TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

HOMEWORK 02

Học kỳ 3 – Năm học 2022-2023

MÃ LƯU TRỮ
(do phòng KT-ĐBCL ghi)

Tên học phần:	**Khai thác Dữ liệu Đồ thị**	Mã HP:	**CSC17103**
Thời gian làm bài:	**07 ngày**	Ngày nộp:	**02/07/2023**

HOMEWORK 02: FREQUENT GRAPH PATTERN MINING

Họ tên sinh viên: Võ Văn Hoàng **MSSV**: 20127028

BÀI TRÌNH BÀY

1. Problem 1. Apriori-based Graph Mining (AGM):

- Describe Apriori-based Graph Mining (AGM) algorithms in a way you understand.
  - Apriori-based Graph Mining (AGM) là một lớp thuật toán được sử dụng để khai thác các mẫu hoặc mối quan hệ trong dữ liệu đồ thị. Các thuật toán này dựa trên các nguyên tắc của thuật toán Apriori, thường được sử dụng để khai thác tập phổ biến trong data transaction.
  - Trong thuật toán Apriori-based Graph Mining - AGM, dữ liệu đầu vào được biểu diễn dưới dạng biểu đồ, trong đó các nút biểu thị các thực thể hoặc đối tượng và các cạnh biểu thị mối quan hệ hoặc kết nối giữa các nút.
  - Có 2 bước chính để giải bài toán này: tạo ứng viên và xác thực mẫu.
    - **Tạo ứng viên:**
    - ✓ Bước này liên quan đến việc tạo một tập hợp các mẫu ứng viên
    - ✓ Thuật toán bắt đầu với các mẫu nhỏ và dần dần mở rộng chúng thành các mẫu lớn hơn bằng cách thêm các node hoặc cạnh. Quá trình mở rộng theo nguyên tắc Apriori: một mẫu không thường xuyên, thì tất cả các tập hợp cha của nó cũng phải không thường xuyên.
    - ✓ Quét dữ liệu biểu đồ và đếm số lần xuất hiện của từng mẫu ứng cử viên.

    - **Xác thực mẫu:**
    - ✓ Trong bước này, các mẫu ứng cử viên đã tạo được xác thực để xác định mức ý nghĩa. Các biện pháp khác nhau có thể được sử dụng để đánh giá mức độ thú vị, chẳng hạn như độ tin cậy.
    - ✓ Nếu một candidate đáp ứng được ngưỡng thú vị thì đó là một mẫu hợp lệ và đưa nó vào tập hợp kết quả. Nếu không thì ta sẽ cắt tỉa nó.
  - Cuối cùng, trả về tập hợp các mẫu hợp lệ được phát hiện trong graph.

- Explain the difference between the support of a frequent subgraph and the confidence of a rule in the context of the AGM algorithm.

→

- **Support of a frequent subgraph:**

- Support of a frequent subgraph biểu thị tỷ lệ các graph trong tập dữ liệu có chứa subgraph đã cho. Ngoài ra, nó cho biết tần suất subgraph xuất hiện trong tập dữ liệu và giúp xác định các patterns quan trọng.
- Support of a frequent subgraph được tính bằng cách chia số lượng graph chứa subgraph đó cho tổng số graph trong tập dữ liệu.
- Support of a frequent subgraph thì ghi lại tần suất xuất hiện của nó trong tập dữ liệu nhằm hỗ trợ nhận dạng mẫu.

- **Confidence of a rule in the context of the AGM algorithm:**

- Confidence of a rule thể hiện độ mạnh của mối quan hệ giữa input và output ý nghĩa với sự xuất hiện của chúng trong dữ liệu đồ thị. Nó giúp ta biết được khả năng tìm thấy output ý nghĩa khi có mặt input.
- Confidence of a rule được tính bằng cách chia độ hỗ trợ của các input hợp lệ và output cho độ hỗ trợ của riêng input.
- Confidence of a rule chỉ ra mối quan hệ giữa input và output nhằm đánh giá quy tắc và ra các quyết định.

- Consider the following market transactional database.

Transaction ID	***List of item's in each transaction***
1	Pasta, Fruits, Butter, Vegetables
2	Burgers, French Fries, Ketchup, Mayo
3	Burgers, French Fries, Ketchup
4	Burgers, Pasta, French Fries
5	Vegetables, Fruits
6	Fruits, Orange Juice, Vegetables
7	Burgers, French Fries, Vegetables

Hình 1

- To find frequently occurring patterns in a given database, use the Apriori algorithm with a support threshold of 25%. After obtaining these frequent patterns, list all the association rules that are considered strong, with a confidence threshold of 50%. Prior to implementing the Apriori algorithm, convert the given database into a graph. It is important to note that only providing the final result will not be sufficient, so all the individual steps involved should be presented and explained in detail.

→ Ta có: ngưỡng support là 25% → Số item của ngưỡng support ≥(0.25 . 7 = 1.75) ≈ 2

Ta có số lần xuất hiện của các item trong 7 transaction như sau:

Item	***Count***
Pasta	**2**
Fruits	**3**
Butter	**1 *(loại)***
Vegetables	**4**
Burgers	**4**
French Fries	**4**
Ketchup	**2**
Mayo	**1 *(loại)***
Orange Juice	**1 *(loại)***

Hình 2

Ta có bảng transaction sau khi loại bỏ các item nhỏ hơn ngưỡng MinSup

Transaction ID	***List of item's in each transaction***
1	Pasta, Fruits, Vegetables
2	Burgers, French Fries, Ketchup
3	Burgers, French Fries, Ketchup
4	Burgers, Pasta, French Fries

5	Vegetables, Fruits
6	Fruits, Vegetables
7	Burgers, French Fries, Vegetables

Hình 3

- Bước tiếp theo, ta khởi tạo ma trận 2D được gọi là webaddr với số không. Giả sử đối với ma trận 2D, tọa độ dọc của biểu đồ được đặt tên là tọa độ X và tọa độ ngang được gọi là tọa độ Y. Các tọa độ X và Y của ma trận là tên của các item đủ điều kiện sau bước đầu tiên, tức là: Pasta, Fruits, Vegetables, Burgers, French Fries, Ketchup.

	Pasta	**Fruits**	**Vegetables**	**Burgers**	**French Fries**	**Ketchup**
Pasta	0	0	0	0	0	0
Fruits	0	0	0	0	0	0
Vegetables	0	0	0	0	0	0
Burgers	0	0	0	0	0	0
French Fries	0	0	0	0	0	0
Ketchup	0	0	0	0	0	0

Hình 4: Webaddr - ma trận được khởi tạo với 0's

- Sau đó, một vòng lặp được chạy cho mọi giao dịch trong cơ sở dữ liệu mẫu từ Transaction ID = 1 đến Transaction ID = 7.
- Tiếp theo thì tạo ra sự kết hợp của hai mục cho mọi transaction. Ví dụ, trong Hình 3 với Transaction ID = 3, các mặt hàng là (Burgers, French Fries, Ketchup). Sự kết hợp của hai mục là (Burgers, French Fries), (Burgers, Ketchup), (French Fries, Ketchup). Ta có các tập kết hợp hai mục như sau:

Transaction ID	*Tập kết hợp 2 mục*
1	- (Pasta, Fruits) - (Pasta, Vegetables) - (Fruits, Vegetables)
2	- (Burgers, French Fries) - (Burgers, Ketchup) - (French Fries, Ketchup)
3	- (Burgers, French Fries) - (Burgers, Ketchup) - (French Fries, Ketchup)
4	- (Burgers, Pasta) - (Burgers, French Fries) - (Pasta, French Fries)
5	- (Vegetables, Fruits)
6	- (Fruits, Vegetables)
7	- (Burgers, French Fries) - (Burgers, Vegetables) - (French Fries, Vegetables)

Hình 5. Các tập kết hợp được sinh ra ở các transaction

- Để không làm tăng tập candidate set, ta có thể xem (Vegetables, Fruits) và (Fruits, Vegetables) là một. → Ta có các tập kết hợp sau:
  - (Pasta, Fruits)
  - (Pasta, Vegetables)
  - (Fruits, Vegetables)
  - (Burgers, French Fries)
  - (Burgers, Ketchup)
  - (French Fries, Ketchup)
  - (Vegetables, Fruits)
  - (Burgers, Vegetables)
  - (French Fries, Vegetables)

- Kế tiếp ta kiểm tra xem các kết hợp được tạo ra, ví dụ. [[Burgers, French Fries] (Burgers ở dạng Tọa độ X và French Fries ở dạng Tọa độ Y)] hiện diện trong ma trận webaddr. Nếu có các kết hợp thì tăng giá trị lên 1. Ví dụ: (Burgers, French Fries) có trong webaddr vì vậy hãy tăng giá trị bằng 1.

	Pasta	**Fruits**	**Vegetables**	**Burgers**	**French Fries**	**Ketchup**
Pasta	0	1	1	1	1	0
Fruits	0	0	3	0	0	0
Vegetables	0	0	0	1	1	0
Burgers	0	0	0	0	4	2
French Fries	0	0	0	0	0	2
Ketchup	0	0	0	0	0	0

Hình 6. Ma trận webaddr

- Các giá trị trong bảng này sẽ là trọng số của 1 đồ thị trong các bước sau.
- Sau bước này ta sẽ Biến danh sách 2D toàn cầu "Trả lời" được khai báo để tất cả các mẫu phổ biến được phát hiện ở dạng danh sách được lưu trữ trong biến này một biến danh sách khác "result_list" được khai báo.
    - Đỉnh mới đucợ tìm thấy không được có trong danh sách result_list.
    - Trọng số Cạnh từ phần tử cuối cùng trong result_list trong đến đỉnh mới được tìm thấy phải ≥ Giá trị hỗ trợ ở hình 2.
    - Ít nhất 1 đỉnh mới phải được phát hiện đáp ứng hai tiêu chí trên tại mỗi lần gọi hàm AprioriGraph. Nếu không quay trở lại.
- Ban đầu, biến danh sách "result_list" được khai báo rỗng

Null	Null	Null	Null	Null

- Tại thời điểm gọi hàm AprioriGraph, biến loại danh sách "result_list" có "**Burgers**" được nối thêm vào nó.

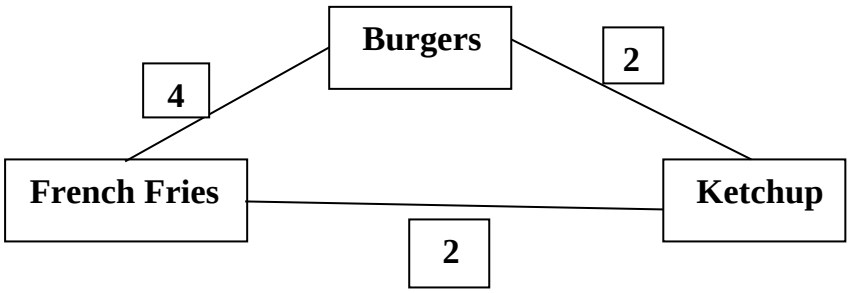
Burgers	Null	Null	Null	Null

- Sau đó, ta thấy Burger và French Fries có trọng số là 4 (thỏa minSup) nên:

Burgers	**French Fries**	Null	Null	Null

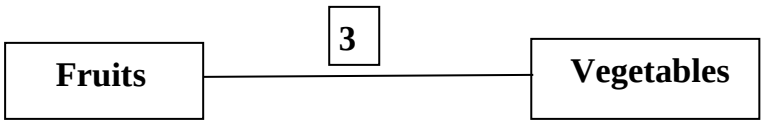
- Cứ lần lượt như thế ta sẽ có được:

Burgers	**French Fries**	**Ketchup**	Null	Null



- Delete result list vì không tìm thấy candidate nào thỏa MinSup nữa, ta được danh sách sau:

Fruits	**Vegetables**	Null	Null	Null



Vậy: Ta có các frequent patterns sau:

- (French Fries, Burgers)
- (Burgers, Ketchup)
- (French Fries, Ketchup)
- (French Fries, Burgers, Ketchup)
- (Vegetables, Fruits)

- Để thử lại, ta sẽ tìm association rules từ các itemset như sau:
- Confidence(Ketchup → Burgers, French Fries) = $\frac{\text{Support(Burgers , French Fries, Ketchup)}}{\text{Support(Ketchup)}} = 1$

- Confidence(Burgers, Ketchup → French Fries) = $\frac{\text{Support(Burgers, French Fries, Ketchup)}}{\text{Support(Burgers, Ketchup)}} = 1$
- Confidence(Vegetables → Fruits) = $\frac{\text{Support(Fruits, Vegetables)}}{\text{Support(Vegetables)}} = 1$
- Confidence(Fruits → Vegetables) = $\frac{\text{Support(Fruits, Vegetables)}}{\text{Support(Fruits)}} = \frac{3}{4}$
- Confidence(Burgers, French Fries → Ketchup) = $\frac{\text{Support(Burgers, French Fries, Ketchup)}}{\text{Support(Burgers, French Fries)}} = \frac{2}{4} = \frac{1}{2}$
- Confidence(French Fries → Burgers, Ketchup) = $\frac{\text{Support(Burgers, French Fries, Ketchup)}}{\text{Support(French Fries)}} = \frac{2}{4} = \frac{1}{2}$
- Confidence(Burgers → French Fries, Ketchup) = $\frac{\text{Support(Burgers, French Fries, Ketchup)}}{\text{Support(Burgers)}} = \frac{2}{4} = \frac{1}{2}$

- Thông qua các phép tính toán trên ta có các luật kết hợp như sau:

- Burgers → French Fries, Ketchup
- Burgers, French Fries → Ketchup
- Burgers, Ketchup → French Fries
- French Fries, Ketchup → Burgers
- French Fries → Burgers, Ketchup
- Vegetables → Fruits
- Fruits →Vegetables
- Ketchup → Burgers, French Fries

➔ Đây điều là các strong association rule vì confidence đều đạt ngưỡng từ 50% trở lên.

2. Problem 2. Graph-based Substructure Pattern Mining (gSpan):

- Describe gSpan algorithms in a way you understand.
    - Thuật toán Graph-based Substructure Pattern Mining (gSpan) tạo các subgraph chung từ phía dưới lên trên.
    - Thuật toán này sử dụng DFS code làm biểu diễn chính tắc của subgraph. Các subgraph trùng lặp được loại bỏ bằng cách chọn mã DFS tối thiểu theo thứ tự từ điển.
    - Kỹ thuật như vậy sẽ giúp gSpan trích xuất hiệu quả các subgraph thường xuyên từ bộ dữ liệu đồ thị lớn.

- What is the comparison made by gSpan algorithm while testing for isomorphism between two graphs and what is the reason for this comparison ?
  - Việc so sánh này là để xác định xem hai biểu đồ có cấu trúc giống nhau nhưng có nhãn nút và cạnh khác nhau hay không (đẳng cấu).
  - Lí do cho việc đó là khi sử dụng các mã chính tắc cho các subgraph đảm bảo rằng phép kiểm tra đẳng cấu là chính xác, vì nó loại bỏ nhu cầu tìm kiếm và so sánh tất cả các subgraph có thể. Ngoài ra, việc sử dụng cùng một thứ tự DFS cho cả hai biểu đồ đảm bảo rằng sự so sánh là tương đương nhất
  - Ngoài ra, mã DFS là một chuỗi nên khi so sánh giữa 2 mã DFS tối thiểu tương ứng với 2 đồ thị sẽ dễ dàng hơn so với việc so sánh giữa 2 đồ thị.

- Which two main expenditures of Apriori were targeted by gSpan for reduction ?

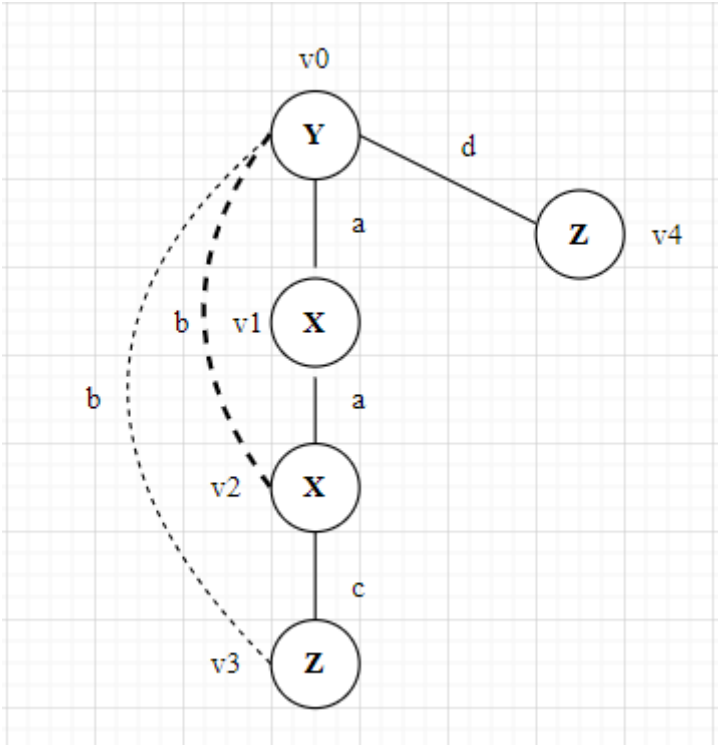
→ Hai khoản chi tiêu chính của Apriori đã được gSpan nhắm mục tiêu để giảm là:

- **Tạo candidate:** Thuật toán Apriori tạo ra một số lượng lớn tập candidate, nhiều trong số đó không phổ biến và phải được cắt tỉa. Quá trình này có thể tốn kém về mặt tính toán và thời gian.
- **Đếm Frequent itemset:** Thuật toán Apriori sẽ phải quét toàn bộ database nhiều lần để đếm độ hỗ trợ của từng tập phổ biến. Đây có thể là một quá trình tốn thời gian khi database càng lớn.

- Draw a DFS tree corresponding to the following DFS code.

Edge	***Code***
0	(0, 1, Y, a, X)
1	(1, 2, X, a, X)
2	(2, 0, X, b, Y)
3	(2, 3, X, c, Z)
4	(3, 0, Z, b, Y)
5	(0, 4, Y, d, Z)

- Ta có DFS tree như sau:



3. Tài liệu tham khảo:

[1]: Modified Apriori Graph Algorithm for Frequent Pattern Mining

[2]: gSpan: Graph-Based Substructure Pattern Mining

Hết