Application of Bron-Kerbosch Algorithm in Determining Personalized Steam Game Bundle Offers Based on Purchases from Other Users

Aryo Wisanggeni - 13523100[1]
Informatics Undergraduate Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]arrryow@gmail.com, 13523100@std.stei.itb.ac.id

***Abstract*— Currently, the video game market is one of the largest and most rapidly growing economic sectors globally, contributing billions of dollars annually to the global economy. One of the largest video game providers, Steam, serves as a marketplace that offers players a convenient platform to purchase and manage their games. To provide users with recommendations that match their preference, Steam uses various methods of filtering. With those methods, Steam is able to see how one game is similar to another game and how one user has similar preferences to another user in the platform. Using that data, relational graphs can be made that can be used to make personalized game bundles that are tailored to users' preference. This approach increases the likelihood of users purchasing the bundles, thereby increasing Steam's sales. To identify appealing game bundles for users, the Bron-Kerbosch algorithm can be used to determine maximal cliques of users who have the same game preferences. By utilizing these grouped cliques and users' recent game purchases that match the clique criteria, a personalized game bundle can be created to suit individual users' preferences, making it more enticing for users to purchase.**

***Keywords*—Bron-Kerbosch, Graph Theory, Game Bundles, Recommendation Systems.**

I. INTRODUCTION

The video game industry is a thriving industry that significantly contributes to the global economy. The rapid growth of this industry seems to be expected, especially after the world faced the 2019 pandemic that forced people to stay in door more often and give people more chances to play video games. Even after the end of the pandemic, the industry continues to show growth. This trend is showing to be favorable for game providers as their market is only growing wider by the second. Thus, with the rise of gamer numbers in recent years, video game providers are seeing a huge influx of new users to their platforms.

Steam, one of the largest game providers in the world, is no exception to this. The platform itself offers a vast array of video games that cater to a diverse player base. Everything from casual games all the way to extreme rogue-like games is available on Steam. Other than the vast option for games, the platform also provides various features to enhance user experience. Those features include how users are able to save their video game progress, manage their game downloads, and even give users a sense of accomplishment by displaying the game achievements they obtained in their Steam account. On top of all this, Steam frequently offers games at a discount even up to 90% of its original price.

As a business, Steam employs discounts and bundling to increase its sales. Decreasing the prices works out well for them because of the nature of their products being software. It costs nothing for Steam to "produce" a single copy of software. With that in mind, any money they make from game sales no matter how small will be a profit. Due to this, it is better for them to sell a game that is originally $100 at $10 and have its user buy it rather than not selling it at all. It is part of their business model to create offers that appear to only benefit customers at the company's expense, when in fact, these sales also benefit the company.

Though it is possible for them to keep the prices down and have the games sell at a discount at all times, the company chooses to give discounts and bundling during special times like Halloween, Black Friday, New Years, etc. This is to also make the deals even more special and make it more likely for users to buy. However, this alone is insufficient; the company must generate sales all year round, not only during special occasions. In order to do that, Steam also has small discounts and bundling all year round.

Bundling several games in one package and making it cheaper than just buying the individual games is one way for the company to increase sales during regular days of the year. This is due to users buying games even during ordinary days when there are no holidays or special events. But usually, when a user buys games, they only buy the one game that they are interested in. Steam wants its users to spend more money, so they offer the one game the user wants with another similar game the user might also be interested in, in order to entice them to spend more. For example, a user is interested in buying a $10 game, but upon seeing that game being bundled with another $10 game that they are also curious about, priced at $15, they are likely to purchase the bundle, perceiving it as a great deal since they are saving $5. In this case, the company also wins out, since rather than just receiving $10, they received $15 for the purchase.

In order to make bundles that are interesting to users, Steam needs to carefully choose what games to bundle with other games. One method of doing so is using graph theory, specifically in this case is with Bron-Kerbosch algorithm. Using

this algorithm, Maximal user cliques interested in similar games can be determined, and the purchases of users within these cliques can be used to create bundles that are appealing to other users in the same clique.

II. Theoretical Framework

A. Graphs

Graph is a data structure that represents relationship between variables. A graph is made up of vertices (nodes) that are connected by the edges (lines). In applications using graphs, nodes represent discrete objects like people or places, while the edges describe a relationship between the discrete objects like routes from one place to another or people's relationships.

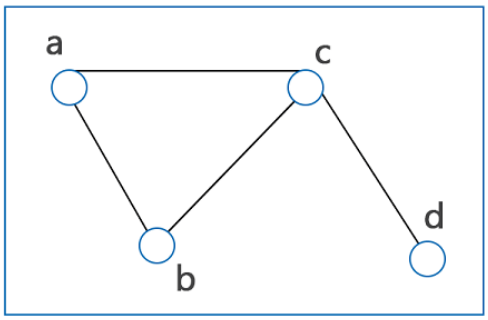


Figure 1. A graph with four vertices and four edges
Source: https://images.javatpoint.com/tutorial/dms/images/graph-theory-in-discrete-mathematics.png

1. Graph Terminologies

   a. Adjacent

   Two vertices are said to be adjacent to each other if there is an edge connecting them. In figure 1, vertex a is adjacent to vertex b and c, but not adjacent to vertex d.

   b. Clique

   A clique is a set of vertices in a graph that are all adjacent to each other. In figure 1, c and d are a clique because c is connected to d and d is connected to c. Another clique that is made of 3 vertices in figure 1 is vertex a, b, and c.

   c. Subgraph

   A subgraph of a graph G is a graph, each of whose vertices belongs to V(G) and each of whose edges belongs to E(G).

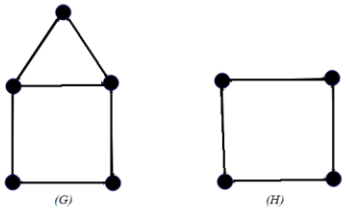


Figure 2. Graph H is a subgraph of G
Source: https://www.researchgate.net

2. Types of Graphs

   a. Undirected Graphs

   Undirected graph is a graph whose edges are not directed. If a vertex A is connected to vertex B via an edge, that edge does neither directs to A nor B.

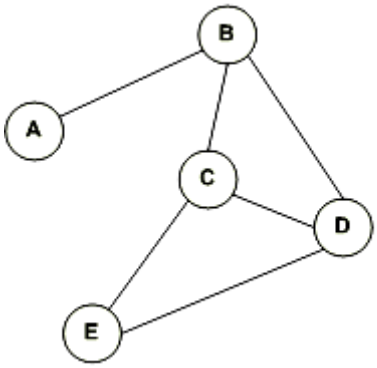


Figure 3. An undirected graph
Source: https://images.javatpoint.com/tutorial/graph-theory/images/types-of-graphs4.png

   b. Directed Graphs

   Directed graph or digraph is a graph whose edges have a direction. If a vertex A is connected to vertex B via an edge, that edge does will point to either A or B. In real life, these directed edges might represent one-way roads that are only traversable in one direction.

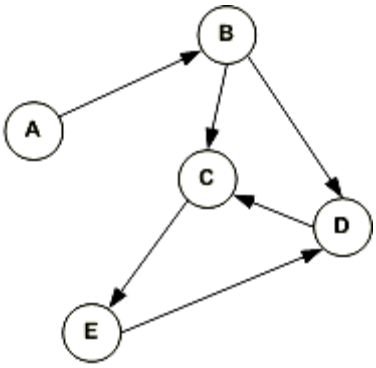


Figure 4. A directed graph
Source: https://images.javatpoint.com/tutorial/graph-theory/images/types-of-graphs5.png

   c. Simple Graphs

   Simple graph is a graph that doesn't have 1) an edge that connects a vertex with itself (loop), 2) two or more edges that connects the same two vertices (multiple edges).

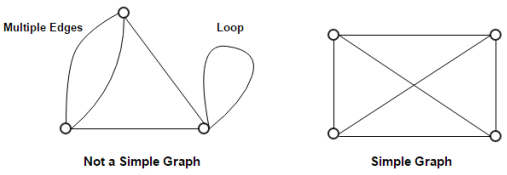


Figure 5. An example of a simple graph and not a simple graph
Source: https://images.javatpoint.com/tutorial/graph-

theory/images/types-of-graphs3.png

d. Regular Graphs

Regular graph is a graph that has the same number of edges for each vertex. The number of edges for each vertex in a regular graph is the degree of the regular graph. In figure 6 below, both graphs are 2-regular graph, because all vertices in both graphs have 2 edges.

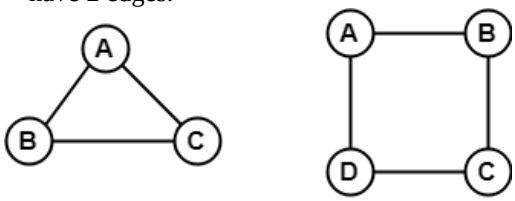


Figure 6. Two examples of a regular graph, both graph has all degree 2 vertices
Source: https://images.javatpoint.com/tutorial/graph-theory/images/types-of-graphs9.png

e. Bipartite Graphs

Bipartite graph is a graph in which the vertex set can be partitioned into two sets such that edges only go between sets, not within

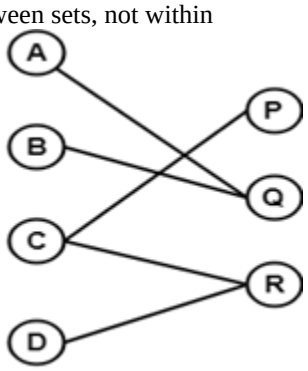
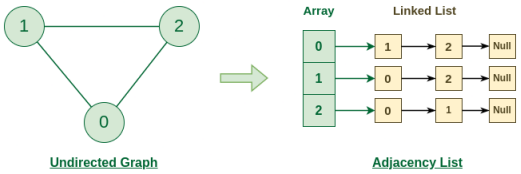


Figure 7. A bipartite graph
Source: https://images.javatpoint.com/tutorial/graph-theory/images/types-of-graphs14.png

3. Graph Representation

a. Adjacency List

Adjacency list represents a graph with an array of lists. The size of array is equal to the number of vertices**.** Each list represents all vertex that is adjacent to the first element of the list.

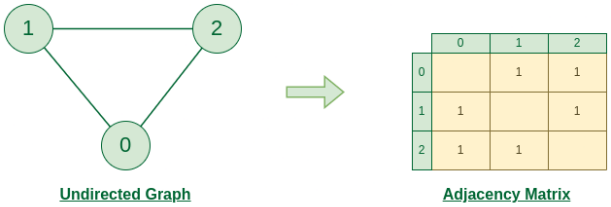


Graph Representation of Undirected graph to Adjacency List

Figure 8. A graph and its equivalent adjacency list
Source: https://geeksforgeeks.org

b. Adjacency Matrix

Adjacency matrix represents a graph as a matrix of Boolean (0 and 1). If there is an edge connecting vertex i to vertex j, then the value in the matrix will be 1, otherwise it would be 0. In an undirected graph, the matrix is always mirrored across the diagonal of the matrix.



Graph Representation of Undirected graph to Adjacency Matrix

Figure 9. A graph and its equivalent adjacency matrix
Source: https://geeksforgeeks.org

B. Bron-Kerbosch Algorithm

Bron-Kerbosch algorithm is an algorithm to find all maximal cliques in an undirected graph. Here are a few variations on the implementation of the Bron-Kerbosch Algorithm.

1. Simple Algorithm

The most basic and standard form of the Bron-Kerbosch Algorithm is with recursive backtracking. The Bron-Kerbosch function has three parameters: R, P, and X. With R being the current clique, P being the candidate set, and X being the exclusion set.

```
Procedure Bron-Kerbosch(R, P, X)
If P and X are empty then
   Output R
For each v in P do
   Bron-Kerbosch(R∪{v}, P∩N(v), X∩N(v))
   P ← P\{v}  {remove v from P}
   X ← X∪{v}  {add v to x}
```

Figure 10. Pseudocode for the simple Bron-Kerbosch algorithm

The way this algorithm works is by checking each node in the graph by recursively calling the function. While for each v in P make up the part of the algorithm that does backtracking. Each successful candidate clique is outputted through R, while the X parameter is there to make sure multiple of the same clique does not get outputted, which prevents copy cliques to be produced. The cliques themselves are created through R that is constantly being added nodes that are adjacent with each other (here N(v) means all nodes that are adjacent with vertex v).

2. Pivoting Algorithm

This is an improved and more efficient version of the Bron-Kerbosch algorithm. The idea is rather than going through all vertices in the P set, a pivot vertex is chosen to start out with and the algorithm will only check vertices that are not adjacent to that pivot vertex. This

way, the algorithm iterates less than the simple version, making it more computationally efficient.

```
Procedure BK-Pivoting(R, P, X)
If P and X are empty then
   Output R
Choose pivot u ∈ P∪X
For each v ∈ P\N(u) do
   BK-Pivoting(R∪{v}, P∩N(v), X∩N(v))
   P ← P\{v}  {remove v from P}
   X ← X∪{v}   {add v to x}
```

Figure 11. Pseudocode for the Bron-Kerbosch algorithm with pivoting

III. Problem Analysis

To successfully implement this system, limitations and definitions need to be set in this study. In that respect, there are four parts that need to be elaborated on: user data, game data, game bundles, and the recommendation system.

A. User Data

The first data type that needs to be established is of users with their respective game preferences. Here are some definitions and limitations that elaborate on this data:

1. What "game preference" means in this study
   In broad terms, game preference can be determined a lot of factors. How a user likes a game can be based on many metrics, such as the game's producer, the story, the art, the music, game genre, the gameplay, and many other variables.
   In this study, to simplify, user preferences will be considered based solely on the game genres with which the users have interacted. This limitation is placed in order to focus the study on the Bron-Kerbosch algorithm rather than classifying user preference.
2. How user preference is determined
   This is the part that is assumed to be already managed by the platform. In Steam, there are two obvious ways to see a user's preference and that is by seeing their game purchases and wish-listed games. Game purchases being the games they have bought in the past, while "wish-list" is a feature in Steam that allows users to put games they want to buy in the future into a list. Thus, if a simple system is made to determine a user's preference, it would be by those two parameters and seeing if those games have a similarity. For example, if a new user bought three horror games and one puzzle game, while they have wish-listed two horror games and two adventure games, the user's most likely favorite genre would be horror, followed by adventure, and lastly by puzzle.
3. Users sharing the same preference
   It will be assumed that two users have the same preference if the two players like the same type of game genre.

B. Game Data

The second data type that needs to be established is of games and their genres. Here games will be limited to only have one genre in this study in order to focus more on the Bron-Kerbosch algorithm and the recommendation system. Furthermore, it will be assumed that none of the users have purchased any of the games listed in the case analysis. This is also to remove excessive if statements that diverts the focus of study.

C. Game Bundles

First is determining the size of the game bundles. To simplify the examples, game bundles will only have two games in each bundle. In this case, it will be necessary to wait for users to purchase games, and based on the genres of two consecutively bought games, the system will determine whether a game bundle will be formed. Here are two cases of this bundle determining system:

1. Games that were bought are of different genres
   Should it be this first case, the system will not return anything, as the two games don't have a correlation. Thus, in this first case, no bundles would be formed.
2. Games that were bought are of the same genre
   In this case, as long as there are two games of the same genre being bought, doesn't matter if it was bought by one person or multiple people, the two games will be made into a bundle.

D. Recommendation system

This bundle recommendation system will be done by integrating the user preference data and the game bundle determining system. The system that will be implemented in this study will be as follows:

The system will be detecting games that are purchased by users, if two different games of the same genre are bought in succession regardless if it was by the same person or not, then those two games will become a bundle. Then, the system will detect the genre of that game bundle and recommend that bundle to other people that have the same preference.

IV. Implementation

The programming language that will be used to implement this system is Python. Here is how every module will be implemented in the program:

A. User and Game Data

To represent both data, in its raw form it can be in the represented as a bipartite graph or adjacency list. Both of these can represent the data well because both have a structure that splits the vertices into two sets. However, adjacency lists will be used, as it is easier to implement in code.

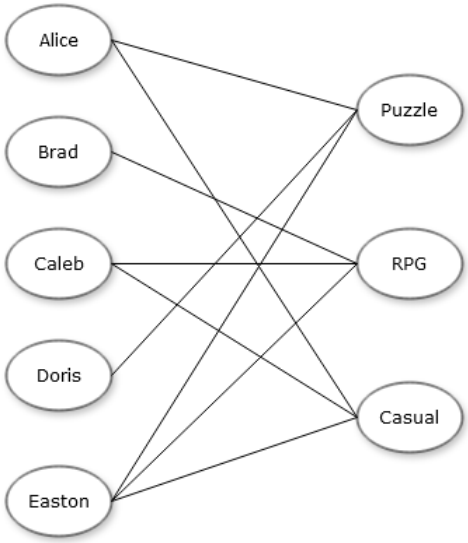


Figure 12. Example implementation of user preference data with bipartite graph

```
preferences = {
    'Alice': ['Puzzle', 'Casual'],
    'Brad': ['RPG'],
    'Caleb': ['RPG', 'Casual'],
    'Doris': ['Puzzle'],
    'Easton': ['Puzzle', 'RPG', 'Casual']
}
```

Figure 13. Example implementation of user preference data with adjacency list

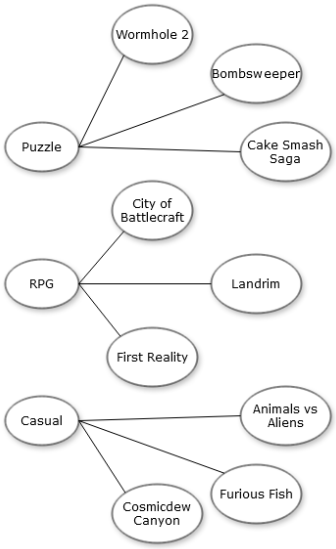


Figure 14. Example implementation of game genre data with bipartite graph

```
games_genre_mapping = {
    'Puzzle': ['Wormhole 2', 'Bombsweeper', 'Cake Smash Saga'],
    'RPG': ['City of Battlecraft', 'Landrim', 'First Reality'],
    'Casual': ['Animals vs Aliens', 'Furious Fish', 'Cosmicdew Canyon']
}
```

Figure 15. Example implementation of game genre data with adjacency list

B. User Preference Graph

In this graph, vertices will represent users. If two users have the same preference, they will be adjacent to each other, meaning that there will be an edge connecting both users. Here is the function used to create this graph.

```
def create_similar_preference_graph(preferences: dict) -> dict:
    output = {} # output dictionary

    for current_person in preferences:
        temp = [] # value array for currentPerson
        for person in preferences:
            if (person != current_person and
                set(preferences[current_person]).intersection(preferences[person])):
                temp.append(person)
        output[current_person] = temp

    return output
```

Figure 16. Function to produce user preference graph from preference adjacency list

```
Alice: ['Caleb', 'Doris', 'Easton']
Brad: ['Caleb', 'Easton']
Caleb: ['Alice', 'Brad', 'Easton']
Doris: ['Alice', 'Easton']
Easton: ['Alice', 'Brad', 'Caleb', 'Doris']
```

Figure 17. Example output printed from the user preference graph function

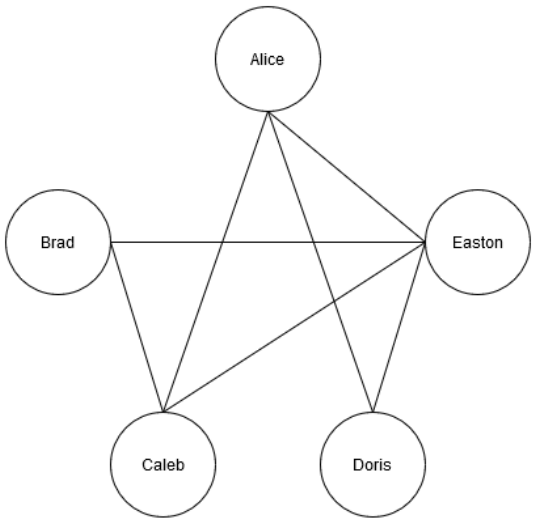


Figure 18. Corresponding graph of the example output of the user preference graph output

C. Bron-Kerbosch Algorithm

Using the graph produced in the previous part, Bron-Kerbosch Algorithm can now be used to segment the graph into different parts by finding the maximal cliques of each game genre. The algorithm that will be using pivoting to increase efficiency. Do note the two extra parameters in the implementation, G is the graph that contains which node is neighboring another node, while cliques is the output variable.

```python
def bron_kerbosch(R: List[str],
                  P: List[str],
                  X: List[str],
                  G: Dict[str, List[str]],
                  cliques: List[List[str]]):
    # if P and X are empty, return maximal clique in R
    if len(P) == 0 and len(X) == 0:
        cliques.append(R.copy())

    # Choosing pivot
    if P:
        pivot = P[0]
    elif X:
        pivot = X[0]
    else:
        return

    # filter P to not include the neighbors of pivot
    P_without_neighbors = [node for node in P if node not in G[pivot]]

    for node in P_without_neighbors:
        # Recursively call the Bron-Kerbosch algorithm
        bron_kerbosch(R + [node],
                      [n for n in P if n in G[node]],
                      [n for n in X if n in G[node]],
                      G,
                      cliques)
        P.remove(node)
        X.append(node)
```

Figure 19. Bron-Kerbosch Algorithm function with pivoting

```
['Alice', 'Caleb', 'Easton']
['Alice', 'Doris', 'Easton']
['Brad', 'Caleb', 'Easton']
```

Figure 20. Bron-Kerbosch function example output

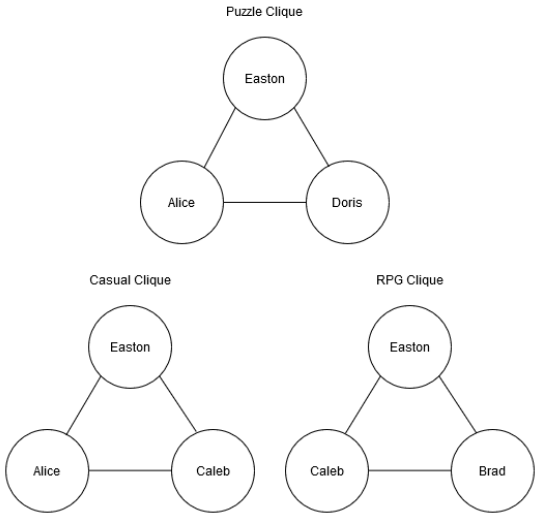


Figure 21. Corresponding subgraphs of cliques of the example output of the Bron-Kerbosch function

D. Labeling Cliques

To simplify the recommendation system implementation, each clique will be labelled with their respective game genre with an intermediary function.

```python
def finalize_cliques(preferences: Dict[str, List[str]],
                     cliques: List[List[str]]) -> Dict[str, List[str]]:
    genre_cliques = {}

    # Loop through cliques
    for clique in cliques:
        genres_in_clique = {}

        # For each person in the clique, gather their preferences
        for person in clique:
            if person in preferences:
                for genre in preferences[person]:
                    if genre in genres_in_clique:
                        # append to list in corresponding key
                        genres_in_clique[genre].add(person)
                    else:
                        # add new key
                        genres_in_clique[genre] = {person}

        # Identify the common genre (the genre with the most people)
        if genres_in_clique:
            # Get the genre with the maximum number of people
            common_genre = max(genres_in_clique,
                               key=lambda genre: len(genres_in_clique[genre]))
            genre_cliques[common_genre] = genres_in_clique[common_genre]

    return genre_cliques
```

Figure 22. Intermediary function to label cliques

```
Casual: {'Alice', 'Caleb', 'Easton'}
Puzzle: {'Alice', 'Doris', 'Easton'}
RPG: {'Caleb', 'Easton', 'Brad'}
```

Figure 23. Intermediary function example output

E. Recommendation System

Here, every module will be utilized and integrated to create the bundle recommendation system.

```python
def recommendation_system(cliques: Dict[str, set],
                          games_genre_mapping: Dict[str, List[str]],
                          purchase1: Tuple[str, str],
                          purchase2: Tuple[str, str]):
    person1, game1 = purchase1
    person2, game2 = purchase2

    # Determine the genre of each game
    genre1 = None
    genre2 = None

    # Find the genres of both games
    for genre, games in games_genre_mapping.items():
        if game1 in games:
            genre1 = genre
        if game2 in games:
            genre2 = genre

    # If genres are different, exit the function
    if genre1 != genre2:
        print(f"{game1} and {game2} are of different genres. Can not be made into a bundle.")
        return

    # If person who bought games are not in genre clique, exit function
    if person1 not in cliques[genre1]:
        print(f"{person1} not in {genre1} clique, "
              "Bundle will not be made nor recommended")
        return

    if person2 not in cliques[genre2]:
        print(f"{person2} not in {genre2} clique, "
              "Bundle will not be made nor recommended")
        return

    # Getting clique based on genre
    recommended_clique = cliques.get(genre1, None)

    # Recommend the games to others in the same clique
    for person in recommended_clique:
        # Ensure the person did not already buy the games
        if person != person1 and person != person2:
            print(f"Recommended {game1} and {game2} as a bundle to {person}")
```

Figure 24. Recommendation system function

From figure 24, the recommendation system function is receiving four extra parameters. The first parameter is the labeled cliques that was made in part D. The second parameter is the game database that tells what genre a game is, an example of this can be seen in figure 15. While, the third and fourth parameters are game purchases. The game purchases are implemented with tuples, with the first element being the user that bought the game, while the second element is the game that is being bought.

The function will bring the conclusion of the bundle recommendation based on the purchases being made. The different cases will be discussed in depth in the next section.

F. Program Structure for Simulation

The file structure of this program would have a structure as follows:

```
|---root
|     |-main.py
|     |-data_processing.py
```

Figure 25. Program file structure

Here, data_processing,py will contain all the functions modules such as create_similar_preference_graph from figure 16, Bron_Kerbosch from figure 19, finalize_cliques from figure 22, and recommendation_system from figure 24.

While main.py will contain the data such as user_preferences from figure 13 and games_genre_mapping from figure 15. In addition to that, it will be the main driver of the program that will use the modular function to integrate it into one program that will simulate the bundle recommendation system.

```python
from data_processing import *

games_genre_mapping = {
    'Puzzle': ['Wormhole 2', 'Bombsweeper', 'Cake Smash Saga'],
    'RPG': ['City of Battlecraft', 'Landrim', 'First Reality'],
    'Casual': ['Animals vs Aliens', 'Furious Fish', 'Cosmicdew Canyon']
}

preferences = {
    'Alice': ['Puzzle', 'Casual'],
    'Brad': ['RPG'],
    'Caleb': ['RPG', 'Casual'],
    'Doris': ['Puzzle'],
    'Easton': ['Puzzle', 'RPG', 'Casual']
}

def main():
    G = create_similar_preference_graph(preferences)
    R, P, X, cliques = [], [], [], []
    for person in G:
        P.append(person)
    bron_kerbosch(R, P, X, G, cliques)
    cliques = finalize_cliques(preferences, cliques)

    person1 = input("\nInput buyer 1: ")
    game1 = input("Input bought game 1: ")
    person2 = input("Input buyer 2: ")
    game2 = input("Input bought game 2: ")

    purchase1 = (person1, game1)
    purchase2 = (person2, game2)

    print(f"\nPurchase 1: {purchase1[0]} bought {purchase1[1]}")
    print(f"Purchase 2: {purchase2[0]} bought {purchase2[1]}\n")

    recommendation_system(cliques, games_genre_mapping, purchase1, purchase2)
    print("")


if __name__ == "__main__":
    main()
```

Figure 26. main.py program

V. TESTING AND ANALYSIS

For testing and analysis, the data that has been used as an example so far will be applied. Please refer to figures 13 and 15 for the complete data set.

As a summary of the main program, user preference data will be transformed into a graph of users connected to each other based on common genres. This graph will then be input into the Bron-Kerbosch algorithm to be split into cliques that correspond to a certain game genre. The output from Bron-Kerbosch will be modified to label each clique with its respective genre. Lastly, inputs will be used to simulate user purchases, and with the clique data and purchases, bundles will be generated and recommended to users.

With the system completed, this section will analyze how the system performs under different scenarios.

A. Case 1: Games that were bought are of different genres

```
Input buyer 1: Caleb
Input bought game 1: Landrim
Input buyer 2: Caleb
Input bought game 2: Furious Fish

Purchase 1: Caleb bought Landrim
Purchase 2: Caleb bought Furious Fish

Landrim and Furious Fish are of different genres. Can not be made into a bundle.
```

Figure 27. Program output for case 1

Caleb is a user that likes RPG and Casual genre. He bought *Landrim* which is an RPG game and Furious Fish which is a Casual game. Since both of these games have different genres, a bundle will not be made and will not be recommended to other users.

B. Case 2: Games that were bought are bought by users of different cliques

```
Input buyer 1: Brad
Input bought game 1: Wormhole 2
Input buyer 2: Brad
Input bought game 2: Bombsweeper

Purchase 1: Brad bought Wormhole 2
Purchase 2: Brad bought Bombsweeper

Brad not in Puzzle clique, Bundle will not be made nor recommended
```

Figure 28. Program output for case 2

Brad is a user that likes RPG genre. He bought *Wormhole 2* and *Bombsweeper* which are both Puzzle games. His purchases did not create a bundle and were not recommended to anyone. The system behaves this way to avoid purchases of users that are new in a genre to be recommended. To make sure the recommendation caters to a genre clique, the system will only take into consideration purchases that were made by users in the genre clique itself. Here, because Brad likes RPG but bought Puzzle games, his purchases were not made into a bundle because he is not considered to be in the Puzzle clique.

C. Case 3: Games are of the same genre and bought by one person in the same clique

```
Input buyer 1: Alice
Input bought game 1: Animals vs Aliens
Input buyer 2: Alice
Input bought game 2: Cosmicdew Canyon

Purchase 1: Alice bought Animals vs Aliens
Purchase 2: Alice bought Cosmicdew Canyon

Recommended Animals vs Aliens and Cosmicdew Canyon as a bundle to Easton
Recommended Animals vs Aliens and Cosmicdew Canyon as a bundle to Caleb
```

Figure 29. Program output for case 3

Alice is a user that likes Puzzle and Casual games. She bought *Animals vs Aliens* and *Cosmicdew Canyon* which are both Casual games. Since the games are in the same genre and Alice belongs in the clique of that genre, her bought games are made into a bundle and offered to other users in the clique. Keep in mind that the Casual genre clique consists of Alice, Caleb, and Easton, with Alice the one buying, the only other users to recommend the game to is Caleb and Easton.

D. Case 4: Games are of the same genre but were bought by different people in the clique

```
Input buyer 1: Easton
Input bought game 1: City of Battlecraft
Input buyer 2: Caleb
Input bought game 2: First Reality

Purchase 1: Easton bought City of Battlecraft
Purchase 2: Caleb bought First Reality

Recommended City of Battlecraft and First Reality as a bundle to Brad
```

Figure 30. Program output for case 4

Easton is player that likes Puzzle, RPG, and Casual games. The game he bought *City of Battlecraft* is an RPG game. Meanwhile, Caleb is a player that likes RPG and Casual games. He bought *First Reality* which is also an RPG game. Since both games are RPG genre and both Easton and Caleb are a part of the RPG clique, the two games will be made into a bundle and recommended to other users in the RPG clique. That clique consists of Brad, Caleb, and Easton, since Caleb and Easton were the ones that purchased the games, the only other user to recommend the bundle to is Brad.

VI. Conclusion

In conclusion, using graph theory, specifically using Bron-Kerbosch algorithm to determine personalized Steam game bundles is definitely possible. From our small implementation and testing, a recommendation system that bases bundling games based on if another user buys a game in the same genre has been made. However, this implementation has many drawbacks that needs to be improved upon should it be applied in real world scenarios. Should there be an opportunity to improve the system, the first change would be to expand user preferences beyond the single dimension of game genres to also include factors such as the game's producer, animation, audio, gameplay, and other elements that may influence user preferences. By taking in other factors, the system would be a lot more accurate. Furthermore, the system should not trigger from only singular purchases and could be improved by taking in a big sample of recent purchases to determine bundles. This is needed because if only the two most recent purchases are considered to create a bundle, in a platform as popular as Steam, the users would be spammed with bundle offers. It would be better, for example, to take all purchases in a day and if a game is bought really often by people in the same clique, then a bundle would be created and recommended. Despite the weaknesses mentioned, Bron-Kerbosch algorithm displays great results for recommendation system, even one that is more complex, like this bundle recommendation system.

VII. Acknowledgment

The completion of this paper could not have been possible without the help of all IF1220 lecturers, especially Ir. Rila Mandala, M.Eng., Ph.D. whom has taught class K02 2024 for discrete mathematics. The author would also like to thank Dr. Ir. Rinaldi Munir, M.T. deeply for providing a great amount of learning resources for students to study from. The author hopes this paper to serve as a useful reference, not only for others interested in the relevant field of study but also as a resource for the author's future work.

References

[1] R. J. Wilson, *Introduction to Graph Theory*, 4th ed. Harlow, Essex, U.K.: Addison Wesley Longman Ltd., 1996, pp. 1–38.

[2] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575-577, 1973.

[3] A. Conte, "Review of the Bron-Kerbosch algorithm and variations," *Level 4 Project*, School of Computing Science, University of Glasgow, 2013.

[4] S. C. Antoro, K. A. Sugeng, and B. D. Handari, "Application of Bron-Kerbosch algorithm in graph clustering using complement matrix," *AIP Conf. Proc.*, vol. 1862, 030141, 2017.

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, accessed at 23 December 2024

[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf, accessed at 23 December 2024

[7] I. J. H. Chan, "Aplikasi Algoritma Bron-Kerbosch dalam Sistem Rekomendasi Konten YouTube," *Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung*, Bandung, Indonesia, 2023.

Statement

Hereby, I declare that this paper I have written is my own work, not a reproduction or translation of someone else's paper, and not plagiarized.

Bandung, 26 Desember 2024

Aryo Wisanggeni 13523100