**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**Logo, icon

Description automatically generated**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN HỌC: MẠNG XÃ HỘI - IS353.N11.HTCL**

**ĐỀ TÀI:**

**KHẢO SÁT ĐỒ THỊ GIỮA CÁC NHÀ PHÁT HÀNH VÀ THỂ LOẠI TRÒ CHƠI ĐIỆN TỬ MÀ HỌ PHÁT HÀNH**

**(VIDEO GAME PUBLISHERS AND THEIR GENRES)**

**Giảng viên hướng dẫn:**

**NGUYỄN THỊ KIM PHỤNG**

**Sinh viên thực hiện:**

**NGUYỄN TRÍ MINH - 19521847**

**Thành phố Hồ Chí Minh, 12/2022**

MỤC LỤC

[MỤC LỤC 3](#_Toc117713021)

[NHẬN XÉT CỦA GIẢNG VIÊN 4](#_Toc117713022)

[QUẢN LÝ TÀI LIỆU 5](#_Toc117713023)

[phân chia công việc 6](#_Toc117713024)

[PHẦN A. XỬ LÝ BẰNG phương pháp LẬP TRÌNH 7](#_Toc117713025)

[CHƯƠNG 1. mở đầu 7](#_Toc117713026)

[1.1. Giới thiệu 7](#_Toc117713027)

[1.2. Tiếp cận bài toán 7](#_Toc117713028)

[1.3. Dữ liệu 7](#_Toc117713029)

[PHẦN b. XỬ LÝ BẰNG phương pháp THỦ CÔNG 9](#_Toc117713030)

NHẬN XÉT CỦA GIẢNG VIÊN

*……., ngày……... tháng…… năm 2022*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên)*

QUẢN LÝ TÀI LIỆU

|  |  |  |  |
| --- | --- | --- | --- |
| **Thời gian** | **Phiên bản** | **Người sửa đổi** | **Chi tiết** |
| 26/10/2022 | v1.0 | Nguyễn Trí Minh | Khởi tạo tài liệu |
| 27/10/2022 | v1.1 | Nguyễn Trí Minh | Thêm nội dung và chỉnh sửa |
| 31/10/2022 | v1.2 | Nguyễn Trí Minh | Thêm nội dung và chỉnh sửa |

phân chia công việc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Nội dung công việc** | **Tỉ lệ đóng góp (%)** |
| 1 | Nguyễn Trí Minh | 19521847 | Thực hiện đồ án | 100 |

QUẢN LÝ FILE TRONG ĐỒ ÁN

Đồ án gồm 3 thư mục chính là **Documents**, **SourceCode** và **Gephi**.

1. **Documents**: Thư mục gồm các file báo cáo của đồ án.

* **19521847\_BaoCaoCuoiKy\_IS353.N11.HTCL.docx** (file hiện tại).
* **19521847\_BaoCaoCuoiKy\_IS353.N11.HTCL.pdf**.

1. **SourceCode**: Thư mục gồm các file source code, dataset gốc và dataset được export ra được sử dụng trong đồ án.

* **MainGraph.ipynb**: File source code chính (đồ thị gốc)
* **SimplerGraph.ipynb**: File source code phụ (đồ thị chỉ gồm 5 đỉnh được trích ra từ đồ thị gốc, dùng cho phần *B. Xử lý bằng phương pháp thủ công*).
* **vgsales.csv**: File dataset gốc.
* **for\_gephi.csv**: File dataset đã được export ra từ đồ thị gốc để sử dụng trong Gephi.
* **for\_simpler\_graph.csv**: File dataset đã được export ra từ đồ thị đơn giản gồm 5 đỉnh dùng phục vụ cho phần *B. Xử lý bằng phương pháp thủ công*).

1. **Gephi**: Thư mục gồm các file sử dụng trong chương trình Gephi từ đồ thị gốc và đồ thị đơn giản. Thư mục cũng gồm các hình ảnh được export ra từ Gephi và được lưu trong thư mục con **ExportedImages**.

PHẦN A. XỬ LÝ BẰNG phương pháp LẬP TRÌNH

Giới thiệu về phương pháp lập trình:

* Ngôn ngữ sử dụng: Python.
* IDE: Visual Studio Code.
* Công cụ xử lý lập trình: Jupyter Notebook
* Extension: Jupiter và Jupiter Keymap, cho phép thực thi Jupyter Notebook trực tiếp trên Visual Studio Code.

# mở đầu

## Giới thiệu

Ngày nay, nhu cầu giải trí đa phương tiện là nhu cầu không thể thiếu đối với mỗi chúng ta. Từ lâu, việc chơi các trò chơi điện tử (hay video games), vốn là một thành phần quan trọng trong nhiều thể loại giải trí đa phương tiện, luôn là một phương thức hiệu quả để giúp chúng ta giải trí sau những giờ làm việc hay học tập căng thẳng. Thị trường trò chơi điện tử luôn có nhiều biến động bởi nhiều tựa game hay, kinh điển và hấp dẫn tương ứng với nhiều thể loại game khác nhau (như Action, Sports, Platform,…) được phát triển và phát hành bởi các nhà phát hành game nổi tiếng trên khắp thế giới như Electronic Arts, Nintendo, Sony,... Với việc các tựa game nổi tiếng đó luôn được săn đón bởi các người chơi hay game thủ trên toàn cầu với thứ hạng theo doanh số bán dẫn đến các nhà phát hành game luôn phải cạnh tranh nhau trên các thể loại game nói trên thông qua các sản phẩm game mà họ phát hành. Chính vì thế, trong môn học *Mạng xã hội* này, em đã lựa chọn đề tài nói trên và áp dụng, sử dụng những kiến thức đã được học trong môn học để có thể khảo sát và phân tích **đồ thị về các nhà phát hành với thể loại game mà họ phát hành trên thị trường game** thông qua các thuật toán gom cụm, xếp hạng đỉnh và các độ đo trong đồ thị.

Các phương pháp và thuật toán sử dụng trong đồ án này ở phần lập trình bao gồm:

* Thuật toán Louvain để xác định, phát hiện các cộng đồng trong đồ thị (community detection) thông qua gom cụm các node có liên quan.
* Thuật toán PageRank và Eigenvector Centrality dùng để xếp hạng các đỉnh trong đồ thị.
* Tính toán độ đo Betweeness centrality ở các đỉnh (nodes) và cạnh (edges).
* Tính toán độ đo Closeness centrality ở các đỉnh (nodes).

## Dữ liệu

* + 1. **Giới thiệu nguồn dữ liệu**
* Nguồn dataset: *Video Game Sales*, <https://www.kaggle.com/datasets/gregorut/videogamesales>.
* Dữ liệu *Video Game Sales* được cung cấp trên nền tảng Kaggle với mỗi dòng dữ liệu là 1 tựa game tương ứng với 1 thể loại game duy nhất do 1 nhà phát hành cho ra mắt. Các tựa game được phát hành (các dòng dữ liệu) được sắp xếp trong tập dữ liệu với thứ hạng dựa vào doanh số bán trong nhiều năm.
* Để rõ hơn, sau đây là thống kê của tập dữ liệu gồm 16598 dòng với 11 thuộc tính bao gồm:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | Rank | int | Thứ hạng của game |
| 2 | Name | char | Tên game |
| 3 | Platform | char | Hệ máy |
| 4 | Year | int | Năm phát hành |
| 5 | Genre | char | Thể loại game |
| 6 | Publisher | char | Nhà phát hành |
| 7 | NA\_Sales | float | Doanh số khu vực Bắc Mĩ |
| 8 | EU\_Sales | float | Doanh số khu vực Châu Âu |
| 9 | JP\_Sales | float | Doanh số ở Nhật Bản |
| 10 | Other\_Sales | float | Doanh số khu vực khác |
| 11 | Global\_Sales | float | Doanh số toàn cầu |

* Ví dụ sơ lược về một dòng dữ liệu (1 tựa game) trong tập dữ liệu nói trên là tựa game *Pokémon Platinum Version* thuộc thể loại *Role-Playing* do nhà phát hành *Nintendo* cho ra mắt vào năm 2008.

Graphical user interface, application

Description automatically generated

* + 1. **Sử dụng dữ liệu nào cho bài toán đồ thị?**
* Vì một thể loại game có thể thuộc về nhiều tựa game do một nhà phát hành phát hành, do dữ liệu trên là rất lớn nên để cho đơn giản trong xử lý của phạm vi đồ án ta sẽ lọc dữ liệu để **1 nhà phát hành** trong **1 thể loại** thì chỉ phát hành **đúng 1 tựa game duy nhất**. Các nhà phát hành khác nhau thì có nhiều thể loại khác nhau và chúng sẽ **chung nhau về một số thể loại**, đây là quan hệ mấu chốt trong đồ thị và sẽ được trình bày cụ thể hơn bên dưới.
* Từ tập dữ liệu và vấn đề đã đặt ra ở trên, do ta chỉ cần quan tâm tới *Nhà phát hành* (*Publisher*) và *Thể loại* (*Genre*) của tựa game trên mỗi dòng dữ liệu nên ta sẽ giữ lại duy nhất 2 cột thuộc tính là **Publisher** và **Genre** để phân tích, còn lại sẽ lượt bỏ đi hết.

# PHÂN TÍCH ĐỒ THỊ

File source code sử dụng ở chương này: **MainGraph.ipynb**.

Dataset sử dụng ở chương này: **vgsales.csv**.

File của phần mềm Gephi sử dụng ở chương này: **MainGraph.gephi** với dataset **for\_gephi.csv**.

**Lưu ý:** Các hình ảnh kết quả bên dưới (output) có thể khác so với trong file souce code vì có thể source code được compile lại nhiều lần sau khi tài liệu được hoàn thành. Lý do có sự khác biệt giữa các lần compile là do tính chất ngẫu nhiên trong một vài thuật toán dẫn đến sự khác nhau của các output, ví dụ như Louvain.

## Sơ lược về dataframe

Như đã phân tích ở chương trước, ta chỉ cần 2 cột **Publisher** và **Genre** trong dataset gốc, vì vậy ta sẽ giữ lại 2 cột này. Sau đó ta sẽ bỏ đi tất cả các hàng có dữ liệu rỗng (*NA*, đó là các hàng mà không có dữ liệu của nhà phát hành cụ thể) và trùng lặp nhau (đó là các hàng mà cho thấy nhà phát hành đó xuất bản về 1 thể loại nào đó nhiều lần, vì ta không cần sự lặp đi lặp lại đó như đã nói ở bên trên nên ta loại bỏ sự trùng lặp đó đi) cho đơn giản hơn để xử lý.

Graphical user interface, application

Description automatically generated

Sau khi xử lý ta được 1 dataframe mới gồm 1825 dòng và 2 cột (*Publisher*, *Genre*). Một dòng dữ liệu khi đó cho biết **1 nhà phát hành game có 1 thể loại cụ thể nào đó** (không lặp lại).

## Vẽ đồ thị

Tiếp theo, đưa dataframe vào đồ thị 2 phía (bipartite graph) và hiển thị một số thông tin về đồ thị 2 phía:

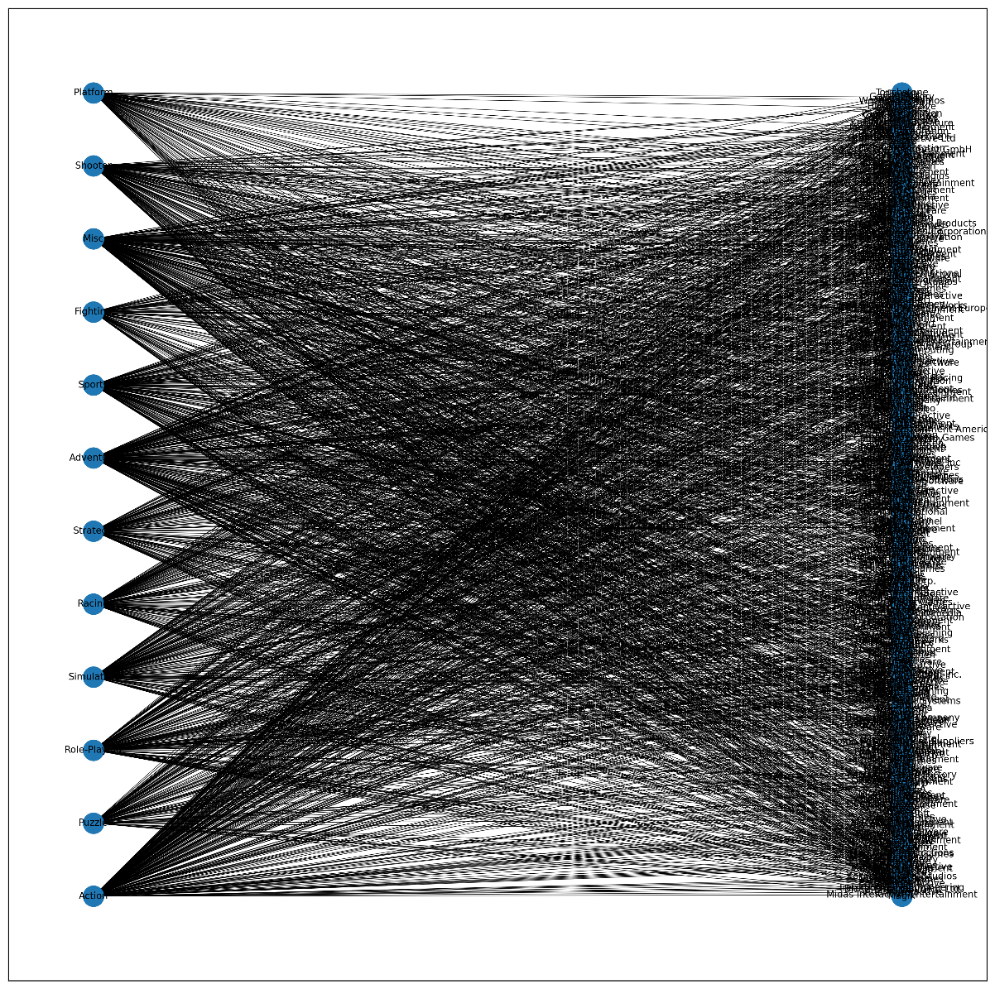


Ta thấy có 2 loại node là *Publisher* với 578 node và *Genre* với 12 node. Các cạnh trong đồ thị 2 phía tính được là 1825 cạnh (tương đương số hàng trong dataframe).

**Ý nghĩa**: Cạnh giữa 1 node nhà phát hành và 1 node thể loại thể hiện cho 1 dòng dữ liệu mà thể loại đó được nhà phát hành đó sản xuất.

Thực hiện vẽ đồ thị 2 phía:





*Hình 1. Đồ thị 2 phía giữa Genres và Publishers*

Thực hiện vẽ tiếp đồ thị 1 phía (non-bipartite graph) với các node là các *Publisher* từ dữ liệu của đồ thị 2 phía. Trước tiên ta cũng phải bỏ đi sự lặp đi lặp lại của các *Publisher* trong các hàng dữ liệu của dataframe vì dữ liệu của các node *Publisher* đưa vào đồ thị 1 phía phải duy nhất và phân biệt (tức là 578 node đã đề cập phía trên):



Map

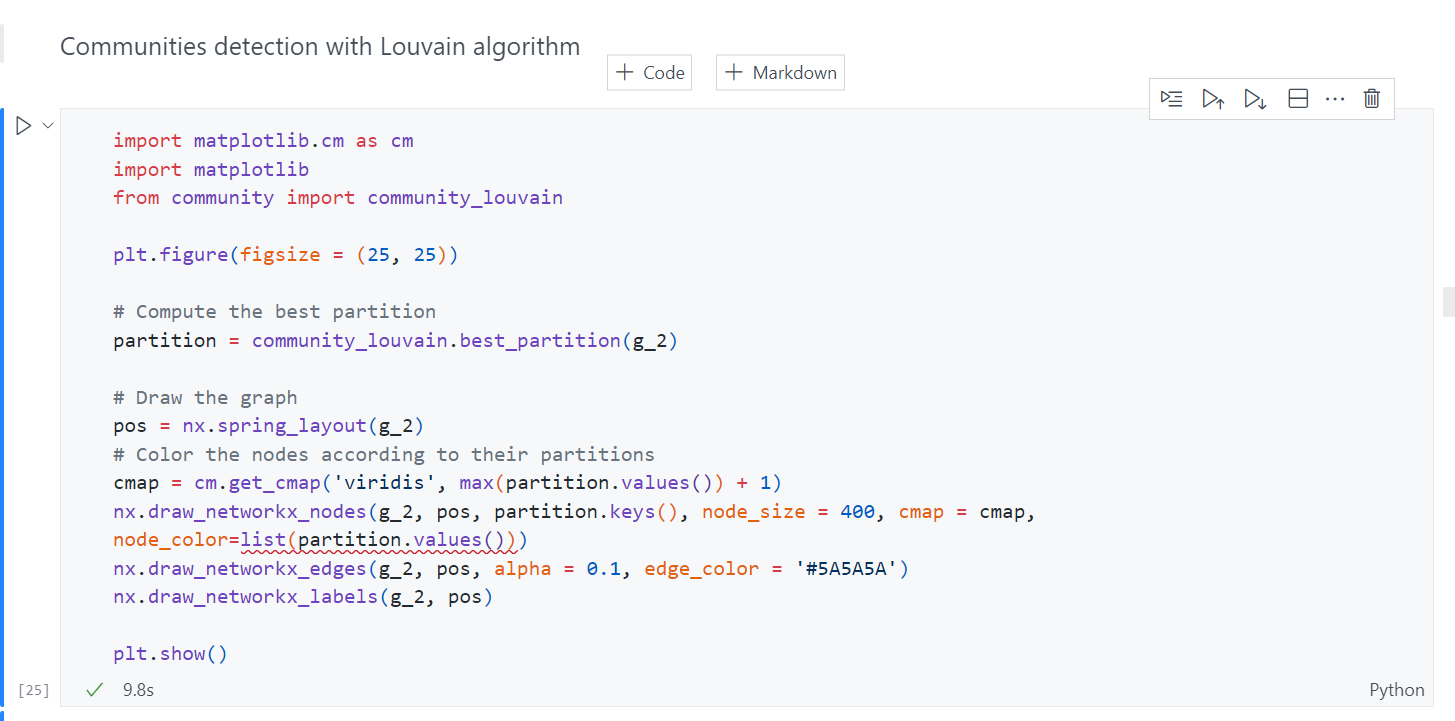
Description automatically generated with medium confidence

*Hình 2. Đồ thị 1 phía giữa các Publisher*

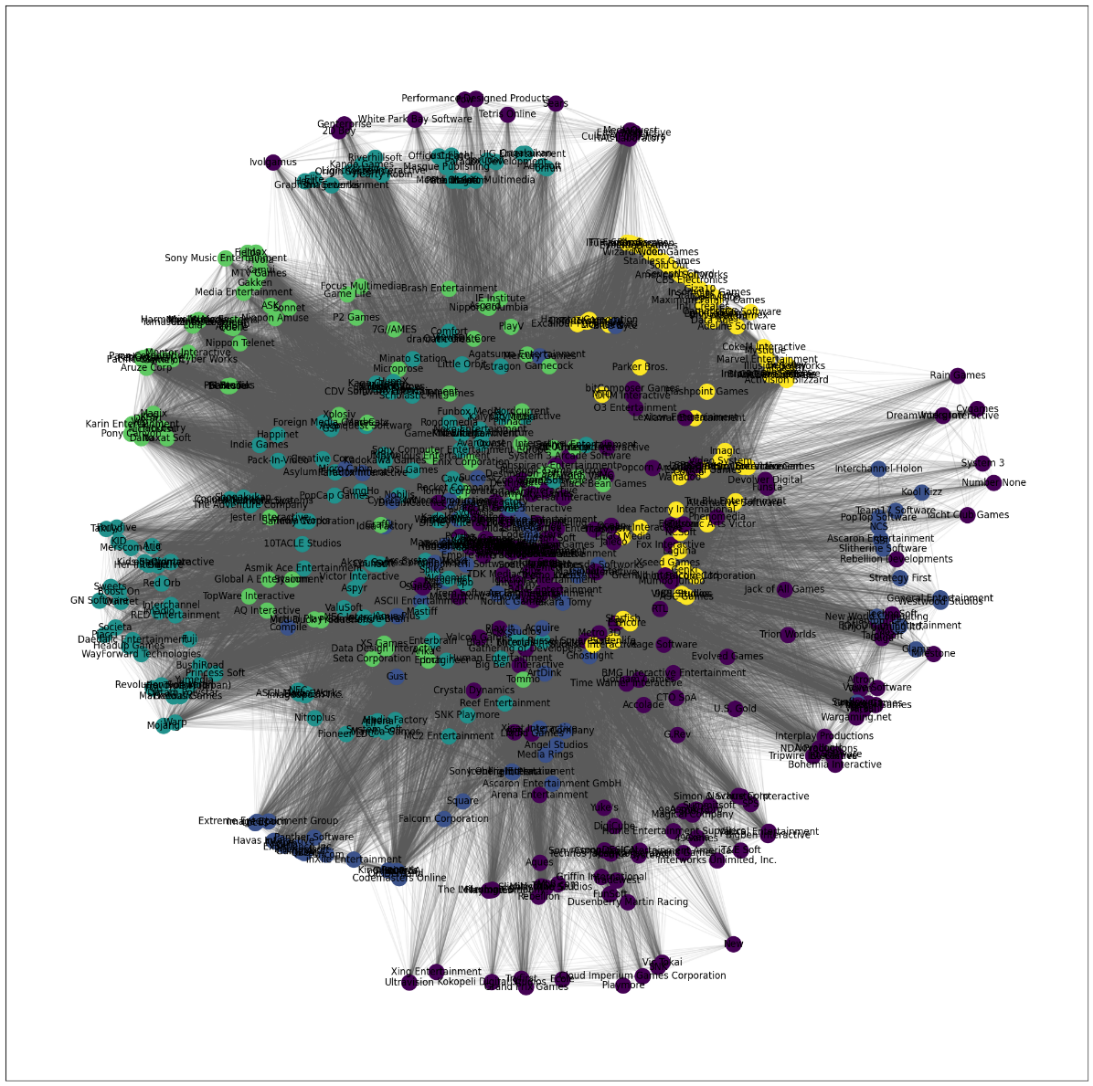
**Ý nghĩa**: 1 cạnh giữa 2 node nhà phát hành cho biết giữa 2 nhà phát hành đó có chung 1 thể loại game mà họ sản xuất. 2 nhà phát hành game có thể có chung nhiều thể loại khác nhau, số lượng các thể loại chung nhau đó chính là **trọng số** của cạnh giữa 2 node đó trong đồ thị.

## Gom cụm đồ thị bằng thuật toán phát hiện cộng đồng Louvain

Đưa đồ thị 1 phía vào thuật toán Louvain để thực hiện gom cụm các node *Publisher*:



Kết quả từ thuật toán với đồ thị mới có các màu tương ứng với các cụm riêng biệt:



*Hình 3. Các cụm được phân chia bằng giải thuật Louvain*

Tính số lượng các cộng đồng sau khi chạy thuật toán, ta thấy có được **5** cộng đồng:



Đồng thời in ra các nhà phát hành tương ứng trong từng cộng đồng (bên dưới là trích lượt vì output thực tế rất dài):

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

…

Graphical user interface, text, application

Description automatically generated

…



…

Graphical user interface, text, application

Description automatically generated

…

Graphical user interface, text, application

Description automatically generated

Do rất khó để có thể biết được ý nghĩa của các cụm trong đồ thị nên ta sử dụng 1 kỹ thuật là phân tích số lượng các node (hay các *Publisher*) cùng có các thể loại nào chung trong toàn bộ các thể loại ở mỗi một cụm, hay nói cách khác là phân tích độ phổ biến của các thể loại game trong mỗi cụm của đồ thị.

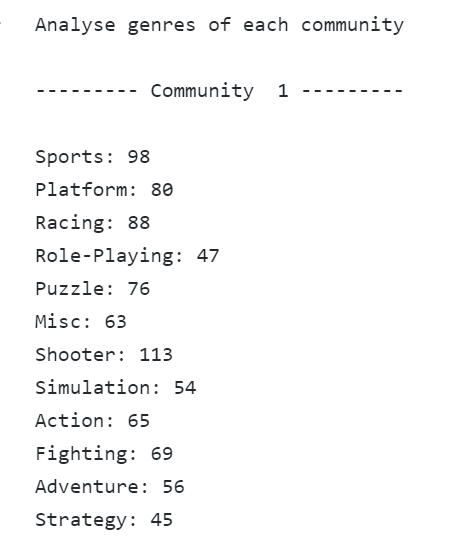
Ta tiến hành thực hiện đoạn code sau để phân tích lần lượt qua tất cả các cụm của đồ thị:



Text

Description automatically generated

Kết quả ở mỗi cụm sẽ cho biết độ phổ biến của tất cả các thể loại game có trong cụm đó bằng số lượng các nhà phát hành nào đã xuất bản thể loại game đó. Ví dụ sau đây là kết quả ở *Cụm 1*:



Đồ thị biểu thị kết quả trên:

Chart, bar chart

Description automatically generated

Ta thấy ở trong *Cụm 1* thì trong tất cả các thể loại game đã có đến **113** nhà phát hành đã xuất bản thể loại **Shooter**.

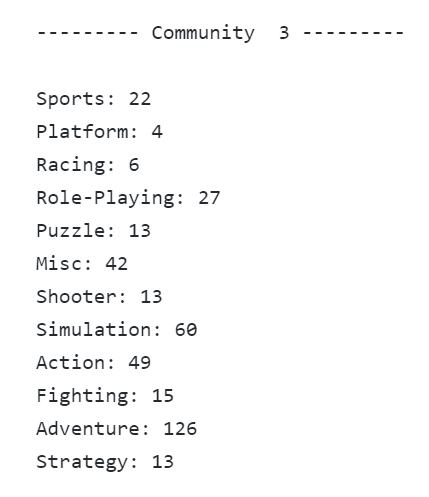
Thực hiện tương tự với các cụm còn lại, ta có các kết quả sau:



Chart, bar chart

Description automatically generated

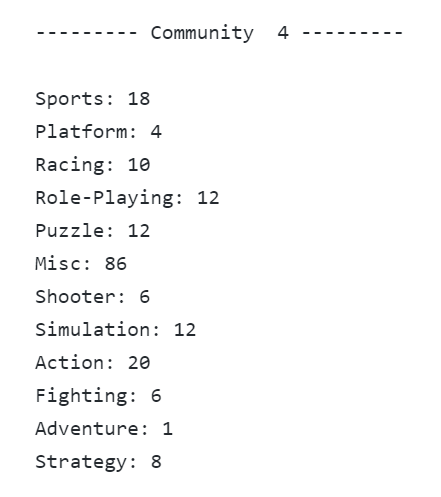
Ta thấy ở trong *Cụm 2* thì trong tất cả các thể loại game đã có đến **62** nhà phát hành đã xuất bản thể loại **Role-Playing**.



Chart, bar chart

Description automatically generated

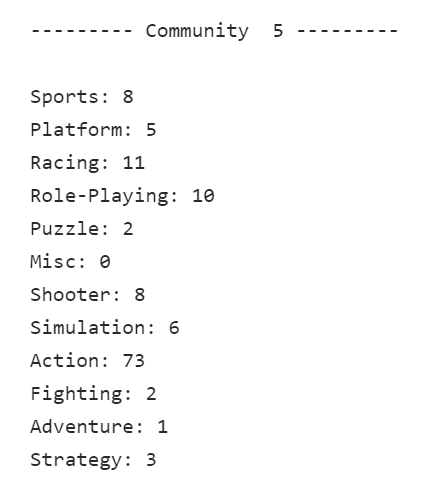
Ta thấy ở trong *Cụm 3* thì trong tất cả các thể loại game đã có đến **126** nhà phát hành đã xuất bản thể loại **Adventure**.



Chart, bar chart

Description automatically generated

Ta thấy ở trong *Cụm 4* thì trong tất cả các thể loại game đã có đến **86** nhà phát hành đã xuất bản thể loại **Misc**.



Chart, bar chart, waterfall chart

Description automatically generated

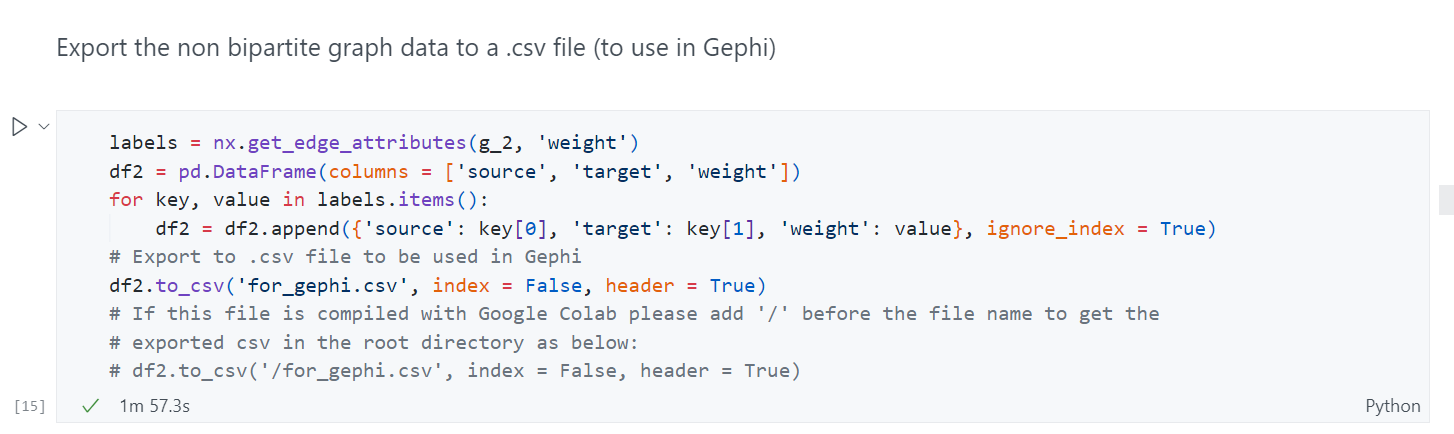
Ta thấy ở trong *Cụm 5* thì trong tất cả các thể loại game đã có đến **73** nhà phát hành đã xuất bản thể loại **Action**.

Từ các thông số trên khi so sánh giữa các cụm với nhau, ta rút ra được ý nghĩa sau:

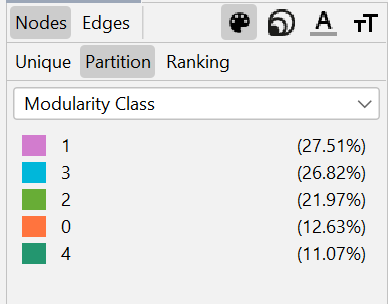
**Ý nghĩa**:

* *Cụm 1* bao gồm các nhà phát hành như **Nintendo**, **Microsoft Game Studios**, **Take-Two Interactive**, **Sony Computer Entertainment**, **Activision**,… là cụm mà tập trung nhiều nhà phát hành xuất bản thể loại **Shooter**, mặc dù các thể loại khác có kết quả cũng khá cao và trải dài hơn so với các cụm khác, chứng tỏ đây là cụm có nhiều nhà phát hành sôi nổi nhất hay nói cách khác là cụm có các nhà phát hành có tiếng tăm trong thị trường.
* *Cụm 2* bao gồm các nhà phát hành như **Bethesda Softworks**, **Square Enix**, **LucasArts**, **Codemasters**, **Level 5**,… là cụm mà tập trung nhiều nhà phát hành xuất bản thể loại **Role-Playing**.
* *Cụm 3* bao gồm các nhà phát hành như **Sony Computer Entertainment Europe**, **Red Orb**, **Maxis**, **GungHo**, **Alchemist**,… là cụm mà tập trung nhiều nhà phát hành xuất bản thể loại **Adventure**.
* *Cụm 4* bao gồm các nhà phát hành như **RedOctane**, **Enix Corporation**, **MTV Games**, **Activision Value**, **Oxygen Interactive**,… là cụm mà tập trung nhiều nhà phát hành xuất bản thể loại **Misc**.
* *Cụm 5* bao gồm các nhà phát hành như **Palcom**, **989 Studios**, **NCSoft**, **Parker Bros.**, **Imagic**,… là cụm mà tập trung nhiều nhà phát hành xuất bản thể loại **Action**.

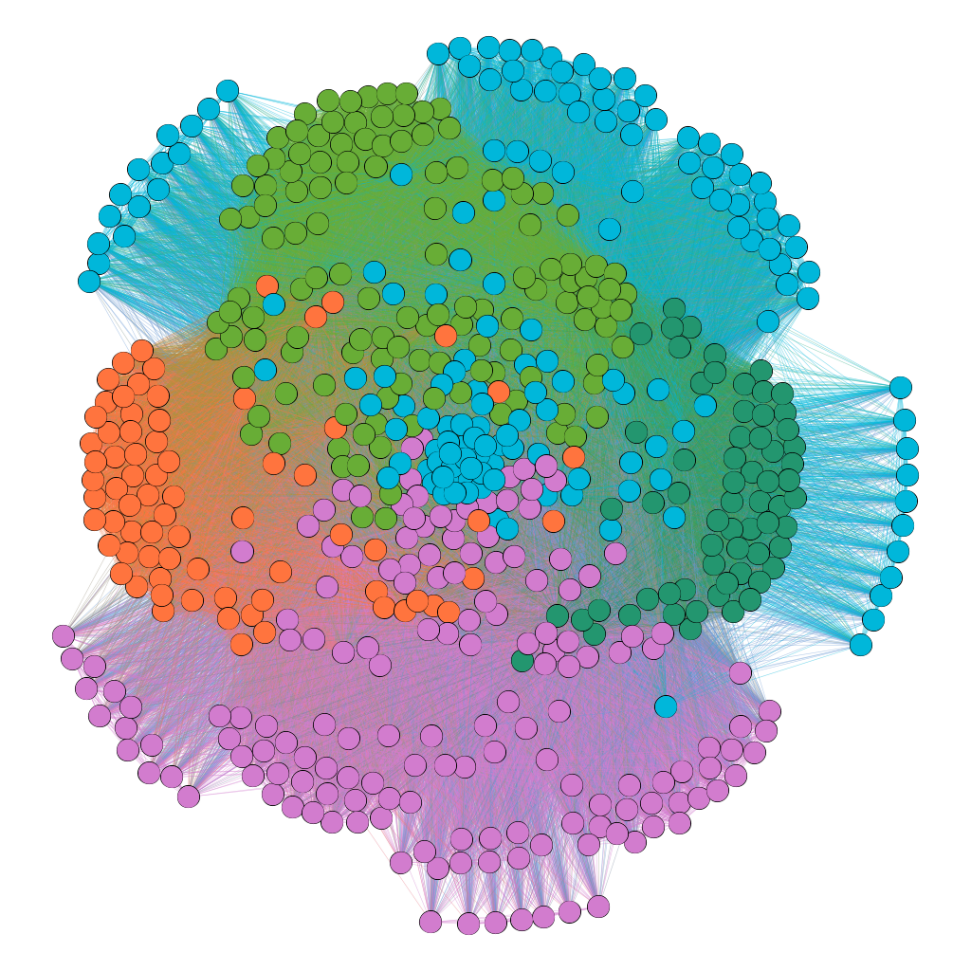
Xuất dữ liệu của đồ thị 1 phía ra 1 file dataset **for\_gephi.csv** để sử dụng cho việc trực quan hoá trong phần mềm *Gephi*. Vì đồ án thực hiện trên *Jupyter Notebook* trực tiếp trên *Visual Studio Code* nên file .csv sau khi export sẽ nằm cùng với source code ở thư mục chứa source code. Có 1 lưu ý nhỏ là nếu thực thi trên *Google Colab* thì cần thêm “/” vào trước tên file .csv để file sau khi được export ra sẽ được nằm ở thư mục root (thư mục mẹ) của project *Google Cola*b:



Sau khi mở file .csv bằng *Gephi*, tiến hành định dạng lại đồ thị và sử dụng tính năng *Community Detection* của *Gephi* bằng *Modularity*, chính là thuật toán Louvain và chờ các partition được phân chia ra. Ta thấy cũng có **5** partition được chia ra tương ứng với **5** cụm hay **5** cộng đồng.

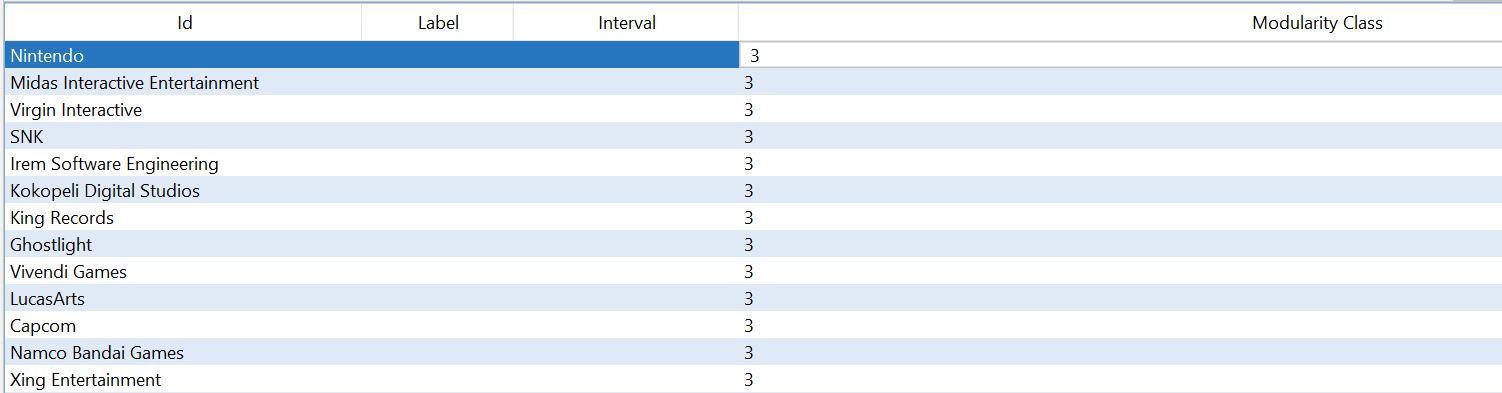


Sau khi thực hiện cùng thuật toán gom cụm Louvain trong phần mềm *Gephi*, kết quả phân cụm được trực quan hoá trên đồ thị như sau với mỗi cụm là 1 màu khác nhau:



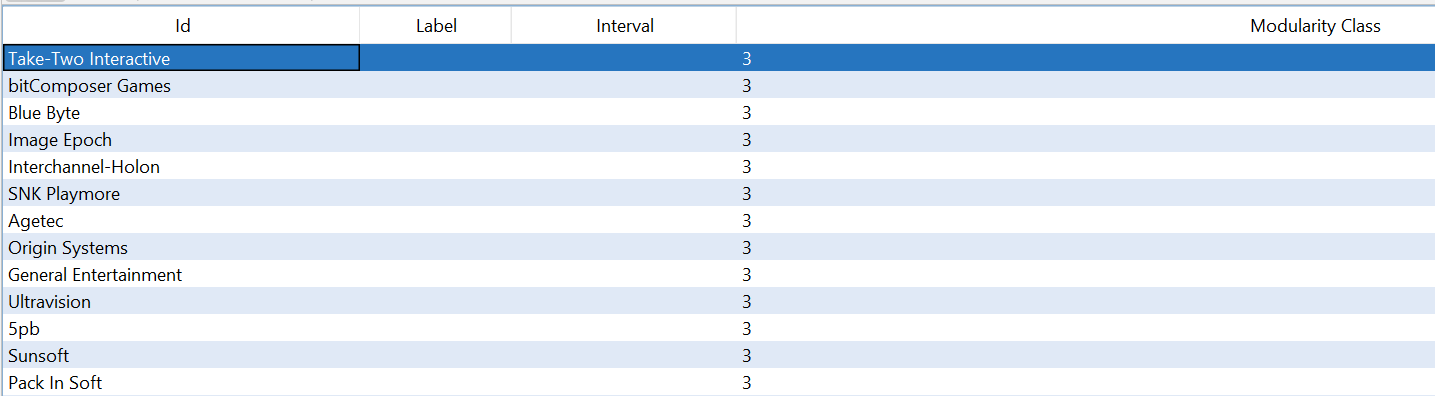
*Hình 4. Các cộng đồng được phát hiện bằng giải thuật Louvain trong Gephi*

Tiếp theo, vào cơ sở dữ liệu của đồ thị trong *Gephi* để kiểm tra thử xem các phần tử trong 1 cụm có giống như các phần tử mà ta đã khảo sát được bằng việc lập trình bằng Python hay không. Ở đây sẽ kiểm tra thử cộng đồng chứa nhà phát hành game **Nintendo**, là cộng đồng có các nhà phát hành tiếng tăm, chính là *Cụm 1* ta khảo sát bằng phương pháp lập trình bên trên. Ở trong *Gephi*, cộng đồng này có *ID* là 3. Tìm một vài phần tử khác cũng có *ID* thuộc *ID* cộng đồng là 3 thì ta thấy đều có sự xuất hiện của **Microsoft Game Studios**, **Take-Two Interactive**, **Sony Computer Entertainment** hay **Activision**,... Chứng tỏ việc chạy thuật toán Louvain bằng Python cho kết quả gần như tương đồng so với kết quả trong *Gephi* cho ra.



Table

Description automatically generated



Table

Description automatically generated

Table

Description automatically generated

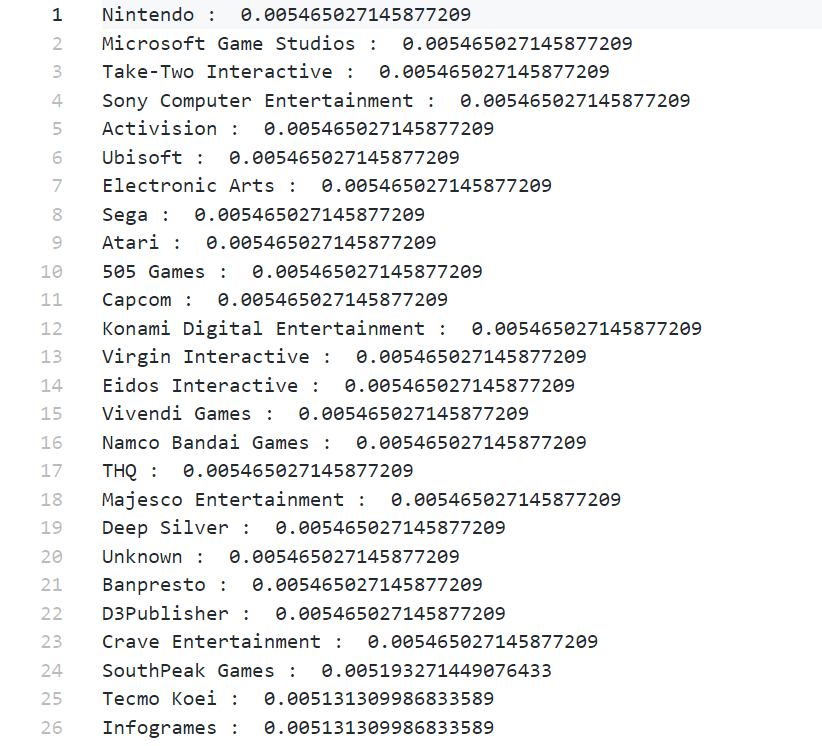
## Thuật toán xếp hạng đỉnh

### Thuật toán PageRank

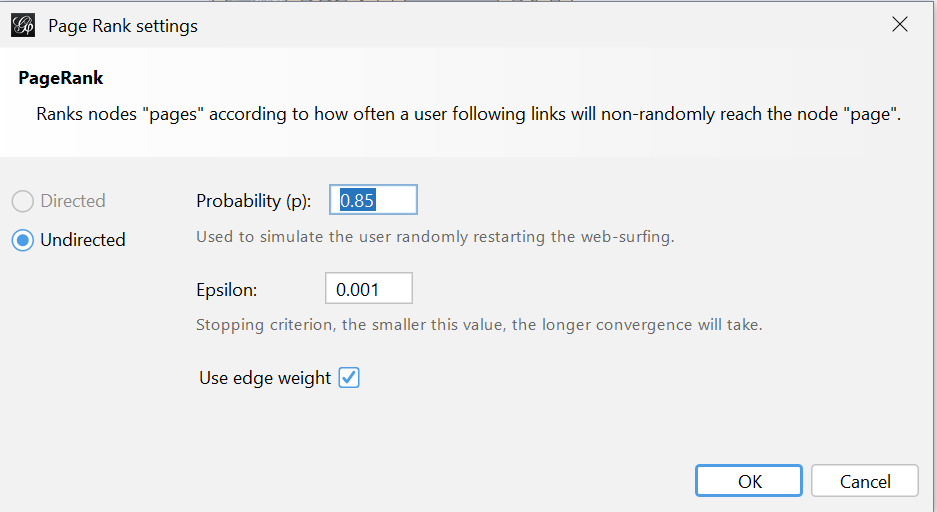
Kết quả lập trình trên Python sau khi chạy thuật toán PageRank (với xếp hạng các đỉnh các nhà phát hành theo thứ tự giảm dần được trích lượt):

Graphical user interface, application

Description automatically generated



Trên *Gephi*, thiết lập thuật toán PageRank với thông số mặc định *Probability* là 0.85 và *Epsilon* là 0.001. Đặc biệt lưu ý tick vào **Use edge weight** để *Gephi* tính PageRank theo **trọng số cạnh**, tương tự như trong Python.



Kết quả chạy thuật toán PageRank trên phần mềm *Gephi*:

Graphical user interface, table

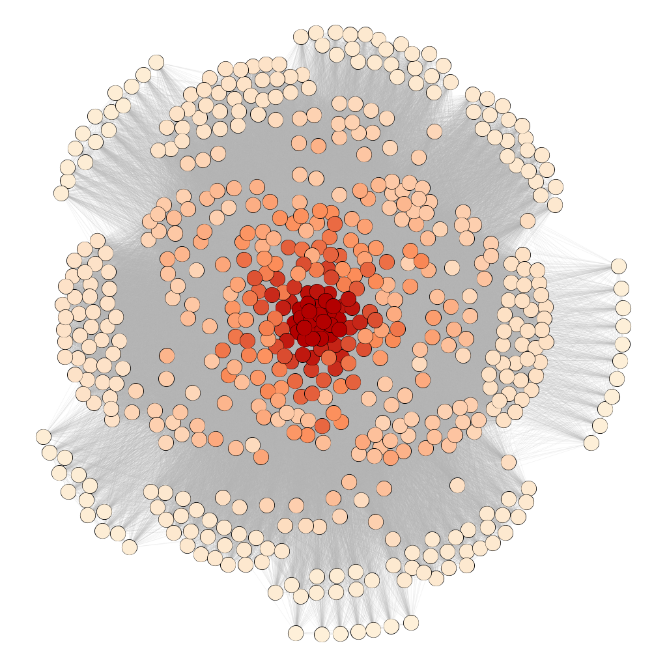
Description automatically generated

Table

Description automatically generated

**Nhận xét:** Do tên của các node, hay các nhà phát hành được sắp xếp theo các cách khác nhau tuỳ theo Python hay *Gephi* nên về hình thức sắp xếp trông có vẻ khác nhau nhưng thực tế thứ hạng là duy nhất dựa vào kết quả của PageRank. Mặt khác ta nhận thấy kết quả PageRank của mỗi node đều tương đối tương đồng và chính xác ở trên cả Python và *Gephi*. Ví dụ như các node từ đầu đến node trước node **SouthPeak Games** là có cùng thứ hạng với nhau do cùng có kết quả là **0.005465** trên cả Python và *Gephi*. Đến node **SouthPeak Games** thì đã có kết quả nhỏ hơn là **0.005193** nên node này được sắp bên dưới các node trên.

Đồ thị PageRank trong *Gephi* cho biết các node có màu càng đậm thì thứ hạng PageRank càng cao (thường là các node trung tâm):



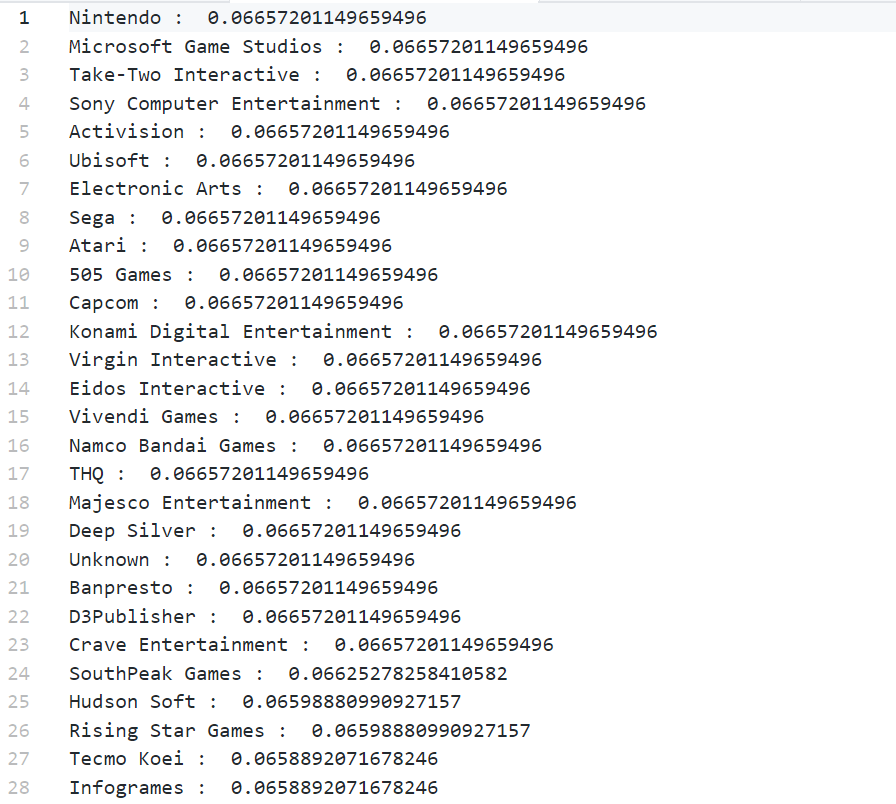
*Hình 5. Đồ thị PageRank trong Gephi*

### Thuật toán Eigenvector centrality

Chạy thuật toán tính Eigenvector centrality của đồ thị trên Python:



Kết quả sau khi chạy thuật toán trên Python:



Kết quả sau khi chạy thuật toán trên *Gephi*:

Graphical user interface, table

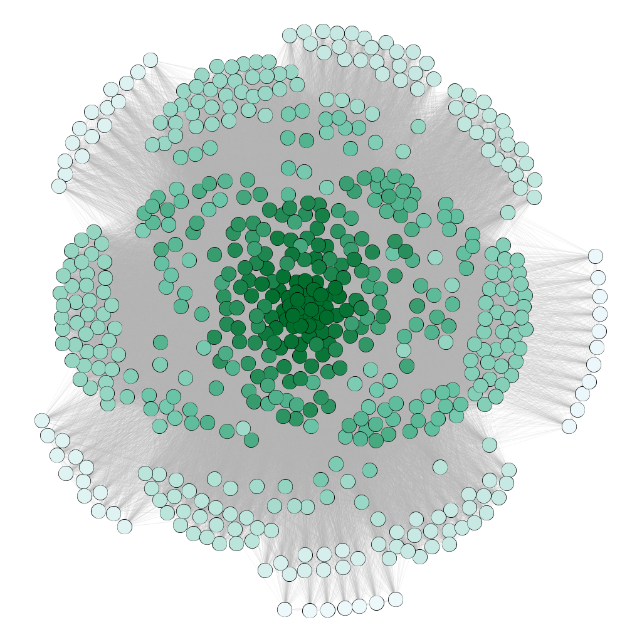
Description automatically generated

Table

Description automatically generated

**Nhận xét:** Như đã nói ở phần PageRank, do cách sắp xếp khác nhau nên tên các node được sắp xếp khác nhau trên Python và *Gephi*. Tuy nhiên ta nhận thấy kết quả Eigenvector centrality giữa Python và *Gephi* có sự khác biệt đó là do kết quả trên *Gephi* là kết quả nguyên gốc của vector còn thuật toán trong method **eigenvector\_centrality()** của **NetworkX** đã sử dụng kỹ thuật **normalise vector** nên vector kết quả đã trải qua quá trình normalisation nên khác với *Gephi*. Ví dụ trên Python thì node **SouthPeak Games** có kết quả là **0.06625278258410582** còn trên *Gephi* là **0.9950900050961164**, các node khác hạng cao hơn bên trên trong Python có kết quả là **0.06657201149659496** còn trong *Gephi* thì đều là 1.0, để kiểm chứng vector đã được normalise ta lấy **1.0 / 0.06657201149659496 = 15.02132769**, sau đó lấy **15.02132769 \* 0.06625278258410582 = 0.9952047576**, thấy gần xấp xỉ độ lớn vector trên *Gephi*, chứng tỏ kết quả trên Python đều đã được normalise đi một lượng là **15.02132769**. Như vậy kết quả là không sai, chỉ là do trải qua phương pháp vector normalisation, và tương tự như PageRank, xếp hạng của các node vẫn rất chính xác và tương đồng ở trên Python và cả *Gephi*.

Đồ thị Eigenvector centrality trong *Gephi* cho biết các node có màu càng đậm thì thứ hạng Eigenvector centrality càng cao (thường là các node trung tâm):



*Hình 6. Đồ thị Eigenvector centrality trong Gephi*

### Ý nghĩa của các thuật toán xếp hạng đỉnh

* Trong một đồ thị thì các thuật toán xếp hạng đỉnh như PageRank hay Eigenvector centrality cho biết thứ hạng của các node được đánh gia thông qua sự “uy tín” hay “tín nhiệm” của các node đó trong đồ thị. Vì mỗi node đều có sự quan hệ ra vào với các node xung quanh, nên node nào càng nhận được chiều quan hệ vào nó nhiều từ các node khác thì thứ hạng càng cao, chứng tỏ đó là những node đó có độ quan trọng đáng kể trong đồ thị.
* Trong bài toán cụ thể của đồ án, ta có thể đánh giá là nhà phát hành nào có thứ hạng càng cao chứng tỏ càng có khả năng tiếp cận dễ dàng với nhiều nhà phát hành khác xung quanh thông qua quan hệ trên việc có chung thể loại game, hay nói cách khác là có tỉ lệ chung thể loại game với nhiều nhà phát hành là cao hơn các node khác có thứ hạng thấp hơn. Chính vì điều đó nên các node nhà phát hành có thứ hạng cao này luôn luôn ở phần trung tâm của đồ thị nên luôn dễ dàng tiếp cận được nhiều node nhà phát hành khác xung quanh, hay nói cách khác họ là những nhà phát hành game lớn có mức độ cạnh tranh trên thể loại game với rất nhiều nhà phát hành khác, ví dụ như **Nintendo**, **Microsoft Game Studios**, **Take-Two Interactive**, **Sony Computer Entertainment** hay **Activision**.

## Tính toán các độ đo

### Betweeness

* **Theo node (node-based):**
* **Theo cạnh (path-based, edge-based):**

### Closeness

PHẦN b. XỬ LÝ BẰNG phương pháp THỦ CÔNG