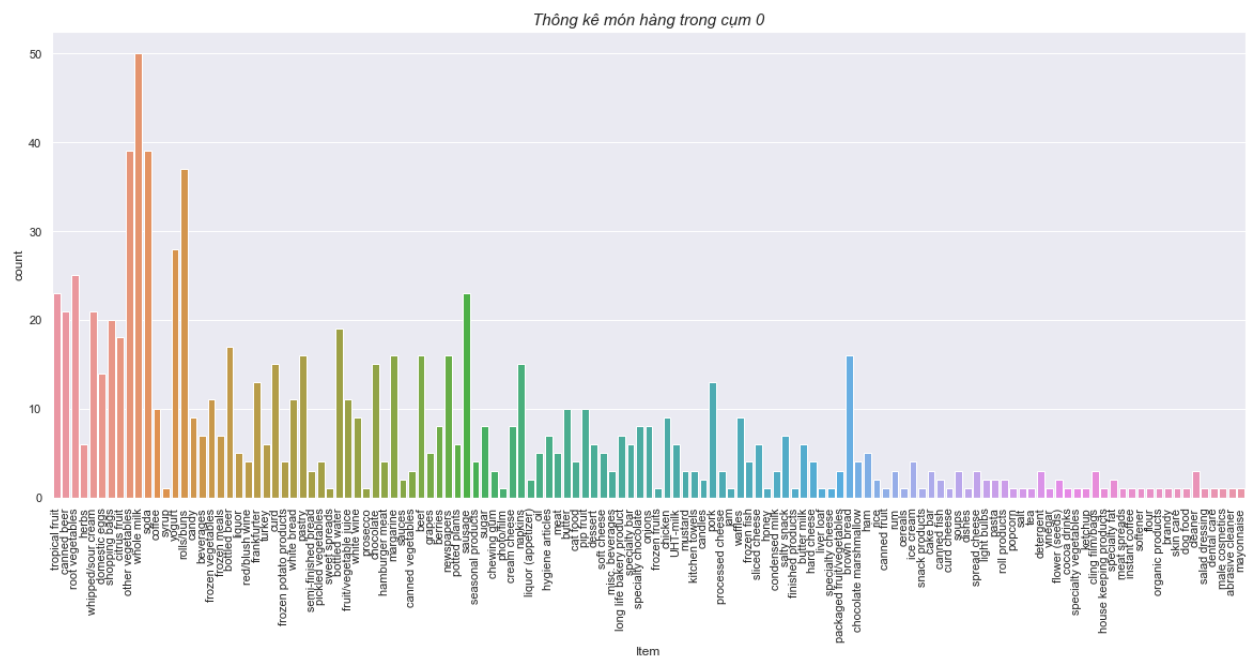
## Bước 2: Trực quan hóa các cụm

### Thuật toán Girvan Newman

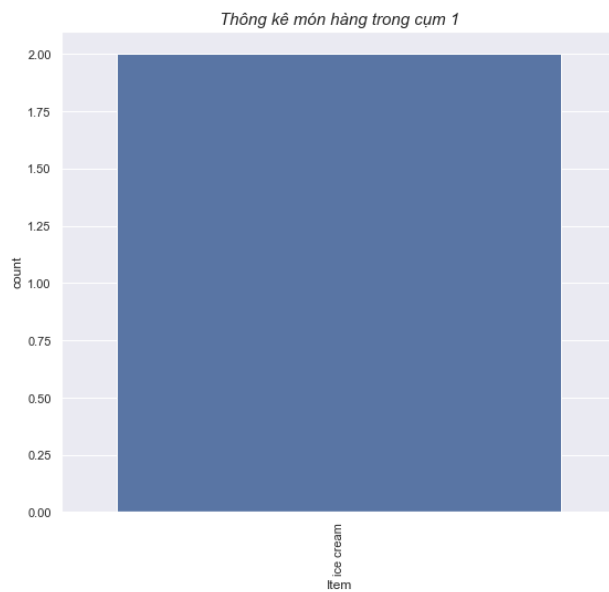


#### ➤ Kết quả khi chạy thuật toán

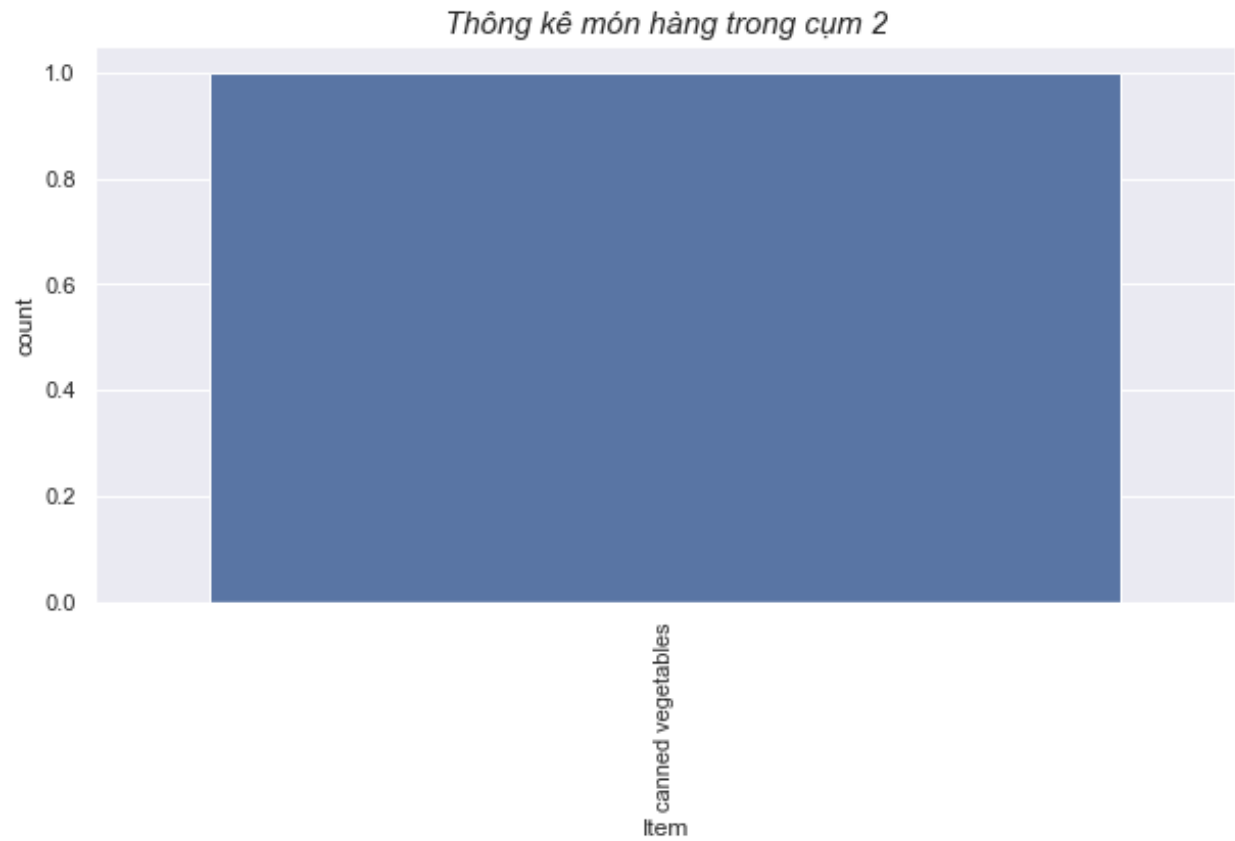
Cụm 0: gồm 197 hóa đơn trong đó các mặt hàng được mua nhiều trong hóa đơn là **other vegetables, whole milk, soda, rolls/buns, yogurt**.



Cụm 1: gồm 2 hóa đơn trong đó các mặt hàng được có trong 2 hóa đơn đó là **ice cream**



Cụm 2: gồm 1 hóa đơn trong đó các mặt hàng được có trong hóa đơn đó là **canned vegetables**



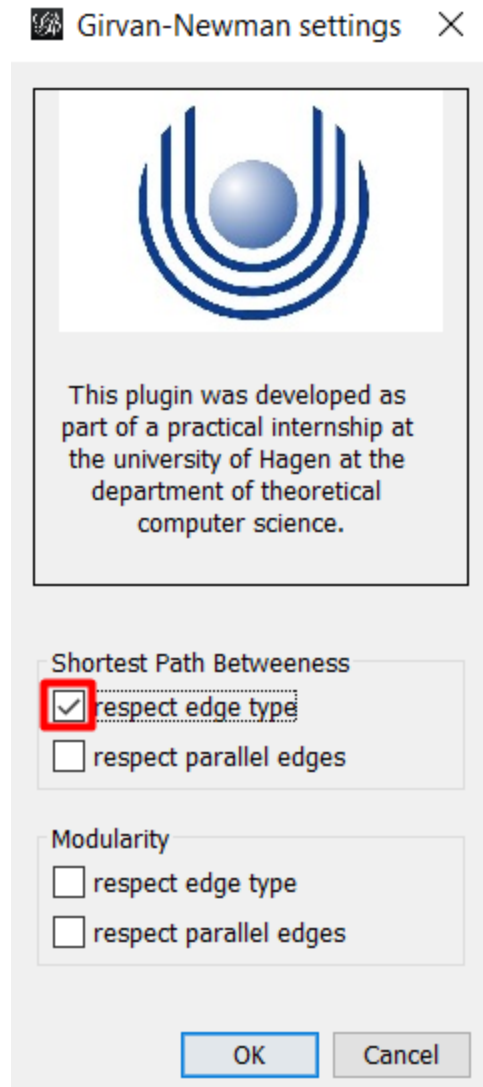
## **b. Gephi**

### ➤ **Cài đặt**

- Chọn “Run” chỗ vị trí Girvan-Newman Clustering trong tab “Statistics”

Statistics × Filters MultiMode Networks Proj... —		
Settings		
☑ <b>Network Overview</b>		
Average Degree	56.78	Run ⓘ
Avg. Weighted Degree	77.88	Run ⓘ
Network Diameter	3	Run ⓘ
Graph Density	0.285	Run ⓘ
HITS		Run ⓘ
Modularity	0.175	Run ⓘ
Clustering Coefficient		Run ⓘ
PageRank		Run ⓘ
Connected Components	1	Run ⓘ
Girvan-Newman Clustering		Run ⓘ

- Check ô “respect edge type” trong Shortest Path Betweenness -> Click “OK”



- Ta được kết quả gom cụm như sau: Số lượng cộng đồng là: 143



## Processed Graph Data

Nodes: 200

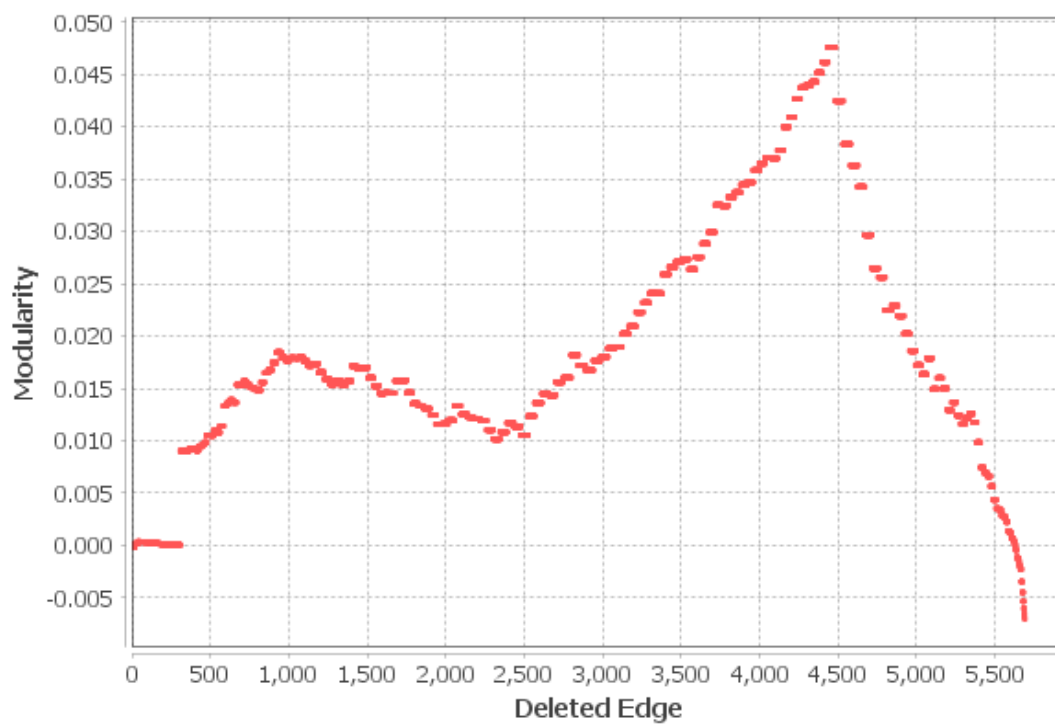
Edges 5678

Processing time: 73.543 sec.

## Communities

Number of communities: 143

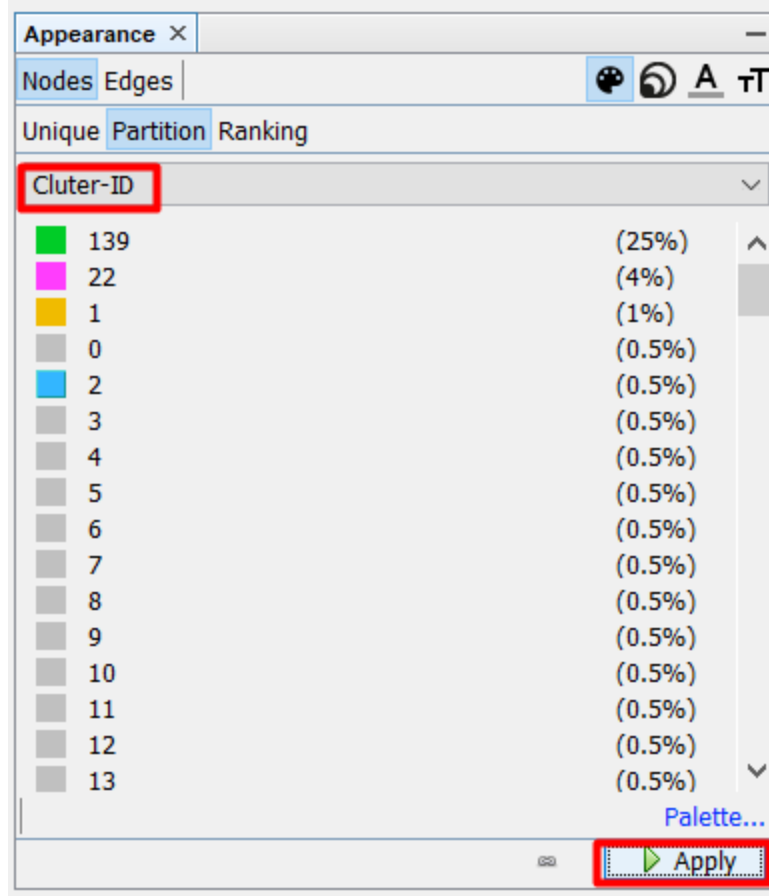
Maximum found modularity: 0.04776463



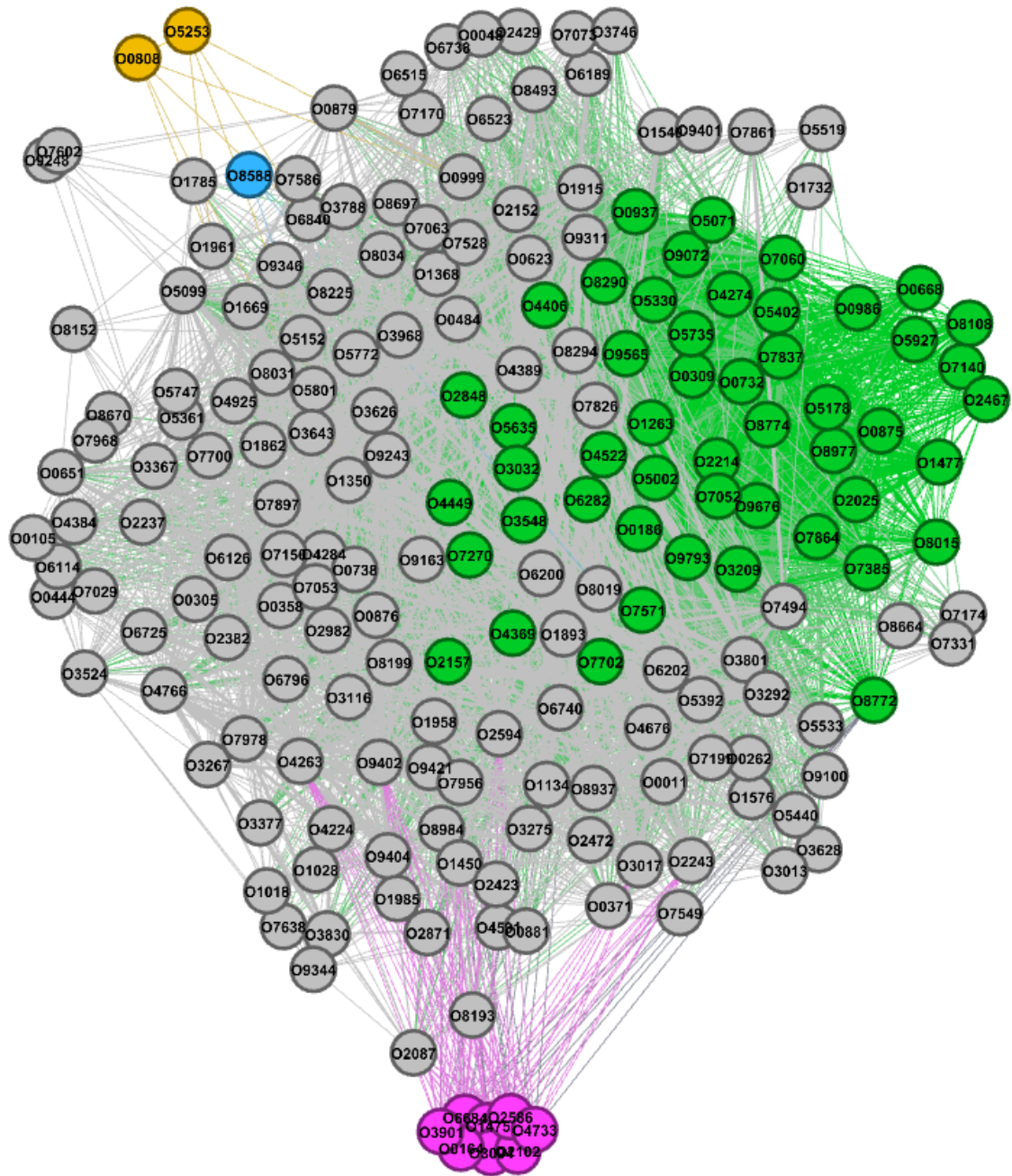
Print Copy Save

Close

- Trực quan hóa bằng đồ thị: Chọn tab Nodes > Chọn tab Partition > Chọn Cluter-ID > Chọn màu trong “Palette” > Click “Apply”



- Ta thu được đồ thị như sau:

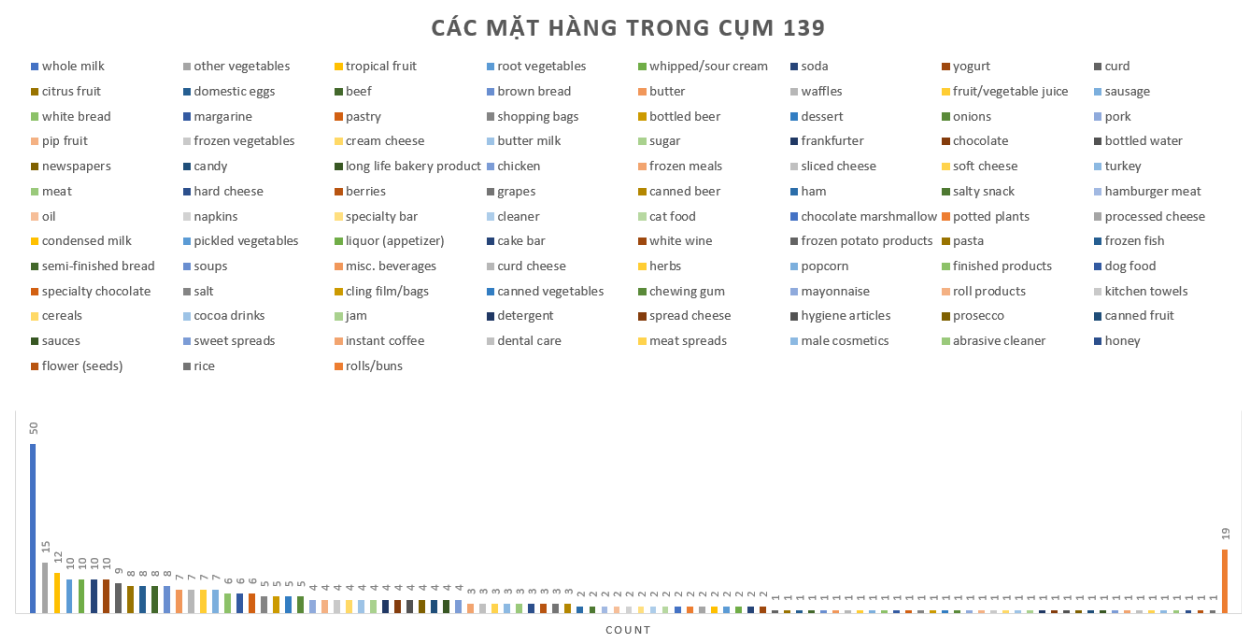


### ➤ Kết quả

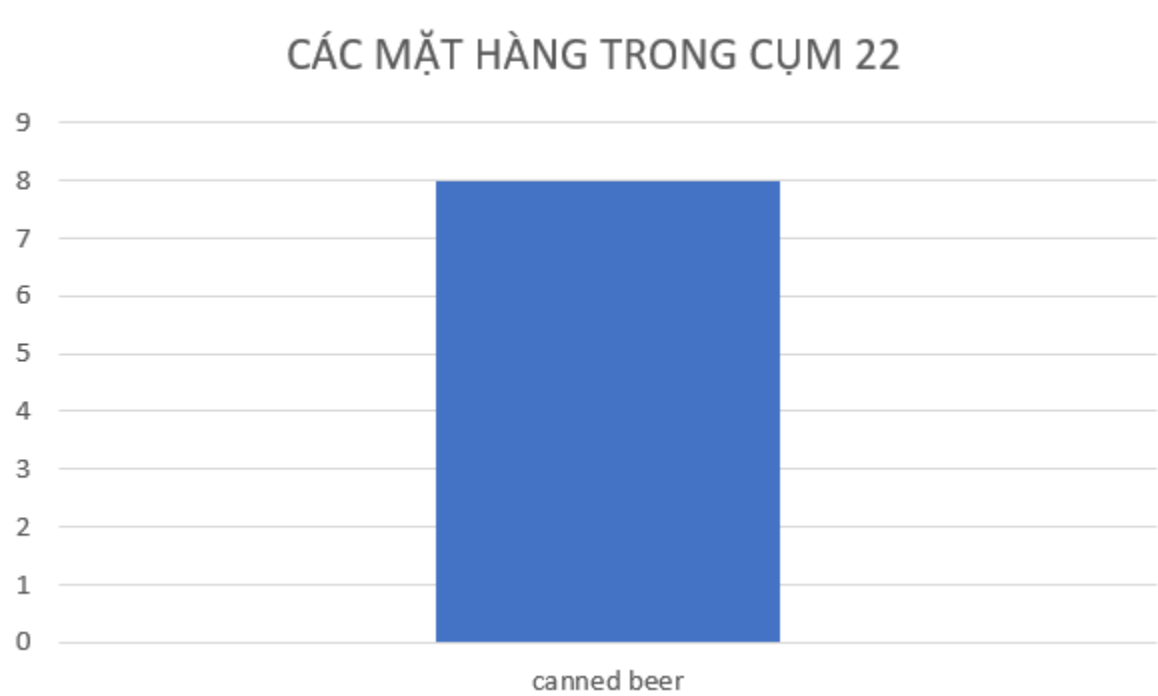
Kết quả gom cụm Girven Newman thu được 143 cụm. Trong đó:

- Cụm 139: được tô màu xanh gồm 50 node
- Cụm 22: được tô màu hồng cánh sen gồm 8 node.
- Cụm 1: được tô màu vàng đất gồm 2 node
- Cụm 2: được tô màu xanh gồm 1 node
- Các cụm còn lại, được tô màu xám, mỗi cụm gồm 1 node.

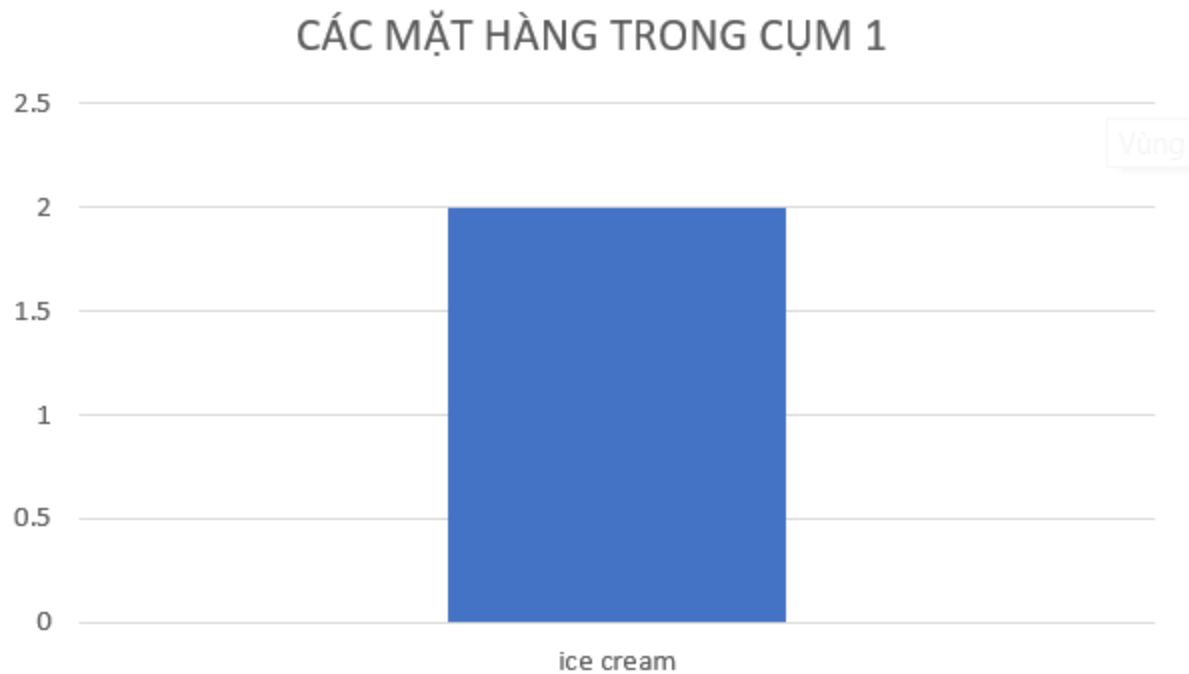
Cụm 139: whole milk



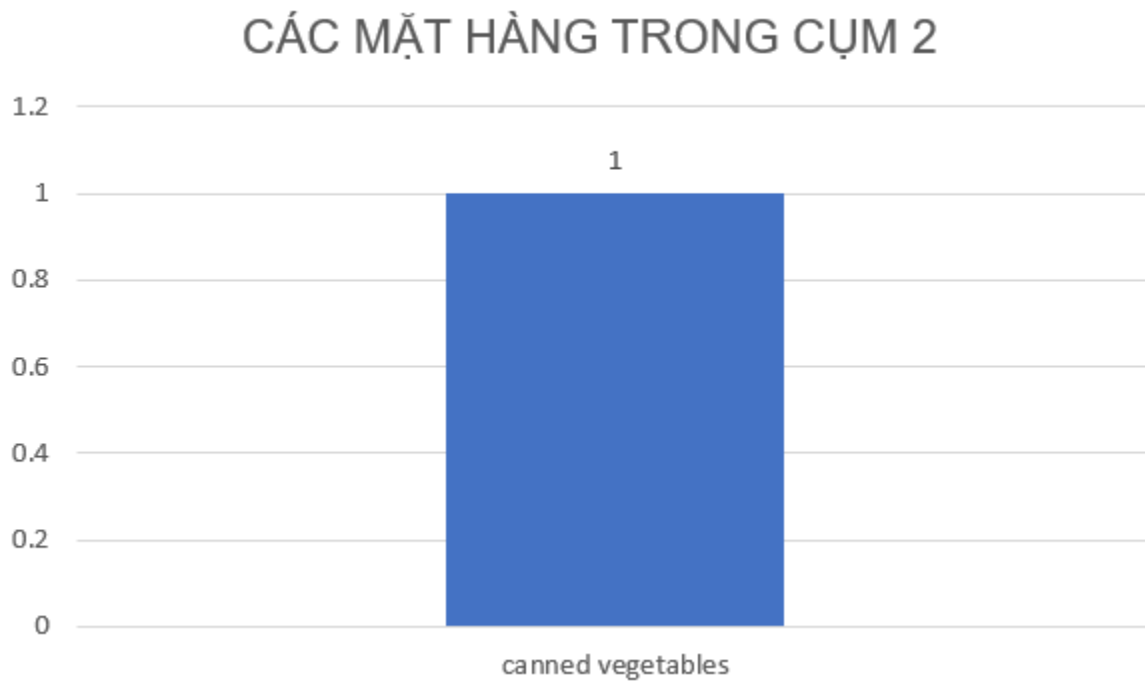
Cụm 22: canned beer



Cụm 1: ice cream



Cụm 2: canned vegetables



#### c. So sánh

Thực hiện so sánh kết quả gom cụm bằng tool Gephi và bằng code trên Jupyter.

- Gephi kết quả 143 cụm;
- Chi tiết mỗi cụm như sau

Jupyter	Gephi
Kết quả được 3 cụm. Trong đó: Cụm 0: là other vegetables, whole milk, soda, rolls/buns, yogurt. Cụm 1: ice cream Cụm 2: canned vegetables	Kết quả được 143 cụm. Trong đó: Cụm 139: whole milk Cụm 22: canned beer Cụm 1: ice cream Cụm 2: canned vegetables

Ta thấy kết quả gom khi thực hiện trên 2 phần mềm ra kết quả có vừa có sự giống nhau và khác biệt. Cụ thể:

Giống nhau: Các mặt hàng được gom chung trong cả 2 phần mềm là:

- Mặt hàng: whole milk
- Mặt hàng: ice cream
- Mặt hàng: canned vegetables

Khác nhau:

- Theo lý thuyết, thuật toán Girvan Newman sẽ chạy cho đến khi không còn cạnh nào trong đồ thị. Còn khi lập trình trên Jupyter, ta chỉ thực hiện vòng lặp 2 lần để xóa các cạnh của nó.
- Do đó, có sự khác nhau về số lượng cụm và tính chất của kết quả mỗi cụm. Nhưng chung quy lại vẫn có sự tương đồng ở kết quả khi chạy trên 2 nền tảng khác nhau.

#### d. Nhận xét

Từ việc so sánh việc gom cụm trên cả 2 phần mềm, ta có thể nhận xét với mỗi thời điểm mua hàng có 3 nhóm:

- Nhóm thứ nhất có xu hướng mua whole milk
- Nhóm thứ hai có xu hướng mua ice cream
- Nhóm thứ ba có xu hướng mua canned vegetables

## VII. Kết luận

Từ dữ liệu đầu vào là một mạng của các mặt hàng đã mua của các hóa đơn của cửa hàng tạp hóa. Ta phân tích mạng này bằng cách tính các độ đo để tìm ra các node quan trọng để tìm ra được các mặt hàng phổ có trong tập dữ liệu. Ngoài ra, ta còn thực hiện gom cụm bằng các thuật toán phát hiện cộng đồng, từ đó ta có thể xem được các mặt hàng hay được mua chung, mua kèm với nhau để đưa ra các quyết định các chiến lược marketing, thứ tự sắp xếp các mặt hàng trên kệ cho phù hợp.