

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



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

**Giảng viên hướng dẫn:**

**NGUYỄN THỊ KIM PHUNG**

**Sinh viên thực hiện:**

**NGUYỄN TRÍ MINH - 19521847**

**Thành phố Hồ Chí Minh, 12/2022**

## MỤC LỤC

<b>MỤC LỤC.....</b>	<b>2</b>
<b>NHẬN XÉT CỦA GIẢNG VIÊN .....</b>	<b>4</b>
<b>LỜI CẢM ƠN.....</b>	<b>5</b>
<b>PHÂN CHIA CÔNG VIỆC.....</b>	<b>6</b>
<b>QUẢN LÝ FILE TRONG ĐỒ ÁN .....</b>	<b>7</b>
<b>PHẦN A. XỬ LÝ BẰNG PHƯƠNG PHÁP LẬP TRÌNH .....</b>	<b>8</b>
<b>CHƯƠNG 1. MỞ ĐẦU.....</b>	<b>8</b>
1.1. Giới thiệu.....	8
1.2. Dữ liệu .....	8
1.2.1. Giới thiệu nguồn dữ liệu	8
1.2.2. Sử dụng dữ liệu nào cho bài toán đồ thị?	9
<b>CHƯƠNG 2. PHÂN TÍCH ĐỒ THỊ .....</b>	<b>10</b>
2.1. Sơ lược về dataframe .....	10
2.2. Vẽ đồ thị .....	11
2.3. Gom cụm đồ thị bằng thuật toán phát hiện cộng đồng Louvain .....	14
2.4. Thuật toán xếp hạng đỉnh.....	23
2.4.1. Thuật toán PageRank	23
2.4.2. Thuật toán Eigenvector Centrality	26
2.4.3. Ý nghĩa của các thuật toán xếp hạng đỉnh	28
2.5. Tính toán các độ đo .....	28
2.5.1. Betweenness Centrality	28
2.5.2. Closeness Centrality	35

PHẦN B. XỬ LÝ BẰNG PHƯƠNG PHÁP THỦ CÔNG .....	38
KẾT LUẬN VỀ ĐỀ TÀI.....	51
TÀI LIỆU THAM KHẢO .....	52

## NHẬN XÉT CỦA GIẢNG VIÊN

....., ngày ..... tháng ..... năm 2022

## **Người nhận xét**

(Ký tên và ghi rõ họ tên)

## LỜI CẢM ƠN

Trước khi đi vào nội dung phần báo cáo đồ án, đầu tiên xin cho phép em gửi lời cảm ơn sâu sắc đến cô **Nguyễn Thị Kim Phụng** - giảng viên hướng dẫn môn học **Mạng xã hội (IS353.N11.HTCL)** đã cung cấp cho em những kiến thức bổ ích và sự trợ giúp cần thiết trong suốt khoảng thời gian thực hiện đồ án, cũng như khoa Hệ thống Thông tin, Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh đã tạo những điều kiện tốt nhất giúp cho em có được cơ hội để thực hiện hóa đê tài và hoàn thiện đồ án này.

Cuối cùng, em xin kính chúc Quý Thầy/Cô của khoa Hệ thống Thông tin cũng như cô Nguyễn Thị Kim Phụng sức khỏe dồi dào và thành công trên lĩnh vực của mình để sẵn sàng tiếp tục trên con đường truyền đạt kiến thức, truyền lửa và nhiệt huyết cho thế hệ mai sau. Xin trân trọng cảm ơn Quý Thầy/Cô rất nhiều.

## 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, một số kết quả của phần thực hiện thủ công và một số hình ảnh của đồ thị của phần lập trình.
  - **19521847\_BaoCaoCuoiKy\_IS353.N11.HTCL.docx** (file hiện tại).
  - **19521847\_BaoCaoCuoiKy\_IS353.N11.HTCL.pdf**.
  - **Images**: Thư mục chứa một số hình ảnh đồ thị của phần lập trình.
  - **Results.xlsx**: File *Excel* có một số kết quả của phần thực hiện thủ công.
2. **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*).
3. **Gephi**:
  - **ExportedImages**: Thư mục chứa các hình ảnh đồ thị export từ *Gephi*.
  - **ExportedResults**: Thư mục chứa các file kết quả export từ *Gephi*.
  - **Plugins**: Thư mục chứa plugin của *Gephi*.
  - **MainGraph.gephi**: File *Gephi* thực hiện trên đồ thị gốc của phần lập trình.

## 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.

## CHƯƠNG 1. MỞ ĐẦU

### 1.1. 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 nổi 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 Betweenness Centrality ở các đỉnh (nodes) và cạnh (edges).
- Tính toán độ đo Closeness Centrality ở các đỉnh (nodes).

### 1.2. Dữ liệu

#### 1.2.1. 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 thông 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 Mỹ
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 1 dòng dữ liệu: Game *Pokémon Platinum Version* thuộc thể loại *Role-Playing* do nhà phát hành *Nintendo* cho ra mắt vào năm 2008.

# Rank	A Name	A Platform	A Year	A Genre	A Publisher
89	Pokémon Platinum Version	DS	2008	Role-Playing	Nintendo

### 1.2.2. 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 máu chót 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 lược bỏ đi hết.

## CHƯƠNG 2. 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 *Gephi* và file source code vì có thể file *Gephi* và 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.

### 2.1. 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.

Just need only 2 columns: "Genre" and "Publisher"

```

df = pd.read_csv('vgsales.csv', usecols = ['Genre', 'Publisher'])
df_new = df.dropna().drop_duplicates()
df_new

```

	Genre	Publisher
0	Sports	Nintendo
1	Platform	Nintendo
2	Racing	Nintendo
4	Role-Playing	Nintendo
5	Puzzle	Nintendo
...	...	...
16565	Platform	Rain Games
16566	Strategy	Trion Worlds
16570	Simulation	UIG Entertainment
16589	Action	dramatic create
16596	Puzzle	7G//AMES

1825 rows × 2 columns

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).

## 2.2. Vẽ đồ thị

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

Get graph's info

```

> <ipython>
    import networkx as nx
    from networkx.algorithms import bipartite
    g_1 = nx.Graph()
    publishers = df_new['Publisher']
    genres = df_new['Genre']
    print("Number of publishers: ", publishers.nunique())
    print("Number of genres: ", genres.nunique())
    print("Number of edges: ", len(df_new))
[11]   ✓  0.3s
...
... Number of publishers:  578
Number of genres:  12
Number of edges:  1825

```

Python

Ta thấy có 2 loại node là *Publisher* với 578 node và *Genre* với 12 node. Các cạnh trong đồ thị 2 phia 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 phia:

```

<ipython>
    for index, row in df_new.iterrows():
        g_1.add_edge(row['Genre'], row['Publisher'], weight = 1)
        g_1.add_nodes_from(genres, bipartite = 0)
        g_1.add_nodes_from(publishers, bipartite = 1)
[12]   ✓  1.6s

```

Python

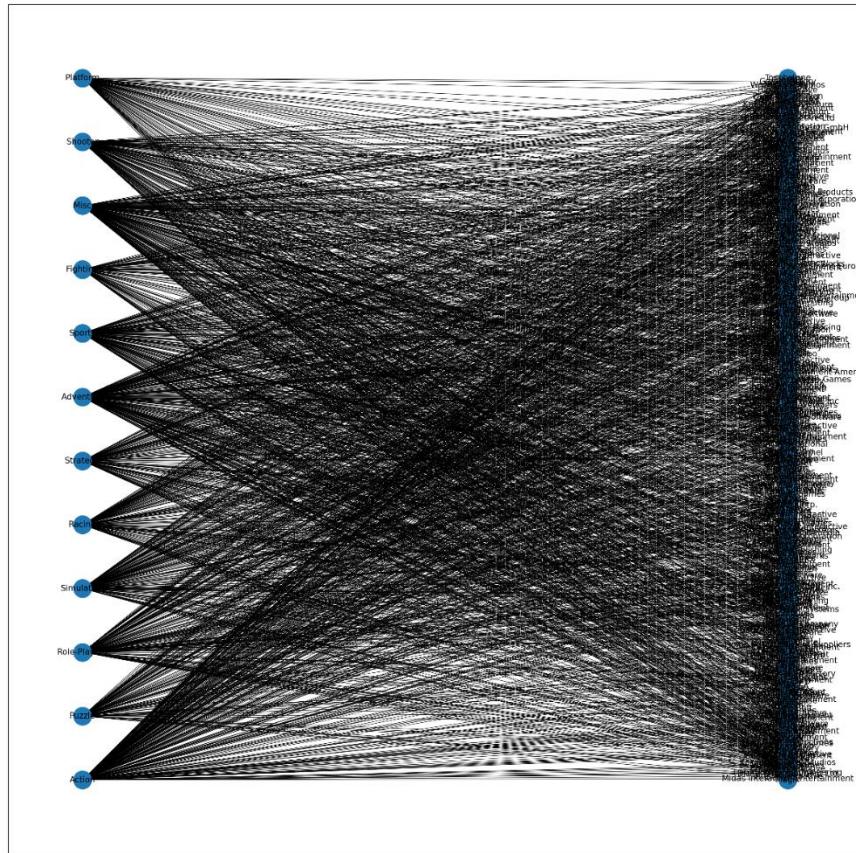
```

    import matplotlib.pyplot as plt
    plt.figure(figsize = (15, 15))
    pos = nx.spring_layout(g_1)
    fig, ax = plt.subplots(1, 1, figsize = (15, 15), dpi = 300)
    nx.draw_networkx(g_1, pos = nx.drawing.layout.bipartite_layout(g_1, genres), font_size = 8, width = 0.4)
[13]   ✓  7.6s

```

Python

.. <Figure size 1500x1500 with 0 Axes>



Hình 1. Đồ thị 2 phia giữa Genres và Publishers

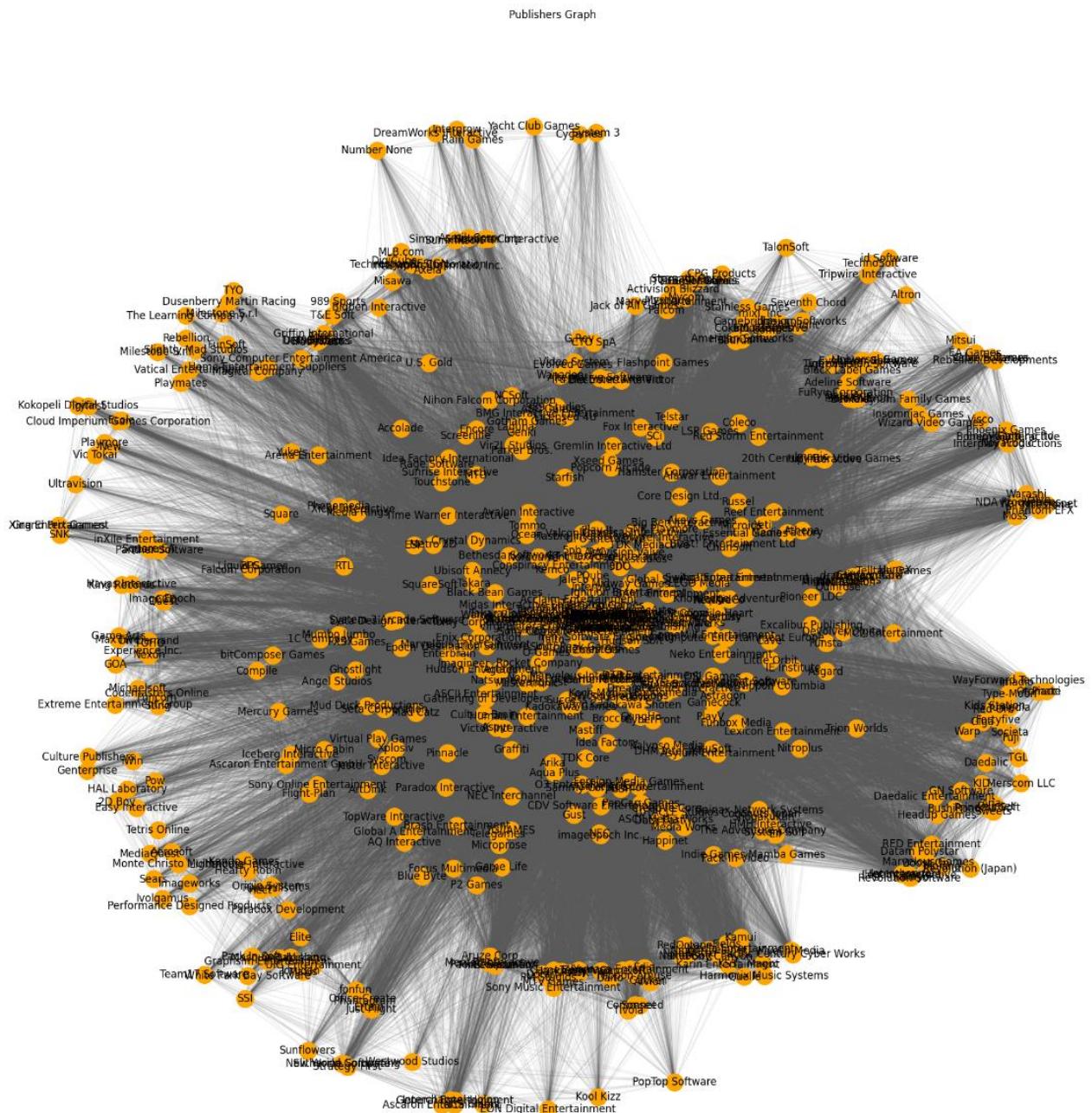
Thực hiện vẽ tiếp đồ thị 1 phia (non-bipartite graph) với các node là các *Publisher* từ dữ liệu của đồ thị 2 phia. Trước tiên ta cũng phải bỏ đi sự lặp đi lặp lại của các *Publisher* trong các hàng dữ liệu của dataframe vì dữ liệu của các node *Publisher* đưa vào đồ thị 1 phia phải duy nhất và phân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

```
# Remove duplicated publishers from rows of the publishers dataframe
# (because we need unique publishers to put into the graph):
final_publishers = publishers.drop_duplicates()
g_2 = bipartite.weighted_projected_graph(g_1, final_publishers)
plt.figure(figsize = (25, 25))

layout = nx.spring_layout(g_2)

nx.draw_networkx_nodes(
    g_2,
    layout,
    nodelist = final_publishers,
    node_size = 400,
    node_color = 'orange'
)
nx.draw_networkx_edges(g_2, layout, edge_color = '#5A5A5A', alpha = 0.1)
node_labels = dict(zip(final_publishers, final_publishers))
nx.draw_networkx_labels(g_2, layout, labels = node_labels)
plt.axis('off')
plt.title('Publishers Graph')

plt.show()
```



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.

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

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

Communities detection with Louvain algorithm

+ Code + Markdown

```
import matplotlib.cm as cm
import matplotlib
from community import community_louvain

plt.figure(figsize = (25, 25))

# Compute the best partition
partition = community_louvain.best_partition(g_2)

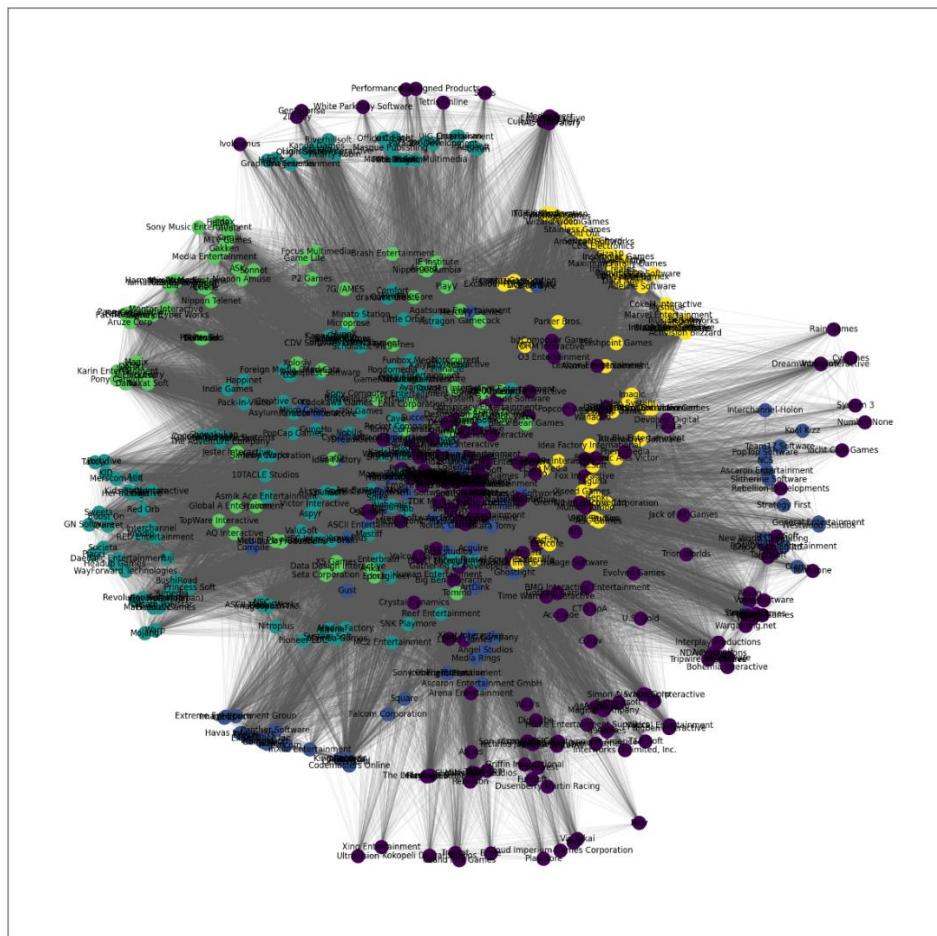
# Draw the graph
pos = nx.spring_layout(g_2)
# Color the nodes according to their partitions
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(g_2, pos, partition.keys(), node_size = 400, cmap = cmap,
node_color=list(partition.values()))
nx.draw_networkx_edges(g_2, pos, alpha = 0.1, edge_color = '#5A5A5A')
nx.draw_networkx_labels(g_2, pos)

plt.show()
```

[25] 9.8s

Python

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:

Calculate number of communities and show the members of each community

```
▷ ~
import numpy as np
values = list(partition.values())

# Print number of communities
print('Number of communities: ', len(np.unique(values)))
[26] ✓ 0.5s
...
... Number of communities: 5
```

Python Python

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):

```
▷ ~
for i in range(len(np.unique(values))):
    print("-----", "Community ", i + 1, "-----")
    print("")
    for name, k in partition.items():
        if k == i:
            print(name)
    print("")
[27] ✓ 0.5s
1 ----- Community  1 -----
2
3 Nintendo
4 Microsoft Game Studios
5 Take-Two Interactive
6 Sony Computer Entertainment
7 Activision
8 Ubisoft
9 Electronic Arts
10 Sega
11 SquareSoft
12 Atari
13 505 Games
14 Capcom
15 GT Interactive

...
196 ----- Community  2 -----
197
198 Bethesda Softworks
199 Square Enix
200 LucasArts
201 Codemasters
202 Level 5
203 ASCII Entertainment
204 Mindscape
205 Square
206 Westwood Studios
207 Sony Online Entertainment
208 Natsume
209 Focus Home Interactive
210 Success
211 Compile
212 Koch Media
213 Square EA
214 Nordic Games
215 Nippon Ichi Software
216 Quest
```

...

```

277 ----- Community 3 -----
278
279 Sony Computer Entertainment Europe
280 Red Orb
281 Maxis
282 GungHo
283 Alchemist
284 City Interactive
285 Russel
286 Taito
287 GSP
288 Sammy Corporation
289 Mojang
290 Scholastic Inc.
291 ChunSoft
292 Avanquest Software
293 Nobilis
294 Spike
295 Funbox Media
296 Rocket Company
297 Rondomedia
298 Kadokawa Shoten

```

...

```

--- -----
429 ----- Community 4 -----
430
431 RedOctane
432 Enix Corporation
433 MTV Games
434 Activision Value
435 Oxygen Interactive
436 Agetec
437 Microprose
438 Tomy Corporation
439 Marvelous Entertainment
440 Pinnacle
441 Quelle
442 Ubisoft Annecy
443 Jester Interactive
444 IE Institute
445 Takara
446 Aruze Corp
447 XS Games
448 Xplosiv
449 Epoch
450 Harmonix Music Systems

```

...

```

518 ----- Community 5 -----
519
520 Palcom
521 989 Studios
522 NCSoft
523 Parker Bros.
524 Imagic
525 Red Storm Entertainment
526 Video System
527 Hello Games
528 ASC Games
529 Black Label Games
530 Genki
531 mixi, Inc
532 Mystique
533 20th Century Fox Video Games
534 Men-A-Vision
535 Touchstone
536 LEGO Media
537 Illusion Softworks
538 Tigervision
539 Universal Gamex
540 Wizard Video Games
541 Laguna
542 Xseed Games
543 CPG Products

```

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:

Analyse genres of each community to get to know its meaning

```

# Remove duplicated genres from rows of the genres dataframe
# (because we need unique genres):
final_genres = genres.drop_duplicates()
comNodes = []
print("Analyse genres of each community")
print("")

for c in range(len(np.unique(values))):
    comNodes.clear()
    genresValues = []
    print("-----", "Community ", c + 1, "-----")
    print("")
    for name, k in partition.items():
        if k == c:
            comNodes.append(name)
    for genre in final_genres:
        hasGenreCount = 0
        for i in comNodes:
            # Search in the dataframe if this publisher in this community
            # has a connection to this genre or not
            if ((df_new['Genre'] == genre) & (df_new['Publisher'] == i)).any():
                hasGenreCount = hasGenreCount + 1
        genresValues.append(hasGenreCount)
    print(genre + ": " + str(hasGenreCount))

# Draw bar chart
print("")
fig = plt.figure(figsize = (15, 5))
# Creating the bar plot
plt.bar(final_genres, genresValues, color ='orange', width = 0.4)
plt.xlabel("Game genres")
plt.ylabel("Number of publishers have published")
plt.title("Popularities of game genres in Community " + str(c + 1))
plt.show()
print("")
```

[28] ✓ 2.9s Python

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*:

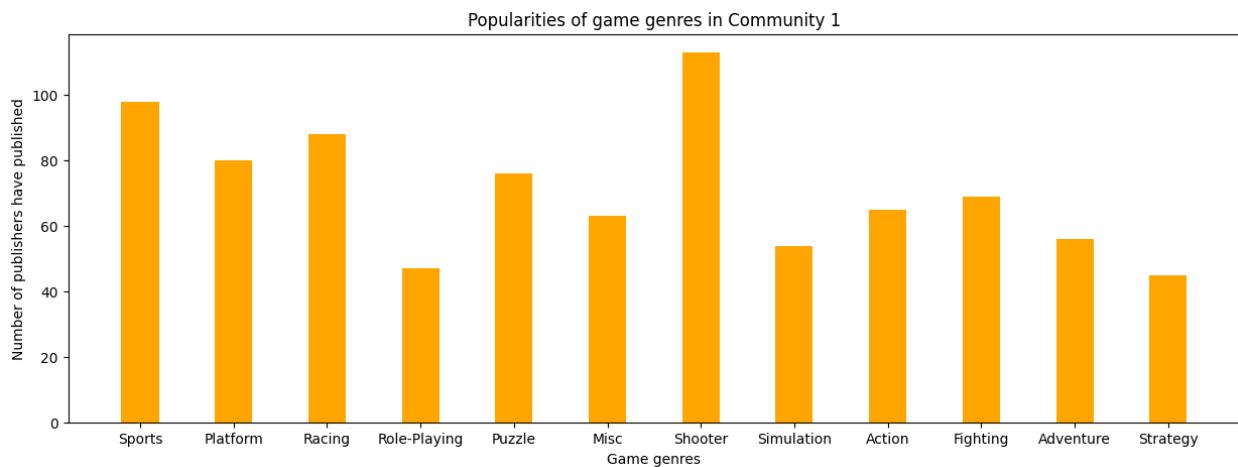
```

Analyse genres of each community

----- Community 1 -----

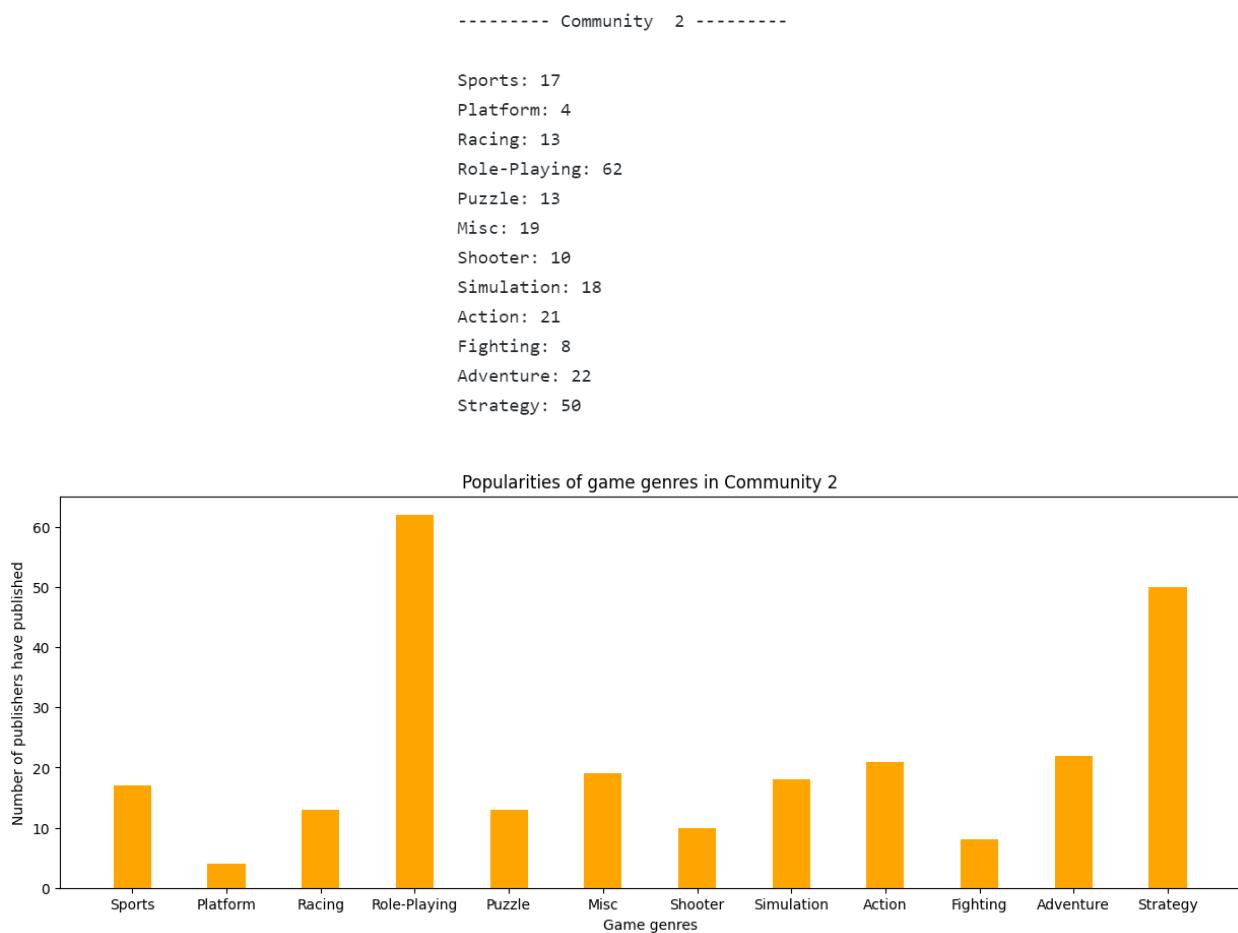
Sports: 98
Platform: 80
Racing: 88
Role-Playing: 47
Puzzle: 76
Misc: 63
Shooter: 113
Simulation: 54
Action: 65
Fighting: 69
Adventure: 56
Strategy: 45
```

Đồ thị biểu thị kết quả trên:



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:

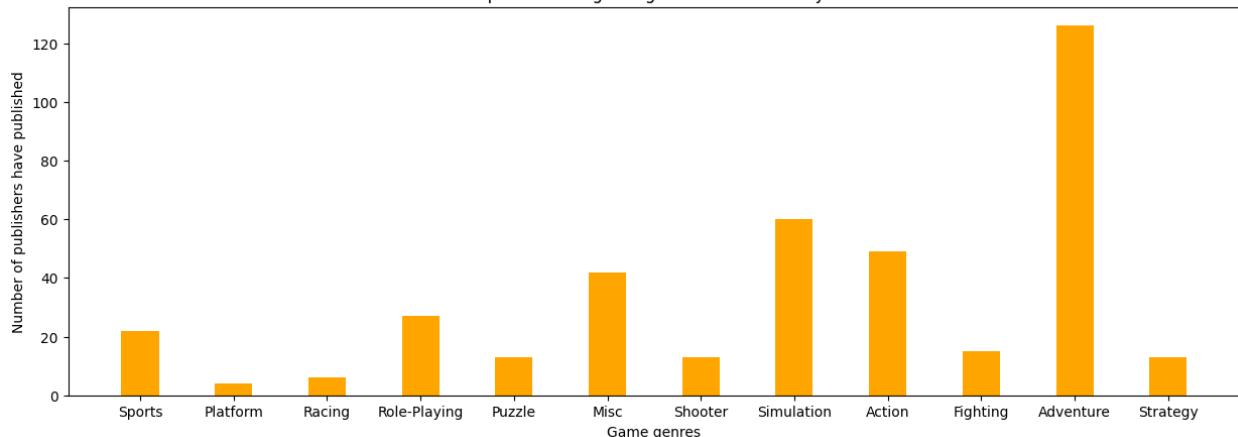


Ta thấy ở trong *Cụm 2* thì trong tất cả các thể loại game đã có đến **62** nhà phát hành đã xuất bản thể loại **Role-Playing**.

----- Community 3 -----

```
Sports: 22
Platform: 4
Racing: 6
Role-Playing: 27
Puzzle: 13
Misc: 42
Shooter: 13
Simulation: 60
Action: 49
Fighting: 15
Adventure: 126
Strategy: 13
```

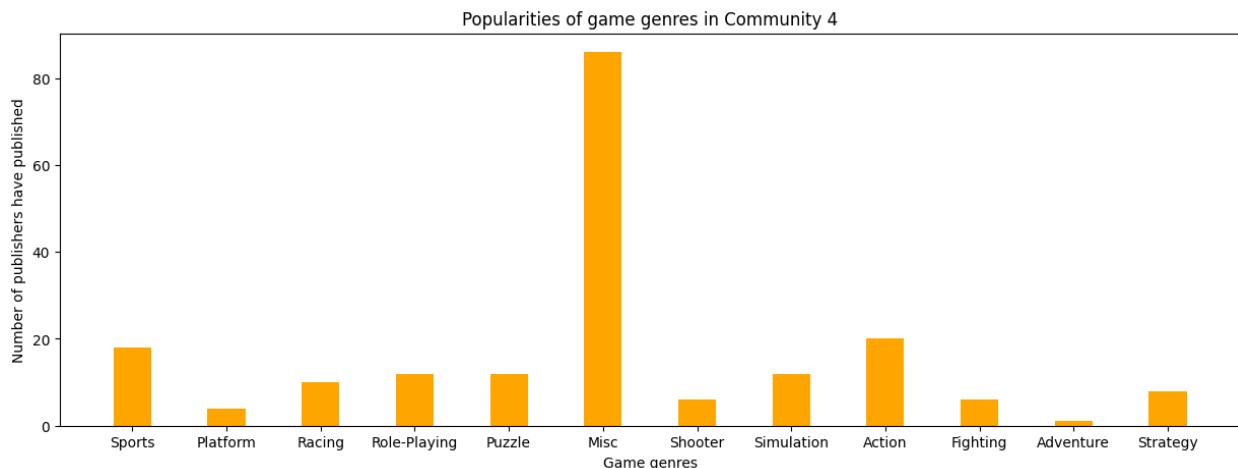
Popularities of game genres in Community 3



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**.

----- Community 4 -----

```
Sports: 18
Platform: 4
Racing: 10
Role-Playing: 12
Puzzle: 12
Misc: 86
Shooter: 6
Simulation: 12
Action: 20
Fighting: 6
Adventure: 1
Strategy: 8
```



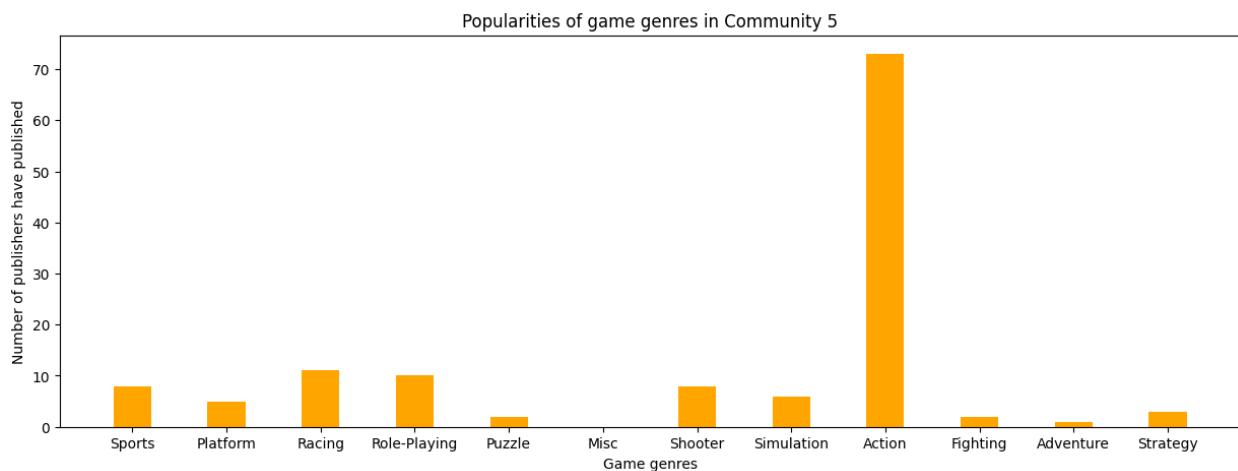
Ta thấy ở trong *Cụm 4* thì trong tất cả các thể loại game đã có đến **86** nhà phát hành đã xuất bản thể loại **Misc**.

----- Community 5 -----

```

Sports: 8
Platform: 5
Racing: 11
Role-Playing: 10
Puzzle: 2
Misc: 0
Shooter: 8
Simulation: 6
Action: 73
Fighting: 2
Adventure: 1
Strategy: 3

```



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 phia ra 1 file dataset **for\_gephi.csv** để sử dụng cho việc trực quan hóa 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 Colab*:

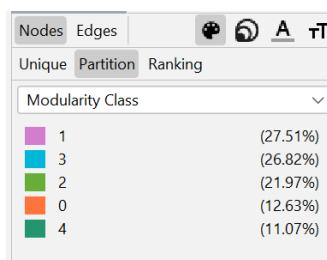
Export the non bipartite graph data to a .csv file (to use in Gephi)

```

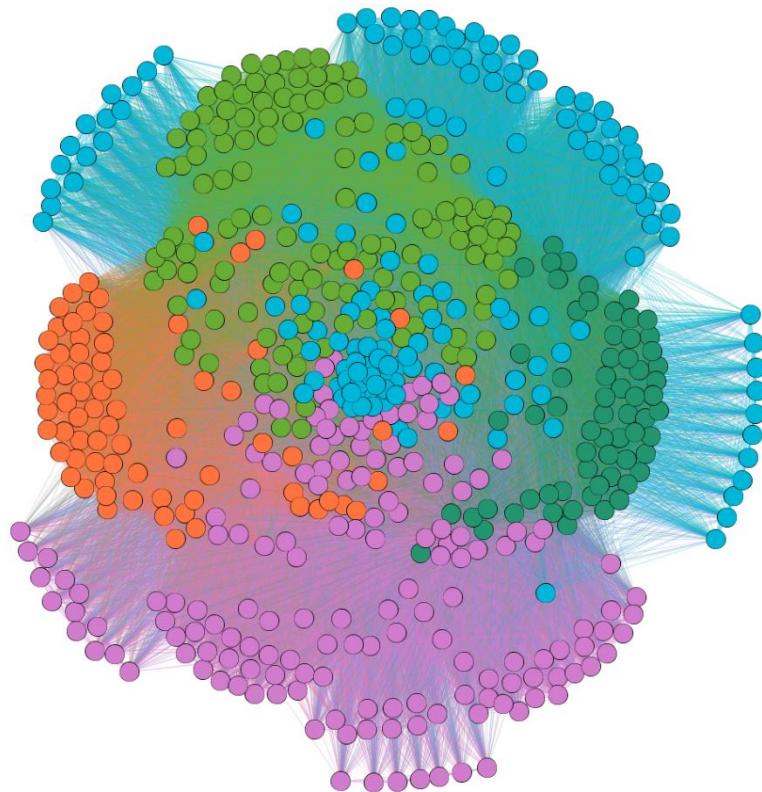
labels = nx.get_edge_attributes(g_2, 'weight')
df2 = pd.DataFrame(columns = ['source', 'target', 'weight'])
for key, value in labels.items():
    df2 = df2.append({'source': key[0], 'target': key[1], 'weight': value}, ignore_index = True)
# Export to .csv file to be used in Gephi
df2.to_csv('for_gephi.csv', index = False, header = True)
# If this file is compiled with Google Colab please add '/' before the file name to get the
# exported csv in the root directory as below:
# df2.to_csv('/for_gephi.csv', index = False, header = True)

```

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 hóa 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**,... Chúng tôi việc chạy thuật toán Louvain bằng Python cho kết quả gần như tương đồng so với kết quả trong *Gephi* cho ra.

Id	Label	Interval	Modularity Class
Nintendo		3	
Midas Interactive Entertainment		3	
Virgin Interactive		3	
SNK		3	
Irem Software Engineering		3	
Kokopelli Digital Studios		3	
King Records		3	
Ghostlight		3	
Vivendi Games		3	
LucasArts		3	
Capcom		3	
Namco Bandai Games		3	
Xing Entertainment		3	

Id	Label	Interval	Modularity Class
Microsoft Game Studios		3	
Marvelous Entertainment		3	
NCS		3	
Focus Home Interactive		3	
Experience Inc.		3	
Micropose		3	
Nippon Ichi Software		3	
Graffiti		3	
Tecmo Koei		3	
Metro 3D		3	
Atari		3	
Cloud Imperium Games Corporation		3	
Take-Two Interactive		3	Modularity Class
bitComposer Games		3	
Blue Byte		3	
Image Epoch		3	
Interchannel-Holon		3	
SNK Playmore		3	
Agetec		3	
Origin Systems		3	
General Entertainment		3	
Ultravision		3	
5pb		3	
Sunsoft		3	
Pack In Soft		3	
Sony Computer Entertainment		3	Modularity Class
Majesco Entertainment		3	
Konami Digital Entertainment		3	
Paradox Development		3	
Office Create		3	
Micro Cabin		3	
Lighthouse Interactive		3	
Ubisoft		3	
Titus		3	
New		3	
New World Computing		3	
inXile Entertainment		3	
Extreme Entertainment Group		3	
Activision		3	Modularity Class
Aqua Plus		3	
ASCII Entertainment		3	
On Demand		3	
Hudson Soft		3	
SouthPeak Games		3	
Mercury Games		3	
Sunrise Interactive		3	
GOA		3	
Panther Software		3	
Sega		3	
Flight-Plan		3	

## 2.4. Thuật toán xếp hạng đỉnh

### 2.4.1. 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 lục):

Node ranking algorithms

#### 1. PageRank

```

pr = nx.pagerank(g_2)
for k, v in sorted(pr.items(), key=lambda item: item[1], reverse=True):
    print(k, ": ", v)

```

[97] ✓ 0.2s

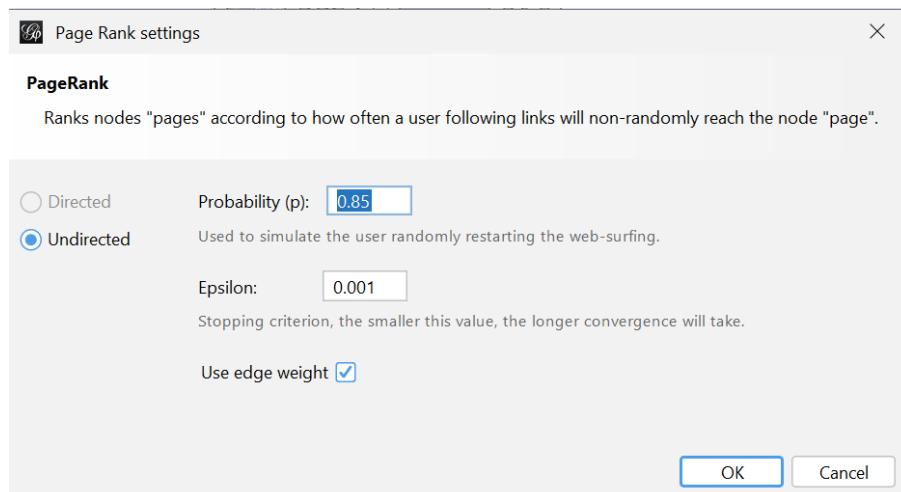
Python

```

1   Nintendo : 0.005465027145877209
2   Microsoft Game Studios : 0.005465027145877209
3   Take-Two Interactive : 0.005465027145877209
4   Sony Computer Entertainment : 0.005465027145877209
5   Activision : 0.005465027145877209
6   Ubisoft : 0.005465027145877209
7   Electronic Arts : 0.005465027145877209
8   Sega : 0.005465027145877209
9   Atari : 0.005465027145877209
10  505 Games : 0.005465027145877209
11  Capcom : 0.005465027145877209
12  Konami Digital Entertainment : 0.005465027145877209
13  Virgin Interactive : 0.005465027145877209
14  Eidos Interactive : 0.005465027145877209
15  Vivendi Games : 0.005465027145877209
16  Namco Bandai Games : 0.005465027145877209
17  THQ : 0.005465027145877209
18  Majesco Entertainment : 0.005465027145877209
19  Deep Silver : 0.005465027145877209
20  Unknown : 0.005465027145877209
21  Banpresto : 0.005465027145877209
22  D3Publisher : 0.005465027145877209
23  Crave Entertainment : 0.005465027145877209
24  SouthPeak Games : 0.005193271449076433
25  Tecmo Koei : 0.005131309986833589
26  Infogrames : 0.005131309986833589

```

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.

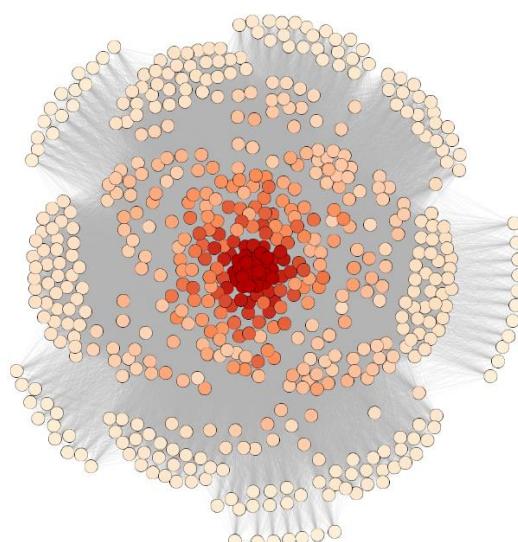


Kết quả chạy thuật toán PageRank trên phần mềm *Gephi*:

Id	Label	Interval	Modularity Class	PageRank
Virgin Interactive		3	0.005465	
Vivendi Games		3	0.005465	
Capcom		3	0.005465	
THQ		3	0.005465	
Sega		3	0.005465	
Deep Silver		3	0.005465	
Majesco Entertainment		3	0.005465	
Konami Digital Entertainment		3	0.005465	
Ubisoft		3	0.005465	
Eidos Interactive		3	0.005465	
D3Publisher		3	0.005465	
Banpresto		3	0.005465	
Banpresto		3	0.005465	
Electronic Arts		3	0.005465	
Electronic Arts		3	0.005465	
Unknown		3	0.005465	
Sony Computer Entertainment		3	0.005465	
Nintendo		3	0.005465	
Namco Bandai Games		3	0.005465	
Activision		3	0.005465	
Crave Entertainment		3	0.005465	
Take-Two Interactive		3	0.005465	
Take-Two Interactive		3	0.005465	
505 Games		3	0.005465	
Microsoft Game Studios		3	0.005465	
Atari		3	0.005465	
SouthPeak Games		3	0.005193	
Infogrames		3	0.005131	
Tecmo Koei		3	0.005131	
Atlus		3	0.005131	
Hudson Soft		3	0.005106	
Rising Star Games		3	0.005106	

**Nhận xét:** Do tên của các node, hay các nhà phát hành được sắp xếp theo các cách khác nhau tùy 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. Một 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):

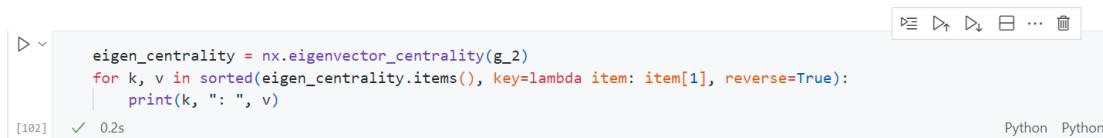


Hình 5. Đồ thị PageRank trong *Gephi*

## 2.4.2. Thuật toán Eigenvector Centrality

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

2. Eigenvector Centrality



```
eigen_centrality = nx.eigenvector_centrality(g_2)
for k, v in sorted(eigen_centrality.items(), key=lambda item: item[1], reverse=True):
    print(k, ":", v)
```

[102] ✓ 0.2s      Python      Python

Kết quả sau khi chạy thuật toán trên Python:

```
1 Nintendo : 0.06657201149659496
2 Microsoft Game Studios : 0.06657201149659496
3 Take-Two Interactive : 0.06657201149659496
4 Sony Computer Entertainment : 0.06657201149659496
5 Activision : 0.06657201149659496
6 Ubisoft : 0.06657201149659496
7 Electronic Arts : 0.06657201149659496
8 Sega : 0.06657201149659496
9 Atari : 0.06657201149659496
10 505 Games : 0.06657201149659496
11 Capcom : 0.06657201149659496
12 Konami Digital Entertainment : 0.06657201149659496
13 Virgin Interactive : 0.06657201149659496
14 Eidos Interactive : 0.06657201149659496
15 Vivendi Games : 0.06657201149659496
16 Namco Bandai Games : 0.06657201149659496
17 THQ : 0.06657201149659496
18 Majesco Entertainment : 0.06657201149659496
19 Deep Silver : 0.06657201149659496
20 Unknown : 0.06657201149659496
21 Banpresto : 0.06657201149659496
22 D3Publisher : 0.06657201149659496
23 Crave Entertainment : 0.06657201149659496
24 SouthPeak Games : 0.06625278258410582
25 Hudson Soft : 0.06598880990927157
26 Rising Star Games : 0.06598880990927157
27 Tecmo Koei : 0.0658892071678246
28 Infogrames : 0.0658892071678246
```

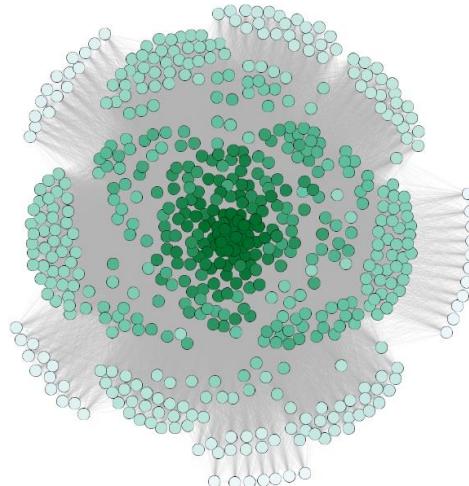
Kết quả sau khi chạy thuật toán trên *Gephi*:

Id	Label	Interval	Modularity Class	PageRank	Eigenvector Centrality
Deep Silver		3		0.005465	1.0
Majesco Entertainment		3		0.005465	1.0
Virgin Interactive		3		0.005465	1.0
Capcom		3		0.005465	1.0
Activision		3		0.005465	1.0
Sega		3		0.005465	1.0
Take-Two Interactive		3		0.005465	1.0
Sony Computer Entertainment		3		0.005465	1.0
Konami Digital Entertainment		3		0.005465	1.0
Ubisoft		3		0.005465	1.0
505 Games		3		0.005465	1.0
Atari		3		0.005465	1.0
D3Publisher		3		0.005465	1.0
Banpresto		3		0.005465	1.0
Electronic Arts		3		0.005465	1.0
Unknown		3		0.005465	1.0
Crave Entertainment		3		0.005465	1.0
Eidos Interactive		3		0.005465	1.0
Microsoft Game Studios		3		0.005465	1.0
Nintendo		3		0.005465	1.0
Vivendi Games		3		0.005465	1.0
Namco Bandai Games		3		0.005465	1.0

Id	Label	Interval	Modularity Class	PageRank	Eigenvector Centrality
Unknown		3	0.005465	1.0	
Crave Entertainment		3	0.005465	1.0	
Eidos Interactive		3	0.005465	1.0	
Microsoft Game Studios		3	0.005465	1.0	
Nintendo		3	0.005465	1.0	
Vivendi Games		3	0.005465	1.0	
Namco Bandai Games		3	0.005465	1.0	
THQ		3	0.005465	1.0	
SouthPeak Games		3	0.005193	0.99509	
Hudson Soft		3	0.005106	0.991109	
Rising Star Games		3	0.005106	0.991109	
Atlus		3	0.005131	0.98954	
Infogrames		3	0.005131	0.98954	
Tecmo Koei		3	0.005131	0.98954	
Zushi Games		1	0.004905	0.987269	
Codemasters		1	0.004905	0.987269	
Empire Interactive		1	0.004848	0.982874	
Psygnosis		1	0.004831	0.979962	
Disney Interactive Studios		1	0.004831	0.979962	
Zoo Digital Publishing		1	0.005013	0.977591	
Interplay		1	0.004786	0.977324	
DTP Entertainment		3	0.004546	0.977243	

**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* đã sử dụng kỹ thuật **normalise vector**, cụ thể là trên đoạn **[0, 1]** nên vector kết quả đã trai qua quá trình normalisation còn thuật toán trong method **eigenvector\_Centrality()** của **NetworkX** là kết quả nguyên gốc của vector 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 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 *Gephi* đều đã được normalise gấp một lượng là **15.02132769** so với trên Python. Như vậy kết quả là không sai, chỉ là do trai 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*

### 2.4.3. 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 giá 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**.

## 2.5. Tính toán các độ đo

### 2.5.1. Betweenness Centrality

- Theo node (node-based):**

Tính Betweenness Centrality theo node trên Python:



```
Betweenness and closeness centrality (descending order)

1. Betweenness centrality
+ Code + Markdown

a. For nodes


bet_centrality_nodes = nx.betweenness_centrality(g_2)
for k, v in sorted(bet_centrality_nodes.items(), key=lambda item: item[1], reverse=True):
    print(k, ":", v)


```

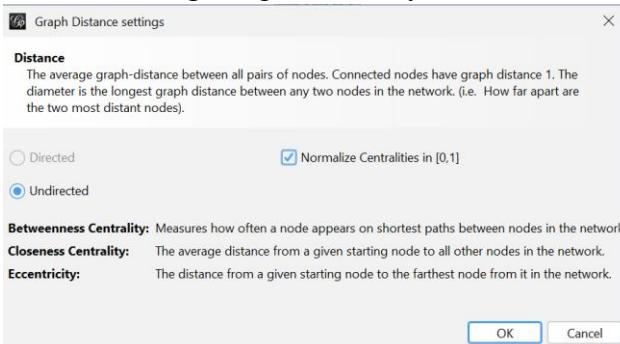
Kết quả nhận được (trích lượ):

```

1  Nintendo :  0.005267974053741265
2  Microsoft Game Studios :  0.005267974053741265
3  Take-Two Interactive :  0.005267974053741265
4  Sony Computer Entertainment :  0.005267974053741265
5  Activision :  0.005267974053741265
6  Ubisoft :  0.005267974053741265
7  Electronic Arts :  0.005267974053741265
8  Sega :  0.005267974053741265
9  Atari :  0.005267974053741265
10 505 Games :  0.005267974053741265
11 Capcom :  0.005267974053741265
12 Konami Digital Entertainment :  0.005267974053741265
13 Virgin Interactive :  0.005267974053741265
14 Eidos Interactive :  0.005267974053741265
15 Vivendi Games :  0.005267974053741265
16 Namco Bandai Games :  0.005267974053741265
17 THQ :  0.005267974053741265
18 Majesco Entertainment :  0.005267974053741265
19 Deep Silver :  0.005267974053741265
20 Unknown :  0.005267974053741265
21 Banpresto :  0.005267974053741265
22 D3Publisher :  0.005267974053741265
23 Crave Entertainment :  0.005267974053741265
24 SouthPeak Games :  0.004960055897086254
25 Hudson Soft :  0.00492259773659793
26 Rising Star Games :  0.00492259773659793
27 Tecmo Koei :  0.004769017080480515
28 Infogrames :  0.004769017080480515
29 Atlus :  0.004769017080480515
30 Zoo Digital Publishing :  0.004660020789258745
31 Acclaim Entertainment :  0.004570301812911178

```

Tiến hành tính Betweenness Centrality theo node trên *Gephi* bằng công cụ **Network Diameter** và công cụ này tính toán các độ đo dựa trên khoảng cách giữa các node trong đồ thị, hay ở đây chính là trọng số (số các thể loại chung nhau giữa 2 nhà phát hành game). Lần này trong cài đặt có xuất hiện việc có muốn normalise kết quả hay không thì ta chọn lựa chọn này, đó chính là **Normalize Centralities in [0, 1]** để kết quả trên *Gephi* có thể được tương đồng với trên Python:

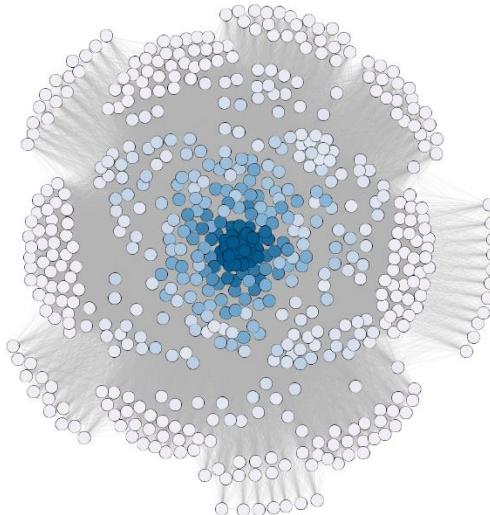


Vào **Data Table** và chọn tab **Node** để hiển thị xếp hạng theo node-based của Betweenness Centrality, ta thấy kết quả tương đồng như trên Python với các node xếp hạng cao nhất có kết quả xấp xỉ **0.005268**:

Id	Modularity Class	Betweenness Centrality
Nintendo	3	0.005268
Virgin Interactive	3	0.005268
Vivendi Games	3	0.005268
Capcom	3	0.005268
Namco Bandai ...	3	0.005268
THQ	3	0.005268
Activision	3	0.005268
Sega	3	0.005268
Deep Silver	3	0.005268
Crave Entertain...	3	0.005268
Take-Two Interac...	3	0.005268
Sony Computer ...	3	0.005268
Majesco Entertai...	3	0.005268
Konami Digital ...	3	0.005268
Ubisoft	3	0.005268
505 Games	3	0.005268
Eidos Interactive	3	0.005268
Microsoft Game ...	3	0.005268
Atari	3	0.005268
D3Publisher	3	0.005268
Banpresto	3	0.005268
Electronic Arts	3	0.005268

Id	Modularity Class	Betweenness Centrality
Electronic Arts	3	0.005268
Unknown	3	0.005268
SouthPeak Games	3	0.00496
Hudson Soft	3	0.004923
Rising Star Games	3	0.004923
Infogrames	3	0.004769
Tecmo Koei	3	0.004769
Altus	3	0.004769
Zoo Digital Publ...	1	0.00466
Acclaim Enterta...	1	0.00457
Ignition Enterta...	1	0.00457
Codemasters	1	0.004496
Zushi Games	1	0.004496
Midas Interactiv...	3	0.004453
Empire Interactive	1	0.004404
Psygnosis	1	0.004244
Disney Interactiv...	1	0.004244
Interplay	1	0.00422
DTP Entertainm...	3	0.004145
Focus Home Int...	3	0.004061
Midway Games	1	0.004014
Square Enix	3	0.003958

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



Hình 7. Đồ thị Betweenness Centrality theo node trong *Gephi*

**Ý nghĩa:** Trong đồ thị này, kết quả của Betweenness Centrality với node-based cho biết độ quan trọng hay khả năng được “ghé qua” của 1 node nào đó trong đồ thị từ quan hệ với các node khác, tức là node nào được cross qua nhiều trên đường đi của tất cả các node trong đồ thị thì càng có độ đo này cao. Hay trong bài toán cụ thể này, độ đo này cho biết 1 nhà phát hành nào đó có độ đo của Betweenness Centrality với node-based càng cao thì nhà phát hành đó càng có vai trò quan trọng trong đồ thị, càng đóng vai trò “trung gian” trong nhiều quan hệ gián tiếp nhất trong việc có chung thẻ loại game. Ví dụ nhà phát hành **A** và **B** có chung thẻ loại game I, nhà phát hành **B** và **C** có chung thẻ loại game II thì trong quan hệ giữa 3 nhà phát hành **A**, **B**, **C** này nhà phát hành **B** là nhà phát hành có độ đo của Betweenness Centrality với node-based cao nhất vì là nhà phát hành trung gian cho quan hệ gián tiếp giữa 2 nhà phát hành **A** và **C**. Vẫn như những phần trước, ta thấy các nhà phát hành có độ đo này cao thường nằm ở trung tâm của đồ thị, và điều là những nhà phát hành có vai trò quan trọng cũng như là những nhà phát hành lớn trong thị trường.

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

Tính Betweenness Centrality theo edge trên Python:

b. For edges

```
▷ ▾
bet_centrality_edges = nx.edge_betweenness_centrality(g_2)
for k, v in sorted(bet_centrality_edges.items(), key=lambda item: item[1], reverse=True):
    print(k, ": ", v)
```

Python

[104]

Kết quả nhận được (trích lọt):

```

1   ('Nintendo', 'Cygames') : 4.983299302415959e-05
2   ('Nintendo', 'Yacht Club Games') : 4.983299302415959e-05
3   ('Nintendo', 'DreamWorks Interactive') : 4.983299302415959e-05
4   ('Microsoft Game Studios', 'Cygames') : 4.983299302415959e-05
5   ('Microsoft Game Studios', 'Yacht Club Games') : 4.983299302415959e-05
6   ('Microsoft Game Studios', 'DreamWorks Interactive') : 4.983299302415959e-05
7   ('Take-Two Interactive', 'Cygames') : 4.983299302415959e-05
8   ('Take-Two Interactive', 'Yacht Club Games') : 4.983299302415959e-05
9   ('Take-Two Interactive', 'DreamWorks Interactive') : 4.983299302415959e-05
10  ('Sony Computer Entertainment', 'Cygames') : 4.983299302415959e-05
11  ('Sony Computer Entertainment', 'Yacht Club Games') : 4.983299302415959e-05
12  ('Sony Computer Entertainment', 'DreamWorks Interactive') : 4.983299302415959e-05
13  ('Activision', 'Cygames') : 4.983299302415959e-05
14  ('Activision', 'Yacht Club Games') : 4.983299302415959e-05
15  ('Activision', 'DreamWorks Interactive') : 4.983299302415959e-05
16  ('Ubisoft', 'Cygames') : 4.983299302415959e-05
17  ('Ubisoft', 'Yacht Club Games') : 4.983299302415959e-05
18  ('Ubisoft', 'DreamWorks Interactive') : 4.983299302415959e-05
19  ('Electronic Arts', 'Cygames') : 4.983299302415959e-05
20  ('Electronic Arts', 'Yacht Club Games') : 4.983299302415959e-05
21  ('Electronic Arts', 'DreamWorks Interactive') : 4.983299302415959e-05
22  ('Sega', 'Cygames') : 4.983299302415959e-05
23  ('Sega', 'Yacht Club Games') : 4.983299302415959e-05
24  ('Sega', 'DreamWorks Interactive') : 4.983299302415959e-05
25  ('Atari', 'Cygames') : 4.983299302415959e-05
26  ('Atari', 'Yacht Club Games') : 4.983299302415959e-05
27  ('Atari', 'DreamWorks Interactive') : 4.983299302415959e-05
28  ('505 Games', 'Cygames') : 4.983299302415959e-05
29  ('505 Games', 'Yacht Club Games') : 4.983299302415959e-05
30  ('505 Games', 'DreamWorks Interactive') : 4.983299302415959e-05
31  ('Capcom', 'Cygames') : 4.983299302415959e-05
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301

```

Do trong *Gephi* mặc định không hỗ trợ tính Betweenness Centrality theo cạnh nên ta phải sử dụng plugin bên ngoài để phục vụ tính độ đo này. Em đã tham khảo project plugin tính độ đo này ở <https://github.com/gephi/gephi-plugins/tree/edge-betweenness-plugin>. Do project này chỉ bao gồm các file đã được tạo theo chuẩn của một plugin *Gephi* và các file source code của thuật toán bằng Java chứ chưa có file plugin nên để có được plugin thì cần tải về toàn bộ file và tiến hành build ra file plugin dành cho *Gephi* theo documentation của repo.

Một project plugin cho *Gephi* được tạo trên *Maven* nên cần đảm bảo cài đặt *Apache Maven* (<https://maven.apache.org/download.cgi>) và lưu ý cần phải có *Java*

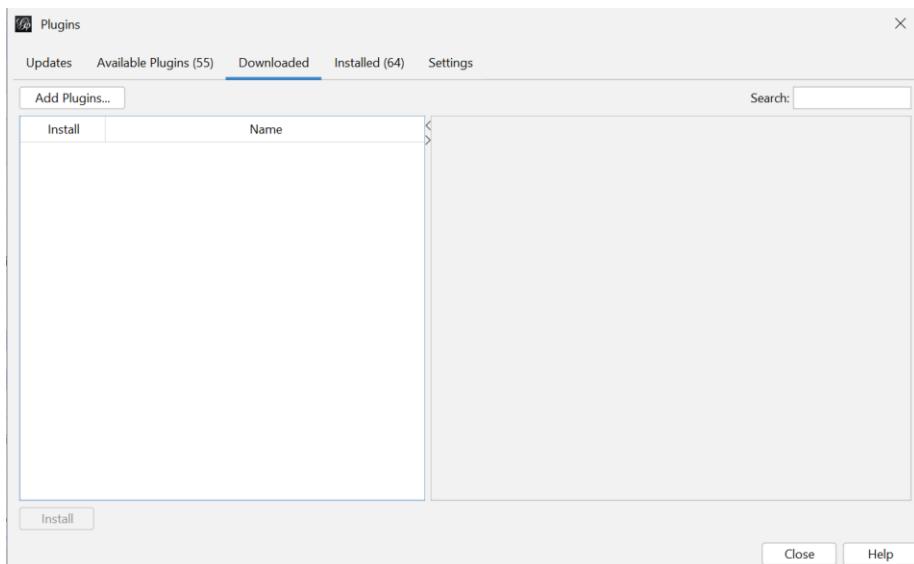
*Development Kit (JDK, <https://www.oracle.com/java/technologies/downloads>)* trước khi tiến hành build ra file plugin cho *Gephi*.

Sau khi quá trình build hoàn tất, ta có được 1 file *.nbm* (là file plugin được tạo ra dựa trên IDE Netbeans) là **edge-betweenness-metric-1.1.0.nbm**. File này cũng được để ở trong đường dẫn “**Gephi/Plugins/EdgeBetweennessMetric/target**” của đồ án:

classes	20/12/2022 17:36	File folder
generated-sources	20/12/2022 17:36	File folder
maven-archiver	20/12/2022 17:36	File folder
maven-status	20/12/2022 17:36	File folder
nbm	20/12/2022 17:36	File folder
edge-betweenness-metric-1.1.0	20/12/2022 13:10	Executable Jar File
<b>edge-betweenness-metric-1.1.0.nbm</b>	20/12/2022 13:10	NBM File

Tiến hành import file **edge-betweenness-metric-1.1.0.nbm** trên vào *Gephi* để cài đặt:

- Trong *Gephi* chọn **Tools → Plugins** để mở cửa sổ plugin.
- Chọn tab **Download → Add Plugins...** và chọn file **edge-betweenness-metric-1.1.0.nbm** để add vào.

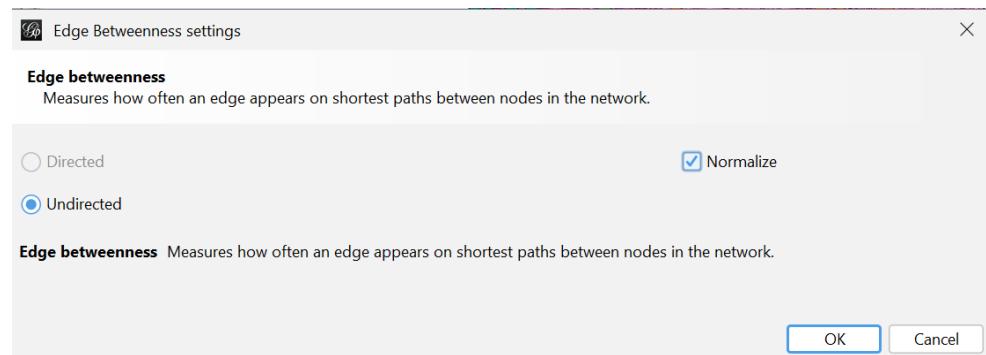


- Khởi động lại *Gephi* để *Gephi* nhận plugin **Edge Betweenness Metric** vừa thêm vào.

Sau khi đã cài đặt plugin, ta có thể thấy lựa chọn tính độ đo này ở **Edge Overview** của **Statistics** với tên là **Edge Betweenness**:

<b>Edge Overview</b>	
Avg. Path Length	1.506 Run
Edge Betweenness	17925965 Run

Tiến hành chạy như các thuật toán thông thường khác, lưu ý tick vào **Normalize** để kết quả nằm trong đoạn **[0, 1]** giúp dễ quản lý hơn:



Sau khi chạy xong thuật toán, cũng tương tự như trong Betweenness centrality của node-based, ta vào **Data Table** và vào tab **Edge** để xem kết quả.

Kết quả nhận được như sau:

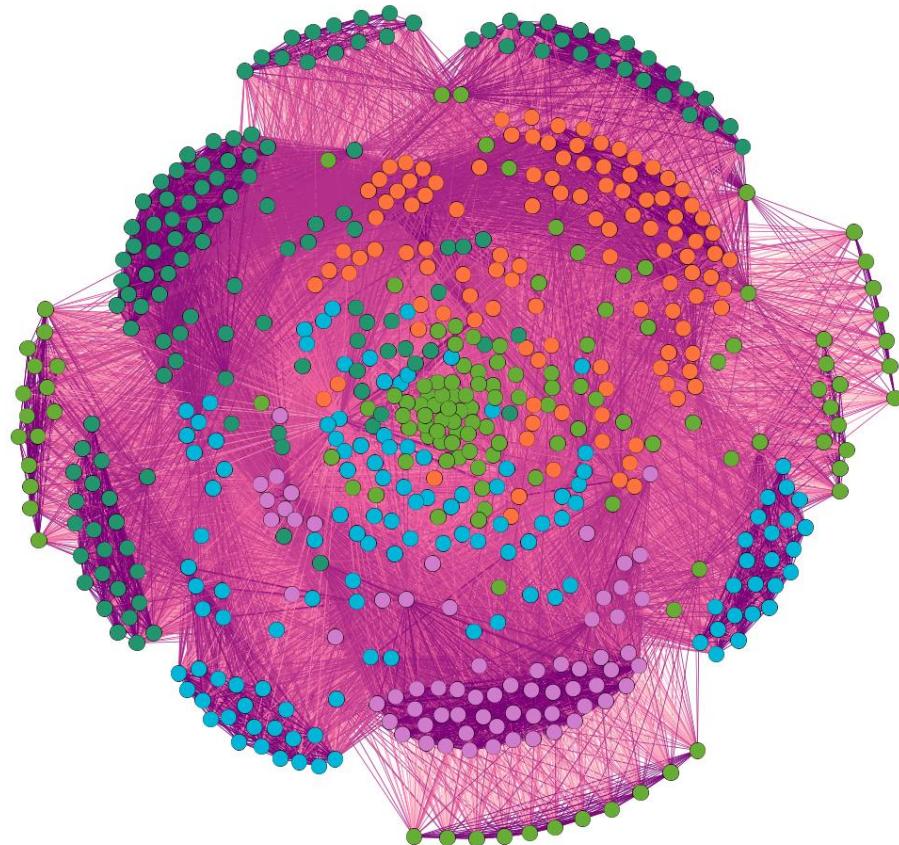
Data Table X							Filter:		
Nodes	Edges	Configuration	Add node	Add edge	Search/Replace	Import Spreadsheet	Export table	More actions	Edge Betweenness
Nintendo	Number None	Undirected	39		1.0	1.0			
Nintendo	Cygames	Undirected	120		1.0	1.0			
Nintendo	Rain Games	Undirected	170		1.0	1.0			
Nintendo	DreamWorks Interactive	Undirected	294		1.0	1.0			
Nintendo	Intergrow	Undirected	462		1.0	1.0			
Nintendo	Yacht Club Games	Undirected	480		1.0	1.0			
Nintendo	System 3	Undirected	485		1.0	1.0			
Microsoft Game Studios	Number None	Undirected	616		1.0	1.0			
Microsoft Game Studios	Cygames	Undirected	697		1.0	1.0			
Microsoft Game Studios	Rain Games	Undirected	747		1.0	1.0			
Microsoft Game Studios	DreamWorks Interactive	Undirected	867		1.0	1.0			
Microsoft Game Studios	Intergrow	Undirected	1038		1.0	1.0			
Microsoft Game Studios	Yacht Club Games	Undirected	1056		1.0	1.0			
Microsoft Game Studios	System 3	Undirected	1061		1.0	1.0			
Take-Two Interactive	Number None	Undirected	1192		1.0	1.0			
Take-Two Interactive	Cygames	Undirected	1273		1.0	1.0			
Take-Two Interactive	Rain Games	Undirected	1323		1.0	1.0			
Take-Two Interactive	DreamWorks Interactive	Undirected	1443		1.0	1.0			
Take-Two Interactive	Intergrow	Undirected	1613		1.0	1.0			
Take-Two Interactive	Yacht Club Games	Undirected	1631		1.0	1.0			
Take-Two Interactive	System 3	Undirected	1636		1.0	1.0			
Sony Computer Entertain...	Number None	Undirected	1767		1.0	1.0			

Data Table X							Filter:		
Nodes	Edges	Configuration	Add node	Add edge	Search/Replace	Import Spreadsheet	Export table	More actions	Edge Betweenness
Crave Entertainment	Rain Games	Undirected	27162		1.0	1.0			
Crave Entertainment	DreamWorks Interactive	Undirected	27274		1.0	1.0			
Crave Entertainment	Intergrow	Undirected	27420		1.0	1.0			
Crave Entertainment	Yacht Club Games	Undirected	27435		1.0	1.0			
Crave Entertainment	System 3	Undirected	27438		1.0	1.0			
Nintendo	SNK	Undirected	4		1.0	0.993776			
Nintendo	Kokopeli Digital Studios	Undirected	14		1.0	0.993776			
Nintendo	Xing Entertainment	Undirected	35		1.0	0.993776			
Nintendo	Playmore	Undirected	96		1.0	0.993776			
Nintendo	Vic Tokai	Undirected	119		1.0	0.993776			
Nintendo	Tryfirst	Undirected	273		1.0	0.993776			
Nintendo	Ultravision	Undirected	343		1.0	0.993776			
Nintendo	New	Undirected	390		1.0	0.993776			
Nintendo	Grand Prix Games	Undirected	424		1.0	0.993776			
Nintendo	Cloud Imperium Games C...	Undirected	476		1.0	0.993776			
Nintendo	Ecole	Undirected	478		1.0	0.993776			
Microsoft Game Studios	SNK	Undirected	581		1.0	0.993776			
Microsoft Game Studios	Kokopeli Digital Studios	Undirected	591		1.0	0.993776			
Microsoft Game Studios	Xing Entertainment	Undirected	612		1.0	0.993776			
Microsoft Game Studios	Playmore	Undirected	673		1.0	0.993776			
Microsoft Game Studios	Vic Tokai	Undirected	696		1.0	0.993776			
Microsoft Game Studios	Tryfirst	Undirected	850		1.0	0.993776			

Có thể thấy kết quả xếp hạng trên cả Python và Gephi đều tương đồng nhau, trong đó kết quả của Python không trải qua quá trình normalisation (như đã giải thích bên trên). Ví dụ cạnh giữa 2 nhà phát hành **Nintendo** và **SNK** xếp 1 bậc dưới so với các cạnh xếp bậc cao nhất.

Tiếp theo ta có đồ thị biểu diễn cho độ đo này, với các cạnh càng có hạng cao hay độ đo Betweenness Centrality theo edge-based càng cao thì có màu (tím) càng đậm. Và để

cho dễ đánh giá ta sẽ kết hợp với màu sắc của các cụm sau khi được phân chia bằng thuật giải Louvain thì thấy là các node có nằm cùng một cụm (1 màu) sẽ thường có rất nhiều cạnh có độ đo cao (màu đậm):



*Hình 8. Đồ thị Betweeness Centrality theo cạnh trong Gephi*

**Ý nghĩa:** Tương tự như Betweeness Centrality theo node thì Betweeness Centrality theo cạnh cho biết cạnh giữa 2 node nào có độ đo càng cao thì nó càng có vai trò quan trọng trong đồ thị, là cạnh mà được cross qua nhiều lần trong các đường đi của đồ thị. Ở đây cho biết 2 ý nghĩa:

- Ở các cụm, các node ở gần và san sát nhau nên số lượng các cạnh có độ Betweeness Centrality theo cạnh cao sê nhiều, đó là lý do ta thấy ở các cụm riêng lẻ luôn có rất nhiều các cạnh có màu đậm.
- 1 cạnh cho biết quan hệ có chung thể loại game giữa 2 nhà phát hành, do vậy khi nó có độ đo cao chứng tỏ quan hệ giữa 2 nhà phát hành này là quan trọng, đóng vai trò gián tiếp cho nhiều quan hệ giữa các nhà phát hành khác trong đồ thị. Ví dụ trong 4 nhà phát hành **A**, **B**, **C**, **D** có 3 quan hệ chung thể loại là **AB**, **BC** và **CD**, do đó trong đồ thị nhà phát hành **A** muốn có quan hệ gián tiếp với **D** thì lúc nào cũng phải thông qua quan hệ **BC** và **CD**, ngược lại **D** muốn có quan hệ gián tiếp với **A** thì phải

thông qua quan hệ **BC** và **AB**. Do quan hệ **BC** nắm vai trò chính trong nhiều quan hệ trung gian (ở đây là 2) nên cạnh **BC** sẽ có độ đo Betweenness Centrality cao nhất.

### 2.5.2. Closeness Centrality

Chạy thuật toán tính độ đo Closeness Centrality trên Python:

2. Closeness centrality

```
clo_centrality = nx.closeness_centrality(g_2)
for k, v in sorted(clo_centrality.items(), key=lambda item: item[1], reverse=True):
    print(k, ": ", v)
```

[105] Python

Kết quả nhận được (trích lues):

```
1  Nintendo : 1.0
2  Microsoft Game Studios : 1.0
3  Take-Two Interactive : 1.0
4  Sony Computer Entertainment : 1.0
5  Activision : 1.0
6  Ubisoft : 1.0
7  Electronic Arts : 1.0
8  Sega : 1.0
9  Atari : 1.0
10 505 Games : 1.0
11 Capcom : 1.0
12 Konami Digital Entertainment : 1.0
13 Virgin Interactive : 1.0
14 Eidos Interactive : 1.0
15 Vivendi Games : 1.0
16 Namco Bandai Games : 1.0
17 THQ : 1.0
18 Majesco Entertainment : 1.0
19 Deep Silver : 1.0
20 Unknown : 1.0
21 Banpresto : 1.0
22 D3Publisher : 1.0
23 Crave Entertainment : 1.0
24 SouthPeak Games : 0.988013698630137
25 Hudson Soft : 0.9829642248722317
26 Rising Star Games : 0.9829642248722317
27 Tecmo Koei : 0.9779661016949153
28 Infogrames : 0.9779661016949153
29 Atlus : 0.9779661016949153
30 Codemasters : 0.9697478991596639
31 Zushi Games : 0.9697478991596639
```

Chạy thuật toán trên Gephi và ta có kết quả nhận được:

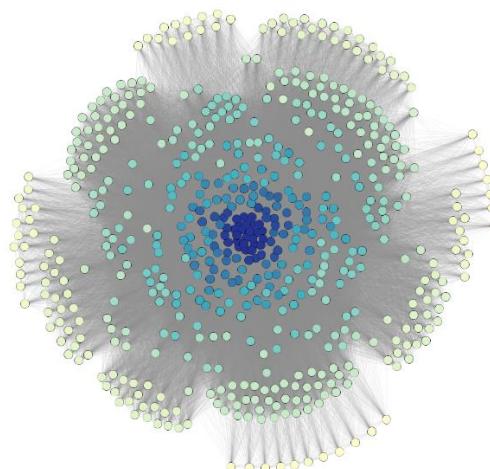
...

Nodes	Edges	Configuration	Add node	Add edge	Search/Replace	Import Spreadsheet	Export table	More actions
Nintendo		3	0.005465	1.0	1.0	1.0		
Virgin Interactive		3	0.005465	1.0	1.0	1.0		
Vivendi Games		3	0.005465	1.0	1.0	1.0		
Capcom		3	0.005465	1.0	1.0	1.0		
Namco Bandai ...		3	0.005465	1.0	1.0	1.0		
THQ		3	0.005465	1.0	1.0	1.0		
Activision		3	0.005465	1.0	1.0	1.0		
Sega		3	0.005465	1.0	1.0	1.0		
Deep Silver		3	0.005465	1.0	1.0	1.0		
Crave Entertain...		3	0.005465	1.0	1.0	1.0		
Take-Two Interac...		3	0.005465	1.0	1.0	1.0		
Sony Computer ...		3	0.005465	1.0	1.0	1.0		
Majesco Entertai...		3	0.005465	1.0	1.0	1.0		
Konami Digital ...		3	0.005465	1.0	1.0	1.0		
Ubisoft		3	0.005465	1.0	1.0	1.0		
505 Games		3	0.005465	1.0	1.0	1.0		
Eidos Interactive		3	0.005465	1.0	1.0	1.0		
Microsoft Game ...		3	0.005465	1.0	1.0	1.0		
Atari		3	0.005465	1.0	1.0	1.0		

Nodes	Edges	Configuration	Add node	Add edge	Search/Replace	Import Spreadsheet	Export table	More actions
Electronic Arts		3	0.005465	1.0	1.0	1.0		
Unknown		3	0.005465	1.0	1.0	1.0		
SouthPeak Games		3	0.005193	0.99509	2.0	0.988014		
Hudson Soft		3	0.005106	0.991109	2.0	0.982964		
Rising Star Games		3	0.005106	0.991109	2.0	0.982964		
Infogrames		3	0.005131	0.98954	2.0	0.977966		
Tecmo Koei		3	0.005131	0.98954	2.0	0.977966		
Atlas		3	0.005131	0.98954	2.0	0.977966		
Codemasters		1	0.004905	0.987269	2.0	0.969748		
Zushi Games		1	0.004905	0.987269	2.0	0.969748		
Zoo Digital Publ...		3	0.005013	0.977591	2.0	0.964883		
Empire Interactive		3	0.004848	0.982874	2.0	0.963272		
Acclaim Entertai...		3	0.005033	0.976647	2.0	0.961667		
Ignition Entertai...		3	0.005033	0.976647	2.0	0.961667		
Psynosis		3	0.004831	0.979962	2.0	0.956882		
Disney Interactiv...		3	0.004831	0.979962	2.0	0.956882		

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



Hình 9. Đồ thị Closeness Centrality trong *Gephi*

Ta nhận thấy kết quả trên cả Python và *Gephi* đều tương đồng nhau, và đều đã được normalise.

**Ý nghĩa:** Cũng giống như kết quả của PageRank, độ đo Closeness Centrality trong đồ thị cho biết nhà phát hành nào có độ đo này càng cao chứng tỏ càng dễ có khả năng tiếp cận, hay gần nhiều nhà phát hành xung quanh, hay nói cách khác là có nhiều cơ hội chung với loại game với nhiều nhà phát hành xung quanh nó. Đó là lý do mà các nhà phát hành có độ đo này cao đều tập trung ở giữa đồ thị vì ở các vị trí này mới có thể đến gần và đến được nhiều node.

Như vậy phần lập trình đến đây là kết thúc. Các file *Excel* được xuất từ kết quả trong *Gephi* được lưu trong thư mục **Gephi** của đồ án là:

File **MainGraph\_Gephi\_Nodes.xlsx** có các kết quả của thuật toán, độ đo trên node như phân cụm theo Louvain với ID cụm là cột **modularity\_class** và kết quả **PageRank**, **Eigenvector Centrality**, **Betweenness Centrality** theo node như sau:

	A	B	C	D	E	F	G	H
1	Id	Label	timeset	modularity_class	pageranks	eigenvcentrality	closenesscentrality	betweenesscentrality
2	Nintendo			3	0.005465	1	1	0.005268
3	Virgin Interactive			3	0.005465	1	1	0.005268
4	Vivendi Games			3	0.005465	1	1	0.005268
5	Capcom			3	0.005465	1	1	0.005268
6	Namco Bandai Games			3	0.005465	1	1	0.005268
7	THQ			3	0.005465	1	1	0.005268
8	Activision			3	0.005465	1	1	0.005268
9	Sega			3	0.005465	1	1	0.005268
10	Deep Silver			3	0.005465	1	1	0.005268
11	Crave Entertainment			3	0.005465	1	1	0.005268
12	Take-Two Interactive			3	0.005465	1	1	0.005268
13	Sony Computer Entertainment			3	0.005465	1	1	0.005268
14	Majesco Entertainment			3	0.005465	1	1	0.005268
15	Konami Digital Entertainment			3	0.005465	1	1	0.005268
16	Ubisoft			3	0.005465	1	1	0.005268
17	505 Games			3	0.005465	1	1	0.005268
18	Eidos Interactive			3	0.005465	1	1	0.005268
19	Microsoft Game Studios			3	0.005465	1	1	0.005268
20	Atari			3	0.005465	1	1	0.005268
21	D3Publisher			3	0.005465	1	1	0.005268
22	Banpresto			3	0.005465	1	1	0.005268

File **MainGraph\_Gephi\_Edges.xlsx** có các kết quả của độ đo **Betweenness Centrality** theo edge:

	A	B	C	D	E	F	G	H
1	Source	Target	Type	Id	Label	timeset	Weight	edgebetweenness
2	Nintendo	Number None	Undirected	39			1	1
3	Nintendo	Cygames	Undirected	120			1	1
4	Nintendo	Rain Games	Undirected	170			1	1
5	Nintendo	DreamWorks Interactive	Undirected	294			1	1
6	Nintendo	Intergrow	Undirected	462			1	1
7	Nintendo	Yacht Club Games	Undirected	480			1	1
8	Nintendo	System 3	Undirected	485			1	1
9	Microsoft Game Studios	Number None	Undirected	616			1	1
10	Microsoft Game Studios	Cygames	Undirected	697			1	1
11	Microsoft Game Studios	Rain Games	Undirected	747			1	1
12	Microsoft Game Studios	DreamWorks Interactive	Undirected	867			1	1
13	Microsoft Game Studios	Intergrow	Undirected	1038			1	1
14	Microsoft Game Studios	Yacht Club Games	Undirected	1056			1	1
15	Microsoft Game Studios	System 3	Undirected	1061			1	1
16	Take-Two Interactive	Number None	Undirected	1192			1	1
17	Take-Two Interactive	Cygames	Undirected	1273			1	1
18	Take-Two Interactive	Rain Games	Undirected	1323			1	1
19	Take-Two Interactive	DreamWorks Interactive	Undirected	1443			1	1
20	Take-Two Interactive	Intergrow	Undirected	1613			1	1
21	Take-Two Interactive	Yacht Club Games	Undirected	1631			1	1
22	Take-Two Interactive	System 3	Undirected	1636			1	1

## PHẦN B. XỬ LÝ BẰNG PHƯƠNG PHÁP THỦ CÔNG

Đồ thị sử dụng ở phần này là đồ thị 1 phía được trích ra **5 node** từ đồ thị đầy đủ của phần lập trình.

Các phương pháp và thuật toán sử dụng trong đồ án này ở phần thủ công cũng tương tự như 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 dùng để xếp hạng các đỉnh trong đồ thị.
- Tính toán độ đo Betweenness Centrality ở các đỉnh (nodes) và cạnh (edges).
- Tính toán độ đo Closeness Centrality ở các đỉnh (nodes).

File source code sử dụng để phục vụ cho phần này: **SimplerGraph.ipynb**.

Dataset sử dụng để phục vụ cho phần này: **for\_simpler\_graph.csv**.

Kết quả bên dưới đều có trong file **Results.xlsx**, trừ Louvain.

### Phương pháp tách đồ thị thành đồ thị nhỏ hơn với 5 node:

- Đầu tiên, giả sử tách được thành 1 đồ thị có 5 node A, B, C, D, E. Ta không thể lấy ngẫu nhiên các node từ đồ thị gốc được vì sẽ không đảm bảo đồ thị sẽ chặt chẽ về quan hệ. Để đồ thị tương đối chặt chẽ về quan hệ ta sẽ chọn 1 node A là node có đủ quan hệ với cả B, C, D, E theo thứ tự với thể loại I, II, III, IV. Sau đó chọn 1 node B có quan hệ với C ở thể loại II, 1 node C có quan hệ với D ở thể loại III, 1 node D có quan hệ với E ở thể loại IV và 1 node E bất kì có thể loại IV. Như vậy đồ thị sẽ được chặt chẽ hơn.
- Áp dụng phân tích trên vào đồ thị gốc, ta thấy node **Nintendo** là nhà phát hành lớn, luôn đứng đầu ở mọi độ đo nên sẽ có chung đa dạng được nhiều thể loại với nhiều nhà phát hành khác nên có thể chọn A = **Nintendo**. Dùng phần mềm *Excel* với chức năng tìm kiếm trên dataset gốc, ta có thể lọc ra các thể loại I, II, III, IV mà A có là **Sports**, **Racing**, **Misc**, **Action** nên ta có **I = Sports**, **II = Racing**, **III = Action**, **IV = Misc**. Tiếp tục tìm kiếm, ta dễ dàng tìm được B, C, D, E thoả mãn lần lượt là **B = Electronic Arts**, **C = Activision**, **D = Sony Computer Entertainment** và **E = MTV Games**.

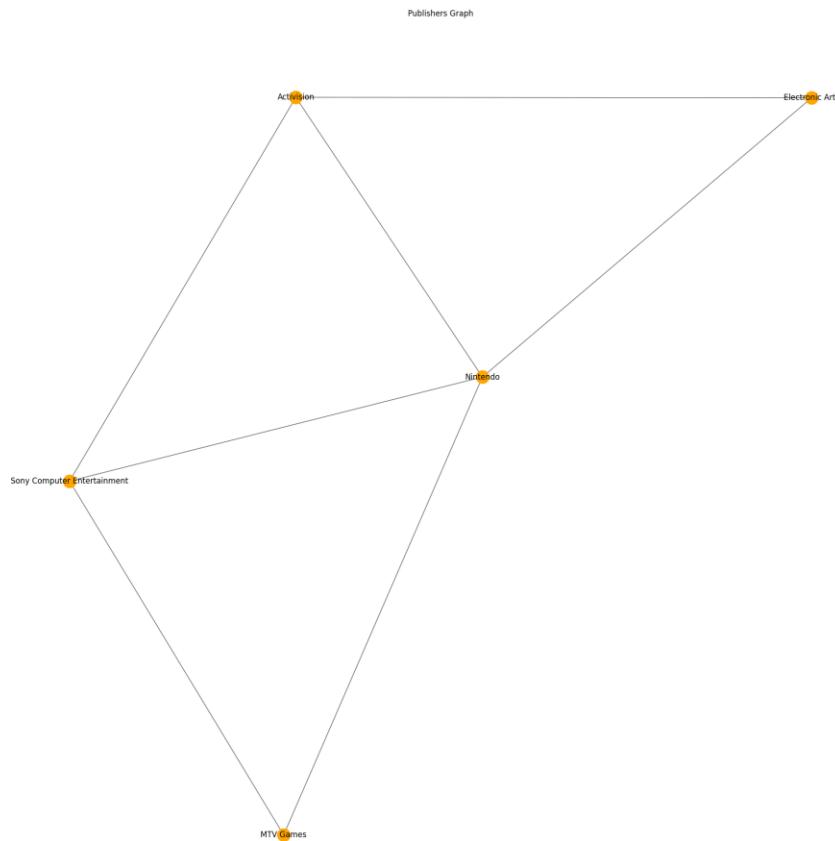
Áp dụng lý thuyết trên code trên Python, ta tạo được dataframe mới (bằng cách chọn những dòng đầu tiên thỏa mãn trong dataframe gốc) tương ứng cho đồ thị mới đúng như những yêu cầu cần thiết:

Create a simpler dataframe which could only form 5 separate nodes based on the default dataframe

```
▷ < df = pd.read_csv('vgsales.csv', usecols = ['Genre', 'Publisher'])
df_temp = df.dropna().drop_duplicates()
# Get indices for the simpler graph nodes - each index is the first row that could be found from the
# dataframe that meets the condition
# 1st node - Nintendo
id1 = df_temp[(df_temp['Publisher']=='Nintendo') & (df_temp['Genre']=='Sports')].index.values.astype(int)[0]
id2 = df_temp[(df_temp['Publisher']=='Nintendo') & (df_temp['Genre']=='Racing')].index.values.astype(int)[0]
id3 = df_temp[(df_temp['Publisher']=='Nintendo') & (df_temp['Genre']=='Action')].index.values.astype(int)[0]
id4 = df_temp[(df_temp['Publisher']=='Nintendo') & (df_temp['Genre']=='Misc')].index.values.astype(int)[0]
# 2nd node - Electronic Arts
id5 = df_temp[(df_temp['Publisher']=='Electronic Arts') & (df_temp['Genre']=='Sports')].index.values.astype(int)[0]
id6 = df_temp[(df_temp['Publisher']=='Electronic Arts') & (df_temp['Genre']=='Racing')].index.values.astype(int)[0]
# 3rd node - Activision
id7 = df_temp[(df_temp['Publisher']=='Activision') & (df_temp['Genre']=='Racing')].index.values.astype(int)[0]
id8 = df_temp[(df_temp['Publisher']=='Activision') & (df_temp['Genre']=='Action')].index.values.astype(int)[0]
# 4th node - Sony Computer Entertainment
id9 = df_temp[(df_temp['Publisher']=='Sony Computer Entertainment') & (df_temp['Genre']=='Action')].index.values.astype(int)[0]
id10 = df_temp[(df_temp['Publisher']=='Sony Computer Entertainment') & (df_temp['Genre']=='Misc')].index.values.astype(int)[0]
# 5th node - MTV Games
id11 = df_temp[(df_temp['Publisher']=='MTV Games') & (df_temp['Genre']=='Misc')].index.values.astype(int)[0]
# Create a list for indices
indicesList = list((id1, id2, id3, id4, id5, id6, id7, id8, id9, id10, id11))
# Create a new dataframe based on these indices from the default dataframe
df_new = df_temp.iloc[df_temp.index.isin(indicesList)]
df_new
```

...	Genre	Publisher
0	Sports	Nintendo
2	Racing	Nintendo
7	Misc	Nintendo
45	Action	Nintendo
77	Sports	Electronic Arts
104	Racing	Electronic Arts
115	Action	Sony Computer Entertainment
168	Misc	Sony Computer Entertainment
241	Action	Activision
427	Misc	MTV Games
1622	Racing	Activision

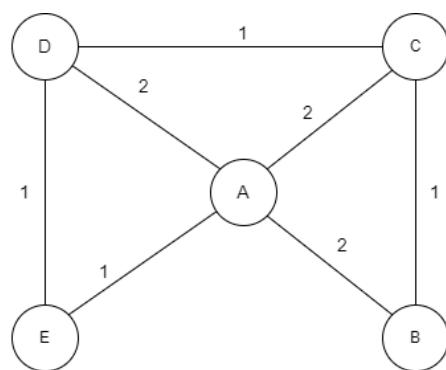
Tương tự như phần lập trình, ta vẽ thử đồ thị 1 phía:



Tương tự như phần lập trình, ta xuất dữ liệu ra dataset để kiểm tra trọng số của các cạnh. Kết quả như trong file **for\_simpler\_graph.csv** là:

source	target	weight
Nintendo	Sony Computer Entertainment	2
Nintendo	Electronic Arts	2
Nintendo	MTV Games	1
Nintendo	Activision	2
Electronic Arts	Activision	1
Sony Computer Entertainment	MTV Games	1
Sony Computer Entertainment	Activision	1

Giản hoá đồ thị với tên gọi A, B, C, D ở trên và thêm trọng số, ta có đồ thị:



Vì là đồ thị vô hướng nên 1 cạnh mặc định đều có 2 chiều. Sử dụng trọng số để tính toán (nếu cần).

Bắt đầu tính toán thủ công:

### 1. PageRank

Công thức tính PageRank của node X trong đồ thị:

$$PR(X) = (1-d) + d \cdot \{ [W(XT_i) \cdot PR(T_i)] / C(T_i) + \dots + [W(XT_n) \cdot PR(T_n)] / C_n \}$$

Trong đó:

- $PR(X)$  là PageRank của X,
- $W(XT_i)$  là trọng số của cạnh  $XT_i$ ,
- $PR(T_i)$  là PageRank của  $T_i$  trỏ tới X,
- $C(T_i)$  là số cạnh mà  $T_i$  trỏ đi,
- $d$  là hệ số, mặc định có thể lấy là 0.85.

Do tổng các PR luôn luôn bằng 1 mà đồ thị có 5 node nên ban đầu giả định mỗi node đều có  $PR = 1 / 5 = 0.2$ .

Bảng giá trị PageRank tính tay qua các lần lặp (có sử dụng **trọng số**) được import từ file Excel:

Iteration	A	B	C	D	E	
0	0.2	0.2	0.2	0.2	0.2	
1	0.51833	0.24107	0.29774	0.32607	0.21679	
2	0.64386	0.33915	0.41347	0.43128	0.28223	
3	0.82116	0.39423	0.49411	0.51418	0.31983	
4	0.93785	0.45442	0.57038	0.59035	0.35897	
5	1.05338	0.49897	0.63196	0.65153	0.38933	
6	1.1437	0.54011	0.68565	0.70558	0.41636	
7	1.22429	0.57346	0.73072	0.75041	0.43881	
8	1.29094	0.60261	0.76927	0.7891	0.45813	
9	1.34849	0.62698	0.80194	0.82169	0.47444	
10	1.39697	0.6479	0.82974	0.84954	0.48835	
11	1.43839	0.66558	0.85336	0.87313	0.50016	
12	1.47349	0.68066	0.87345	0.89323	0.5102	
13	1.50338	0.69346	0.89051	0.91029	0.51874	
14	1.52876	0.70434	0.90502	0.9248	0.52599	
15	1.55035	0.71359	0.91736	0.93713	0.53216	
16	1.56869	0.72145	0.92784	0.94762	0.5374	
17	1.58429	0.72813	0.93675	0.95653	0.54185	
18	1.59754	0.73381	0.94432	0.9641	0.54564	
19	1.6088	0.73864	0.95076	0.97054	0.54886	

20	1.61838	0.74275	0.95623	0.97601	0.55159	
21	1.62652	0.74623	0.96088	0.98066	0.55392	
22	1.63344	0.7492	0.96484	0.98462	0.5559	
23	1.63932	0.75172	0.9682	0.98798	0.55758	
24	1.64432	0.75386	0.97105	0.99083	0.55901	
25	1.64857	0.75568	0.97348	0.99326	0.56022	
26	1.65218	0.75723	0.97554	0.99532	0.56125	
27	1.65525	0.75855	0.9773	0.99708	0.56213	
28	1.65786	0.75966	0.97879	0.99857	0.56287	
29	1.66008	0.76062	0.98006	0.99984	0.56351	
30	1.66196	0.76142	0.98113	1.00091	0.56405	
31	1.66356	0.76211	0.98205	1.00183	0.5645	
32	1.66492	0.76269	0.98283	1.00261	0.56489	
33	1.66608	0.76319	0.98349	1.00327	0.56522	
34	1.66707	0.76361	0.98405	1.00383	0.5655	
<b>35</b>	<b>1.6679</b>	<b>0.76397</b>	<b>0.98453</b>	<b>1.00431</b>	<b>0.56574</b>	<b>(Stop)</b>
<b>Normalisation</b>	<b>0.33449</b>	<b>0.15321</b>	<b>0.19744</b>	<b>0.20141</b>	<b>0.11346</b>	
<b>Rank</b>	<b>1</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>5</b>	

Ta thấy thuật toán dừng ở lần lặp **35** do các kết quả PR của lần lặp này so với lần trước đó là nhỏ hơn  $\varepsilon = 0.001$ . Do tổng PR của tất cả các node phải bằng 1 nên ta phải normalise các kết quả bằng cách lấy từng kết quả PR chia cho tổng kết quả PR của tất cả các node.

Kiểm tra kết quả PageRank tính tay với thuật toán trên Python tương tự như phần lập trình thì thấy kết quả gần như tương đồng và thứ hạng các node đều chính xác:

Node ranking algorithms

1. PageRank

```

D ▾
  pr = nx.pagerank(g_2)
  for k, v in sorted(pr.items(), key=lambda item: item[1], reverse=True):
    print(k, ":", v)

```

[142]

... Nintendo : 0.33452934459428385  
Sony Computer Entertainment : 0.20140338818644235  
Activision : 0.19744768609560442  
Electronic Arts : 0.1532000095883099  
MTV Games : 0.11341957153535934

### Kết luận:

Rank	Node	Publisher	PageRank
1	A	Nintendo	0.33449
2	D	Sony Computer Entertainment	0.20141
3	C	Activision	0.19744
4	B	Electronic Arts	0.15321
5	E	MTV Games	0.11346

## 2. Tính toán các độ đo

### Betweenness Centrality cho đỉnh:

Công thức tính Betweenness Centrality của node  $v$  trong đồ thị:

$$BC(v) = \sum_{u, w \in V} \left( \frac{\sigma_{uw}(v)}{\sigma_{uw}} \right)$$

Trong đó:

- $\sigma_{uw}$  là tổng số đường đi ngắn nhất giữa node  $u$  và  $w$ ,
- $\sigma_{uw}(v)$  là tổng số đường đi ngắn nhất giữa node  $u$  và  $w$  mà có đi ngang qua node  $v$ .

Để cho đơn giản, ta không sử dụng trọng số trong tính toán như trong Python.

Bảng giá trị Betweenness Centrality theo node tinh tay được import từ file Excel:

	A	B	C	D	E	F
1						
2						
3	AB			0	0	0
4	AC		0		0	0
5	AD		0	0		0
6	AE		0	0	0	
7	BC	0			0	0
8	BD	0.5		0.5		0
9	BE	1		0	0	
10	CD	0	0			0
11	CE	0.5	0		0.5	
12	DE	0	0	0		
13	Betweenness Centrality	2	0	0.5	0.5	0
14	Normalisation	0.3333333333	0	0.0833333333	0.0833333333	0
15	Rank	1	3	2	2	3

Do đây là đồ thị vô hướng nên kết quả có thể được normalise bằng cách nhân với 1 lượng là  $\frac{2}{(n-1)(n-2)} = \frac{2}{(5-1)(5-2)} = \frac{1}{6}$ , với  $n$  chính là số node của đồ thị (để giống trong Python).

So sánh với kết quả trong Python thì đều chính xác:

1. Betweenness centrality

a. For nodes

```

D ▾
  bet_centrality_nodes = nx.betweenness_centrality(g_2)
  for k, v in sorted(bet_centrality_nodes.items(), key=lambda item: item[1], reverse=True):
    print(k, ":", v)
[144]                                         Python
...
  Nintendo : 0.3333333333333333
  Sony Computer Entertainment : 0.0833333333333333
  Activision : 0.0833333333333333
  Electronic Arts : 0.0
  MTV Games : 0.0

```

## Kết luận:

Rank	Node	Publisher	Betweenness Centrality
1	A	Nintendo	0.33333
2	D	Sony Computer Entertainment	0.08333
2	C	Activision	0.08333
3	B	Electronic Arts	0
3	E	MTV Games	0

### **Betweenness Centrality cho cạnh:**

Công thức tính Betweenness Centrality của cạnh  $v$  trong đồ thị:

$$BC(v) = \sum_{u, v \in V} \left( \frac{\sigma_{uw(v)}}{\sigma_{uw}} \right)$$

Trong đó:

- $\sigma_{uw}$  là tổng số đường đi ngắn nhất giữa cạnh  $u$  và  $w$ ,
  - $\sigma_{uw(v)}$  là tổng số đường đi ngắn nhất giữa cạnh  $u$  và  $w$  mà có đi ngang qua cạnh  $v$ .

Để cho đơn giản, ta không sử dụng trọng số trong tính toán như trong Python.

Bảng giá trị Betweenness Centrality theo edge tính tay được import từ file Excel:

	A	B	C	D	E	F	G	H
1		AB	AC	AD	AE	BC	CD	DE
2		sum(uw cross v) / sum(uw)						
3	AB	1	0	0	0	0	0	0
4	AC	0	1	0	0	0	0	0
5	AD	0	0	1	0	0	0	0
6	AE	0	0	0	1	0	0	0
7	BC	0	0	0	0	1	0	0
8	BD	0.5	0	0.5	0	0.5	0.5	0
9	BE	1	0	0	1	0	0	0
10	CD	0	0	0	0	0	1	0
11	CE	0	0.5	0	0.5	0	0.5	0.5
12	DE	0	0	0	0	0	0	1
13	Betweenness Centrality	2.5	1.5	1.5	2.5	1.5	2	1.5
14	Normalisation	0.25	0.15	0.15	0.25	0.15	0.2	0.15
15	Rank	1	3	3	1	3	2	3

Do đây là đồ thị vô hướng nên kết quả có thể được normalise bằng cách nhân với 1 lượng là

$\frac{2}{n(n-1)} = \frac{2}{5(5-1)} = \frac{1}{10}$ , với  $n$  chính là số node của đồ thị (để giống trong Python).

So sánh với kết quả trong Python thì đều chính xác:

b. For edges

```
[23] bet_centrality_edges = nx.edge_betweenness_centrality(g_2)
    for k, v in sorted(bet_centrality_edges.items(), key=lambda item: item[1], reverse=True):
        print(k, ":", v)
    ✓ 0.3s
```

Python

```
... ('Nintendo', 'Electronic Arts') : 0.25
('Nintendo', 'MTV Games') : 0.25
('Sony Computer Entertainment', 'Activision') : 0.2
('Nintendo', 'Sony Computer Entertainment') : 0.1500000000000002
('Nintendo', 'Activision') : 0.1500000000000002
('Electronic Arts', 'Activision') : 0.1500000000000002
('Sony Computer Entertainment', 'MTV Games') : 0.1500000000000002
```

### Kết luận:

<b>Rank</b>	<b>Edge</b>	<b>Publishers</b>	<b>Betweeness Centrality</b>
1	AB	Nintendo, Electronic Arts	0.25
1	AE	Nintendo, MTV Games	0.25
2	CD	Sony Computer Entertainment, Activision	0.2
3	AD	Nintendo, Sony Computer Entertainment	0.15
3	AC	Nintendo, Activision	0.15
3	BC	Electronic Arts, Activision	0.15
3	DE	Sony Computer Entertainment, MTV Games	0.15

### Closeness Centrality:

Để cho đơn giản, ta không sử dụng trọng số trong tính toán như trong Python.

Bảng giá trị Closeness Centrality tinh tay được import từ file *Excel*:

	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
<b>A</b>	0	1	1	1	1
<b>B</b>	1	0	1	2	2
<b>C</b>	1	1	0	1	2
<b>D</b>	1	2	1	0	1
<b>E</b>	1	2	2	1	0
<b>Sum of distances</b>	4	6	5	5	6
<b>Closeness Centrality</b>	<b>0.25</b>	<b>0.166667</b>	<b>0.2</b>	<b>0.2</b>	<b>0.166667</b>
<b>Normalisation</b>	<b>1</b>	<b>0.666667</b>	<b>0.8</b>	<b>0.8</b>	<b>0.666667</b>
<b>Rank</b>	<b>1</b>	<b>3</b>	<b>2</b>	<b>2</b>	<b>3</b>

Do đây là đồ thị vô hướng nên kết quả có thể được normalise bằng cách nhân với 1 lượng là  $n-1=5-1=4$ , với  $n$  chính là số node của đồ thị (để giống trong Python).

So sánh với kết quả trong Python thì đều chính xác:

## 2. Closeness centrality

```

clo_centrality = nx.closeness_centrality(g_2)
for k, v in sorted(clo_centrality.items(), key=lambda item: item[1], reverse=True):
    print(k, ":", v)
[24]   ✓ 0.2s
...
...  Nintendo : 1.0
Sony Computer Entertainment : 0.8
Activision : 0.8
Electronic Arts : 0.6666666666666666
MTV Games : 0.6666666666666666

```

Python

**Kết luận:**

<b>Rank</b>	<b>Node</b>	<b>Publisher</b>	<b>Closeness Centrality</b>
1	A	Nintendo	1
2	D	Sony Computer Entertainment	0.8
2	C	Activision	0.8
3	B	Electronic Arts	0.666667
3	E	MTV Games	0.666667

**Phân cụm bằng thuật toán Louvain (tham khảo ở:  
[https://en.wikipedia.org/wiki/Louvain\\_method](https://en.wikipedia.org/wiki/Louvain_method)):**

Thuật toán Louvain được dựa trên việc làm cho độ đo  $Q$  (modularity) trong đồ thị lớn nhất, và được tính bằng công thức như sau:

$$Q = \frac{1}{2m} \cdot \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \cdot \delta(c_i, c_j)$$

Trong đó:

- $Q$  là modularity,
- $m$  là tổng trọng số của tất cả các cạnh trong đồ thị,
- $A_{ij}$  là trọng số cạnh giữa 2 node  $i$  và  $j$ ,
- $k_i, k_j$  là bậc của các node  $i$  và  $j$ ,
- $\delta(c_i, c_j)$  bằng 1 cho biết 2 node  $i$  và  $j$  ở cùng 1 cụm, ngược lại bằng 0 thì khác cụm.

Từ công thức trên, ta rút ra được công thức tính modularity trong 1 cụm, đó là:

$$Q_c = \frac{\sum in}{2m} - \left( \frac{\sum tot}{2m} \right)^2$$

Trong đó:

- $Q_c$  là modularity trong cụm,
- $m$  là tổng trọng số của tất cả các cạnh trong đồ thị,
- $\sum in$  là tổng trọng số của các cạnh trong cụm (không có cạnh ngoài)
- $\sum tot$  là tổng trọng số của các cạnh tạo từ toàn bộ node trong cụm (có cả cạnh ngoài).

Vậy một node  $i$  muốn khảo sát xem nó có thể tham gia vào 1 cụm hay không chỉ cần phát triển thêm công thức ở trên để tính được  $\Delta Q$  của nó với mỗi cụm cần khảo sát. Kết quả  $\Delta Q$  của cụm nào lớn nhất cho node  $i$  thì node  $i$  có thể tham gia vào cụm đó. Công thức như sau:

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

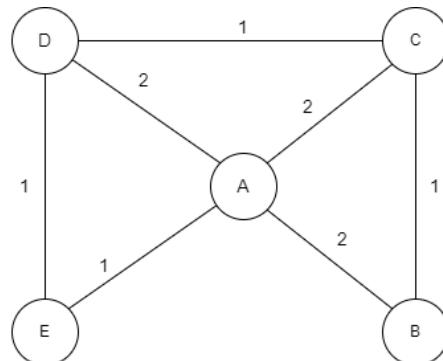
Trong đó có các đại lượng mới là:

- $k_{i,in}$  là tổng trọng số giữa cạnh mà node  $i$  liên kết với cụm và toàn bộ cạnh trong cụm,
- $k_i$  là tổng trọng số của các cạnh được tạo từ node  $i$ .

Do công thức là khá phức tạp nên rất khó để đưa vào Excel, do vậy thuật toán được áp dụng như các bước được trình bày dưới đây:

Đầu tiên, đồ thị có 5 node nên mỗi node theo thuật toán là 1 cụm riêng biệt.

Nhắc lại đồ thị:



Xuất phát với A, áp dụng công thức (\*) để tính, ta có kết quả sau:

$$\Delta Q_{AB} = \left[ \frac{0+2.2}{2.10} - \left( \frac{3+7}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{3}{2.10} \right)^2 - \left( \frac{7}{2.10} \right)^2 \right] = 0.095$$

$$\Delta Q_{AC} = \left[ \frac{0+2.2}{2.10} - \left( \frac{4+7}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{4}{2.10} \right)^2 - \left( \frac{7}{2.10} \right)^2 \right] = 0.06$$

$$\Delta Q_{AD} = \left[ \frac{0+2.2}{2.10} - \left( \frac{4+7}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{4}{2.10} \right)^2 - \left( \frac{7}{2.10} \right)^2 \right] = 0.06$$

$$\Delta Q_{AE} = \left[ \frac{0+2.1}{2.10} - \left( \frac{2+7}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{2}{2.10} \right)^2 - \left( \frac{7}{2.10} \right)^2 \right] = 0.03$$

Do  $\Delta Q_{AB}$  lớn nhất nên A và B về 1 cụm.

Lúc này xét tiếp C với các cụm A, B và D với công thức (\*):

$$\Delta Q_{CA} = \left[ \frac{0+2.2}{2.10} - \left( \frac{7+4}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{7}{2.10} \right)^2 - \left( \frac{4}{2.10} \right)^2 \right] = 0.06$$

$$\Delta Q_{CB} = \left[ \frac{0+2.1}{2.10} - \left( \frac{3+4}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{3}{2.10} \right)^2 - \left( \frac{4}{2.10} \right)^2 \right] = 0.04$$

$$\Delta Q_{CD} = \left[ \frac{0+2.1}{2.10} - \left( \frac{4+4}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{4}{2.10} \right)^2 - \left( \frac{4}{2.10} \right)^2 \right] = 0.02$$

Do  $\Delta Q_{CA}$  lớn nhất nên C và về cụm của A, B.

Tiếp tục với D và áp dụng công thức (\*) với các cụm C, A và E:

$$\Delta Q_{DC} = \left[ \frac{0+2.1}{2.10} - \left( \frac{4+4}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{4}{2.10} \right)^2 - \left( \frac{4}{2.10} \right)^2 \right] = 0.02$$

$$\Delta Q_{DA} = \left[ \frac{0+2.2}{2.10} - \left( \frac{7+4}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{7}{2.10} \right)^2 - \left( \frac{4}{2.10} \right)^2 \right] = 0.06$$

$$\Delta Q_{DE} = \left[ \frac{0+2.1}{2.10} - \left( \frac{2+4}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{2}{2.10} \right)^2 - \left( \frac{4}{2.10} \right)^2 \right] = 0.06$$

Do  $\Delta Q_{DA} = \Delta Q_{DE}$  nên D có thể thuộc cụm C, A, B hoặc E.

Cuối cùng xét D với công thức (\*):

$$\Delta Q_{ED} = \left[ \frac{0+2.1}{2.10} - \left( \frac{4+2}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{4}{2.10} \right)^2 - \left( \frac{2}{2.10} \right)^2 \right] = 0.06$$

$$\Delta Q_{EA} = \left[ \frac{0+2.1}{2.10} - \left( \frac{7+2}{2.10} \right)^2 \right] - \left[ \frac{0}{2.10} - \left( \frac{7}{2.10} \right)^2 - \left( \frac{2}{2.10} \right)^2 \right] = 0.03$$

Do  $\Delta Q_{ED}$  lớn nhất nên E thuộc về cụm với D, như vậy D và E có thể tạo thành riêng 1 cụm và D không thuộc chung cụm với C, A, B.

Vậy đồ thị gồm 2 cụm là C, A, B và D, E, là số cụm tốt nhất có thể phân tách nên không cần chạy tiếp bước tiếp theo của thuật toán (tức là tiếp tục gom cụm trong các cụm mới được tạo thành).

Kiểm tra với kết quả trên Python như phần lập trình thì thấy kết quả tính tay hoàn toàn chính xác:

Communities detection with Louvain algorithm

```

import matplotlib.cm as cm
import matplotlib
from community import community_louvain

plt.figure(figsize = (25, 25))

# Compute the best partition
partition = community_louvain.best_partition(g_2)

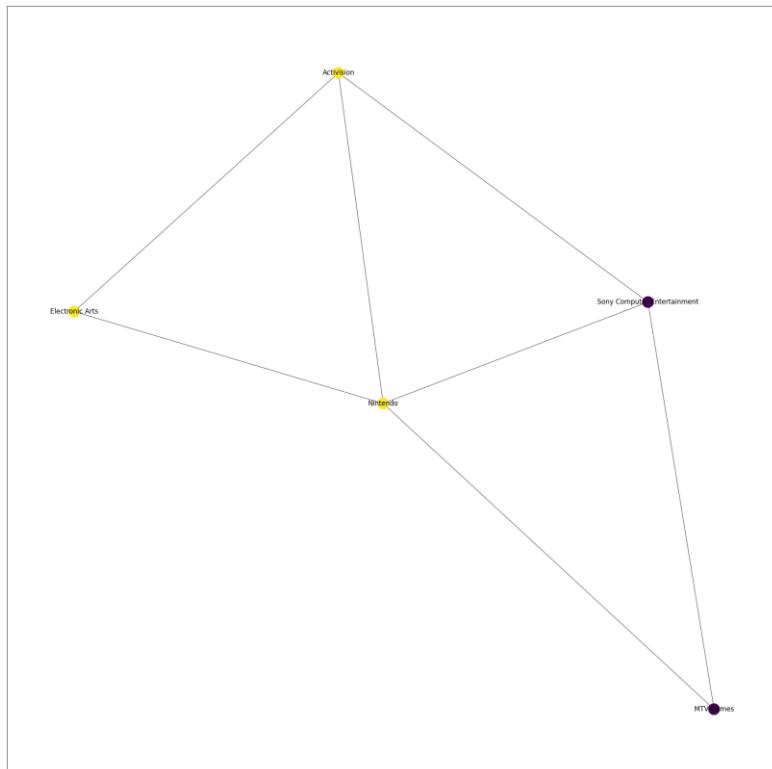
# Draw the graph
pos = nx.spring_layout(g_2)
# Color the nodes according to their partitions
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(g_2, pos, partition.keys(), node_size = 400, cmap = cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(g_2, pos, edge_color = '#5A5A5A')
nx.draw_networkx_labels(g_2, pos)

plt.show()

```

[52] ✓ 0.3s

Python



```

▷ <
    for i in range(len(np.unique(values))):
        print("-----", "Community ", i + 1, "-----")
        print("")
        for name, k in partition.items():
            if k == i:
                print(name)
        print("")
[54]   ✓ 0.6s
...
----- Community  1 -----
Sony Computer Entertainment
MTV Games

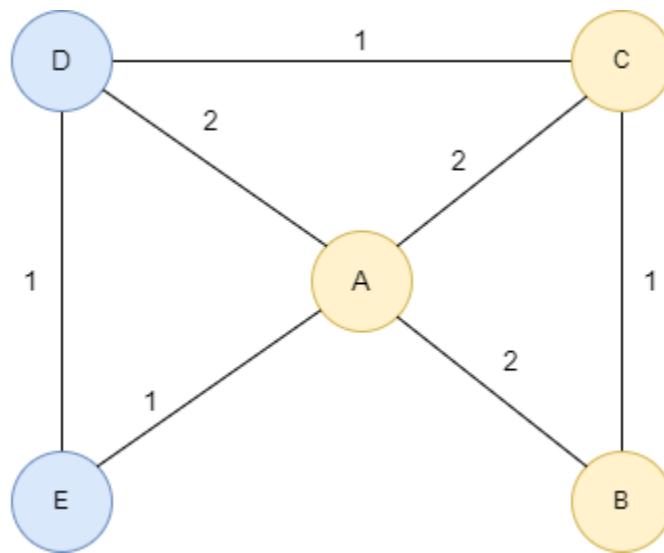
----- Community  2 -----
Nintendo
Electronic Arts
Activision

```

### Kết luận:

Community	Nodes	Publishers
1	A	Nintendo
	B	Electronic Arts
	C	Activision
2	D	Sony Computer Entertainment
	E	MTV Games

Các cụm trong đồ thị (được giản lược):



## KẾT LUẬN VỀ ĐỀ TÀI

1. Qua đề tài, bằng việc kết hợp với kiến thức đã được học để thực hiện xử lý trên một đồ thị thực tế với dữ liệu rất lớn, em đã có thể hiểu hơn về việc ý nghĩa của các thuật toán xếp hạng, phân cụm và các độ đo là quan trọng như thế nào trong một bài toán cụ thể. Chính nhờ chúng mà ta mới nắm bắt được hầu như toàn bộ các thông tin quan trọng của đồ thị, từ các node đóng vai trò quan trọng như thế nào đến đóng vai trò kém quan trọng như thế nào thông qua các mối quan hệ của chúng. Cụ thể ở đây chính là thị trường trò chơi điện tử với rất nhiều các nhà phát hành game và thể loại của họ, khi mới nhìn vào tập dữ liệu rất lớn đó, ta hầu như bị “mù tịt” về thông tin, nhưng nhờ các phương thức xử lý đã áp dụng mà ta có thể hiểu rõ và đánh giá được đồ thị một cách khái quát nhất. Đó là giá trị từ lý thuyết đến thực tiễn. Bên cạnh đó ta còn được cung cấp nhiều thông tin bổ ích khác về đồ thị.

2. Phần tính toán thủ công với đồ thị nhỏ được tách ra từ đồ thị lớn cho ta ý nghĩa hầu như về cả đồ thị lớn vì các kết quả ở đồ thị nhỏ nếu đem so với kết quả trong đồ thị lớn thì chúng hầu như cho cùng một kết quả về xếp hạng. Điều này cho ta biết có thể đánh giá sơ bộ đồ thị gốc rất đồ sộ một cách khá hiệu quả khi ta chỉ phải thông qua một đồ thị có quy mô nhỏ hơn rất nhiều đã được trích ra từ đồ thị gốc ấy.

## TÀI LIỆU THAM KHẢO

- [1] Tài liệu của giảng viên cung cấp trong quá trình học.
- [2] *PageRank, NetworkX*,  
[https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.link\\_analysis.page\\_rank\\_alg.page\\_rank.html](https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.link_analysis.page_rank_alg.page_rank.html).
- [3] *Centrality, NetworkX*,  
<https://networkx.org/documentation/stable/reference/algorithms/Centrality.html>.
- [4] *Community, NetworkX*,  
<https://networkx.org/documentation/stable/reference/algorithms/community>.
- [5] *Louvain method, Wikipedia*, [https://en.wikipedia.org/wiki/Louvain\\_method](https://en.wikipedia.org/wiki/Louvain_method).
- [6] <https://www.python.org>.
- [7] <https://docs.jupyter.org/en/latest>.
- [8] *Learn how to use Gephi, Gephi*, <https://gephi.org/users>.
- [9] *Gephi Plugins, GitHub*, <https://github.com/gephi/gephi-plugins/tree/edge-betweenness-plugin> (for *Edge Betweenness Metric*).