`TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO GIỮA KỲ MÔN: PHÂN TÍCH DỮ LIỆU THIẾT KẾ GIẢI THUẬT**

**NGHIÊN CỨU GIẢI THUẬT CỦA BÀI BÁO KHAI THÁC CÁC BỘ PHẬN CÓ TRỌNG SỐ THƯỜNG XUYÊN XẾP HẠNG ĐẦU BẰNG CÁC SỬ DỤNG CẤU TRÚC DANH SÁCH WN VÀ CHIẾN LƯỢC CẮT TỈA SỚM**

*Người hướng dẫn*: **TS NGUYỄN CHÍ THIỆN**

*Người thực hiện*: **LÊ HỮU LỢI– 52100819**

**KHƯƠNG TRUNG TUẤN – 52100862**

**NGUYỄN THỌ THẮNG-52100839**

Lớp **: 21050261**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO GIỮA KỲ MÔN: PHÂN TÍCH DỮ LIỆU THIẾT KẾ GIẢI THUẬT**

**NGHIÊN CỨU GIẢI THUẬT CỦA BÀI BÁO KHAI THÁC CÁC BỘ PHẬN CÓ TRỌNG SỐ THƯỜNG XUYÊN XẾP HẠNG ĐẦU BẰNG CÁC SỬ DỤNG CẤU TRÚC DANH SÁCH WN VÀ CHIẾN LƯỢC CẮT TỈA SỚM**

*Người hướng dẫn*: **TS NGUYỄN CHÍ THIỆN**

*Người thực hiện*: **LÊ HỮU LỢI– 52100819**

**KHƯƠNG TRUNG TUẤN – 52100862**

**NGUYỄN THỌ THẮNG-52100839**

Lớp **: 21050261**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Chúng em xin cảm ơn thầy trong quá trình làm bài báo cáo, nhờ sự đánh giá tận tâm và giúp đỡ tụi em trong việc lựa chọn đề tài. Nhờ đó chúng em mới có thể hoàn thành bài báo cáo.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Nguyễn Văn A;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 20 tháng11 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Lê Hữu Lợi*

*Nguyễn Thọ Thắng*

*Khương Trung Tuấn*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trước hết chúng em xin tóm tắt lại nội dung của bài báo cáo. Bài báo cáo có tiêu đề "Mining top-rank-k frequent weighted itemsets using WN-list structures and an early pruning strategy" được viết bởi Bay Vo, Huong Bui, Thanh Vo, và Tuong Le. Trong bài báo, tác giả giới thiệu vấn đề khai thác top-rank-k frequent weighted itemsets (FWIs) từ cơ sở dữ liệu có trọng số. Các FWIs là một biến thể của frequent itemsets (FIs) và có khả năng xem xét sự quan trọng hay trọng số khác nhau cho mỗi mục trong các itemsets.

Các FWIs có thể được khai thác từ cơ sở dữ liệu bằng nhiều thuật toán khác nhau, nhưng các thuật toán truyền thống thường tạo ra một số lượng lớn các FWIs, gây khó khăn trong việc áp dụng với các hệ thống thông minh. Vì vậy, trong bài báo này, tác giả đề xuất giải pháp khai thác top-rank-k FWIs từ cơ sở dữ liệu có trọng số bằng cách kết hợp giai đoạn khai thác và xếp hạng thành một giai đoạn duy nhất mà không cần tìm tất cả các FWIs, nhằm tăng tính ứng dụng của chúng trong các ứng dụng thực tế.

Bài báo cũng trình bày 3 thuật toán cơ sở để khai thác top-rank-k FWIs, bao gồm TFWIT, TFWID, TFWIN và một giải thuật tối ưu TFWINPlus sử dụng các cấu trúc dữ liệu tiên tiến như tidset, diffset và WN-list structures. Ngoài ra, bài báo cũng đề xuất các chiến lược nâng cao hiệu suất khai thác top-rank-k FWIs, bao gồm chiến lược tăng ngưỡng và chiến lược cắt tỉa sớm, được hỗ trợ bởi một định lý mới.

Cuối cùng, bài báo trình bày kết quả thực nghiệm về thời gian xử lý và sử dụng bộ nhớ giữa các thuật toán đã đề xuất để chứng minh tính hiệu quả của thuật toán TFWIN+. Kết quả thực nghiệm cho thấy TFWIN+ vượt trội hơn so với các thuật toán TFWIT, TFWID và TFWIN trong việc khai thác top-rank-k FWIs.

MỤC LỤC

[LỜI CẢM ƠN ii](#_Toc387692905)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN ii](#_Toc387692906)

[TÓM TẮT ii](#_Toc387692907)

[MỤC LỤC 1](#_Toc387692908)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 2](#_Toc387692909)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 3](#_Toc387692909)

[CHƯƠNG 1 – GIỚI THIỆU 2](#_Toc387692910)

[1.1 Giới thiệu các thuật toán 2](#_Toc387692911)

[1.2 Nội dung của bài báo cáo 2](#_Toc387692916)

[CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT/NGHIÊN CỨU THỰC NGHIỆM VÀ TÀI LIỆU THAM KHẢO 2](#_Toc387692917)

[1 Thuật toán TFWIT 2](#_Toc387692918)

[1.1 Xác định input và output 2](#_Toc387692919)

[1.2 Giải thuật 2](#_Toc387692919)

[1.3 Giải thích thuật toán 2](#_Toc387692919)

[2 Thuật toán TFWID 2](#_Toc387692918)

[1.1 Xác định input và output 2](#_Toc387692919)

[1.2 Giải thuật 2](#_Toc387692919)

[1.3 Giải thích thuật toán 2](#_Toc387692919)

[3 Thuật toán TFWIN 2](#_Toc387692918)

[1.1 Xác định input và output 2](#_Toc387692919)

[1.2 Giải thuật 2](#_Toc387692919)

[1.3 Giải thích thuật toán 2](#_Toc387692919)

[4 Thuật toán TFWIN+ 2](#_Toc387692918)

[1.1 Xác định input và output 2](#_Toc387692919)

[1.2 Giải thuật 2](#_Toc387692919)

[1.3 Giải thích thuật toán 2](#_Toc387692919)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC CHỮ VIẾT TẮT**

FWIs:

TFWIT:  
TFWID:  
TFWIN:  
TFWIN+:

CHƯƠNG 1 – Giới thiệu

* 1. Giới thiệu về các thuật toán

Thuật toán TFWIT:

Thuật toán TFWIT bắt đầu bằng việc xây dựng một danh sách ban đầu chứa tất cả các item có trọng số lớn hơn hoặc bằng ngưỡng trọng số đã cho. Sau đó, thuật toán sử dụng một phương pháp tạo ra các tập ứng viên kế tiếp dựa trên các tập hợp tidset đã được xây dựng trước đó.

Tiếp theo, thuật toán TFWIT áp dụng một quy tắc cắt tỉa để loại bỏ các tập ứng viên không thỏa mãn một số điều kiện nhất định, giúp giảm bớt không gian tìm kiếm. Sau đó, thuật toán duyệt qua các tập ứng viên còn lại để tính toán trọng số hỗ trợ cho từng tập ứng viên. Cuối cùng, thuật toán trả về tập các top-rank-k FWIs dựa trên trọng số hỗ trợ đã tính toán được.

Thuật toán TFWIT được đánh giá về hiệu suất khai thác thông qua thực nghiệm về thời gian xử lý và sử dụng bộ nhớ. Kết quả thực nghiệm cho thấy TFWIT có hiệu suất tốt trong việc khai thác top-rank-k FWIs từ cơ sở dữ liệu có trọng số.

Thuật toán TFWID:

Thuật toán TFWID bắt đầu bằng việc xây dựng một danh sách ban đầu chứa tất cả các item có trọng số lớn hơn hoặc bằng một ngưỡng đã cho. Sau đó, nó sử dụng một phương pháp tạo ra các tập ứng viên kế tiếp dựa trên các tập hợp diffset đã được xây dựng trước đó.

TFWID áp dụng một quy tắc cắt tỉa để loại bỏ các tập ứng viên không thỏa mãn một số điều kiện, giúp giảm không gian tìm kiếm. Thuật toán duyệt qua các tập ứng viên còn lại để tính toán trọng số hỗ trợ cho từng tập ứng viên. Cuối cùng, nó trả về tập các top-rank-k FWIs dựa trên trọng số hỗ trợ đã tính toán được.

TFWID đã được đánh giá thông qua thực nghiệm về thời gian xử lý và sử dụng bộ nhớ. Kết quả thực nghiệm cho thấy thuật toán này có hiệu suất tốt trong việc khai thác top-rank-k FWIs từ cơ sở dữ liệu có trọng số.

Thuật toán TFWIN:

Thuật toán TFWIN được thiết kế để khai thác các tập hợp mục (itemset) phổ biến có trọng số từ cơ sở dữ liệu. Mục tiêu của thuật toán là tìm ra các tập hợp mục có trọng số lớn nhất hoặc top-k tập hợp mục có trọng số cao nhất.

Thuật toán TFWIN bắt đầu bằng việc xây dựng một danh sách các tập hợp mục ban đầu bằng cách duyệt qua dữ liệu và tính toán trọng số của từng mục. Sau đó, thuật toán sử dụng một phương pháp tạo ra các tập ứng viên kế tiếp dựa trên các tập hợp mục đã được xây dựng trước đó.

TFWIN áp dụng các quy tắc cắt tỉa để loại bỏ các tập ứng viên không thỏa mãn một số điều kiện, giúp giảm không gian tìm kiếm. Tiếp theo, thuật toán duyệt qua các tập ứng viên còn lại để tính toán trọng số hỗ trợ cho từng tập hợp mục. Cuối cùng, nó trả về tập các tập hợp mục phổ biến có trọng số lớn nhất hoặc top-k tập hợp mục có trọng số cao nhất dựa trên trọng số hỗ trợ đã tính toán được.

Thuật toán TFWIN+:

Thuật toán TFWIN+ là một phiên bản cải tiến của thuật toán TFWIN, được thiết kế để khai thác các tập hợp mục phổ biến có trọng số từ cơ sở dữ liệu. Mục tiêu của thuật toán là tìm ra các tập hợp mục có trọng số lớn nhất hoặc top-k tập hợp mục có trọng số cao nhất.

TFWIN+ sử dụng một số kỹ thuật cắt tỉa để giảm không gian tìm kiếm và tăng hiệu suất khai thác. Cụ thể, thuật toán áp dụng các phương pháp như pruning dựa trên upper bound (giới hạn trên) và pruning dựa trên lower bound (giới hạn dưới) để loại bỏ các tập hợp mục không có khả năng trở thành kết quả top-k hoặc không thỏa mãn các điều kiện cần thiết.

TFWIN+ cũng sử dụng một cấu trúc dữ liệu được tối ưu hóa để lưu trữ và truy vấn dữ liệu, giúp tăng tốc quá trình xử lý. Bằng cách kết hợp các kỹ thuật cắt tỉa và cấu trúc dữ liệu tối ưu, TFWIN+ đạt được hiệu suất cao trong việc khai thác các tập hợp mục phổ biến có trọng số từ cơ sở dữ liệu.

TFWIN+ đã được đánh giá và thử nghiệm trên các tập dữ liệu thực tế và đã cho thấy khả năng khai thác hiệu quả và hiệu suất tốt trong việc tìm kiếm top-k tập hợp mục có trọng số cao nhất từ dữ liệu có trọng số.

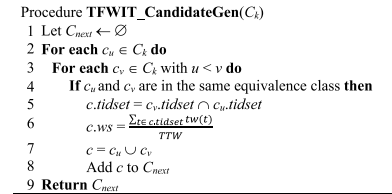
1.2 Nội dung của bài báo cáo

Chúng em tìm hiểu các giải thuật trên, từ mã giả đã cho , chúng em đã viết lại bằng ngôn ngữ Python 3, với mục tiêu là hiểu được giải thuật.

CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT / NGHIÊN CỨU THỰC NGHIỆM

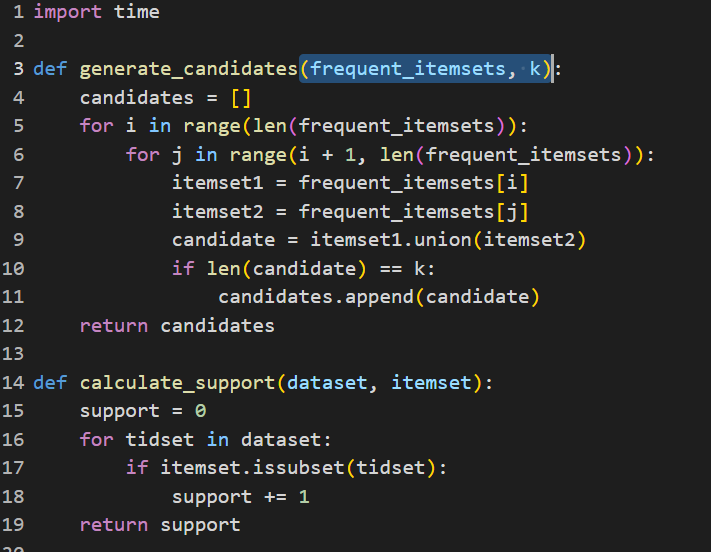
**TÀI LIỆU THAM KHẢO**

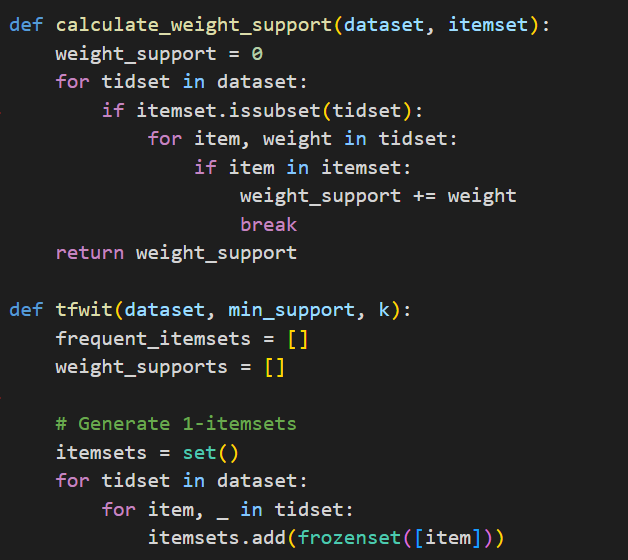
1. **Thuật toán TFWIT:**

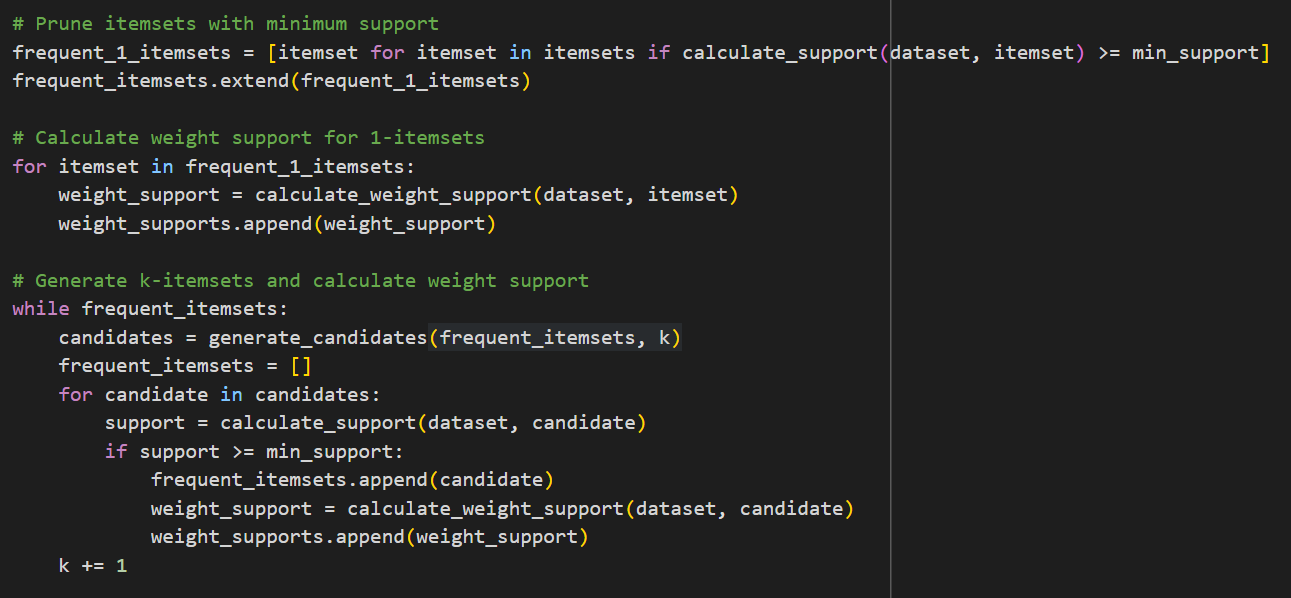


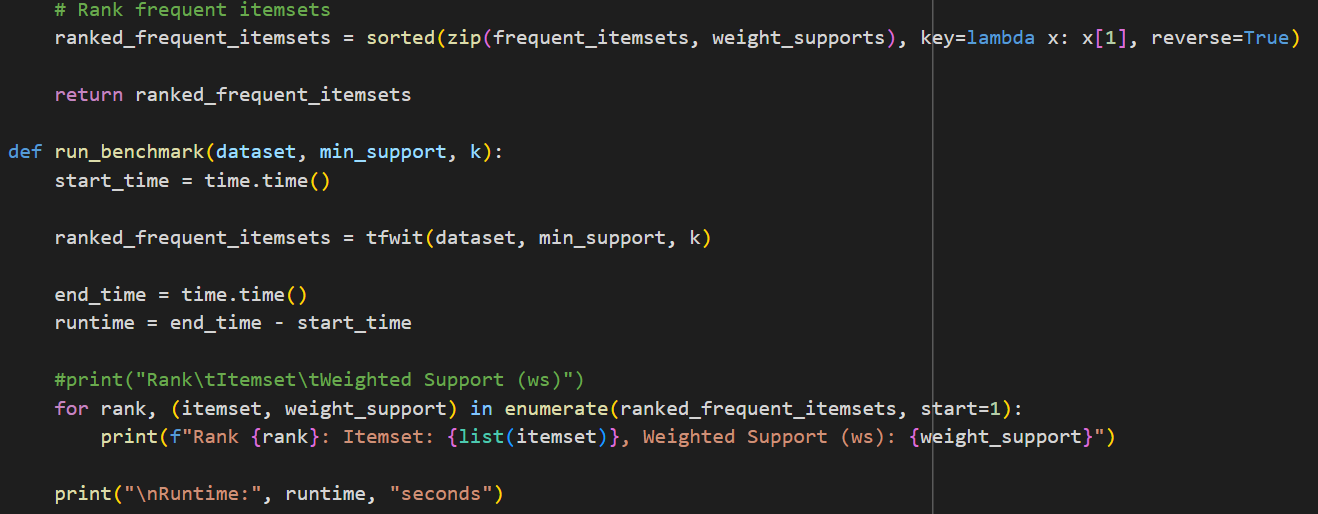
**1.1 Xác định input và output của từng phương thức trong lớp `TFWIT`:**

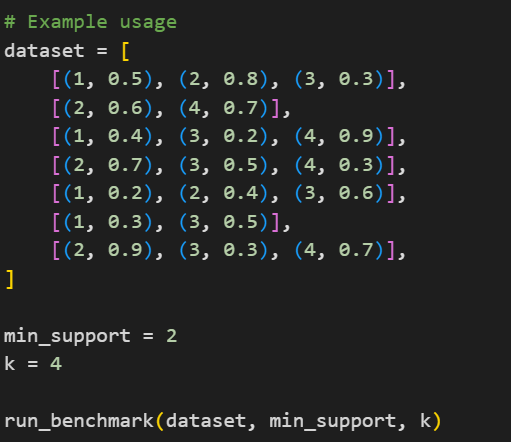
* **Phương thức generate\_candidates(frequent\_itemsets, k):**
* Đầu vào:
* frequent\_itemsets: Danh sách các tập hợp mục phổ biến k-itemsets trước đó.
* k: Kích thước mong muốn của các tập hợp ứng viên k-itemsets.
* Đầu ra: Danh sách các tập hợp ứng viên k-itemsets.
* **Phương thức calculate\_support(dataset, itemset):**
* Đầu vào:
* dataset: Tập dữ liệu gồm danh sách các tập giao dịch.
* itemset: Tập hợp mục cần tính toán hỗ trợ.
* Đầu ra: Số lần xuất hiện của tập hợp mục trong tập dữ liệu (hỗ trợ).
* **Phương thức calculate\_weight\_support(dataset, itemset):**
* Đầu vào:
* dataset: Tập dữ liệu gồm danh sách các tập giao dịch.
* itemset: Tập hợp mục cần tính toán trọng số hỗ trợ.
* Đầu ra: Trọng số hỗ trợ của tập hợp mục trong tập dữ liệu.
* **Phương thức tfwit(dataset, min\_support, k):**
* Đầu vào:
* dataset: Tập dữ liệu gồm danh sách các tập giao dịch.
* min\_support: Ngưỡng hỗ trợ tối thiểu.
* k: Kích thước ban đầu của các tập hợp mục.
* Đầu ra: Danh sách các cặp (tập hợp mục, trọng số hỗ trợ) được sắp xếp theo trọng số hỗ trợ giảm dần.
* **Phương thức run\_benchmark(dataset, min\_support, k):**
* Đầu vào:
* dataset: Tập dữ liệu gồm danh sách các tập giao dịch.
* min\_support: Ngưỡng hỗ trợ tối thiểu.
* k: Kích thước ban đầu của các tập hợp mục.
* Đầu ra: không có giá trị trả về (void). Nó chỉ in ra danh sách các tập hợp mục phổ biến và thời gian thực thi của thuật toán.
  1. **Thực hiện giải thuật TFWIT:**

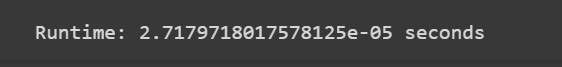
****

****

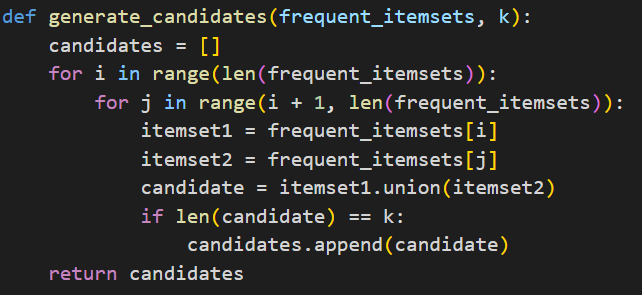
****

****

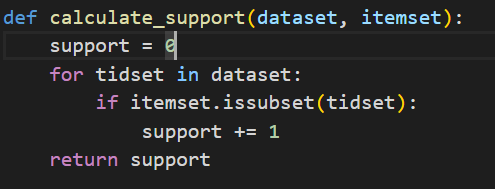
****

****

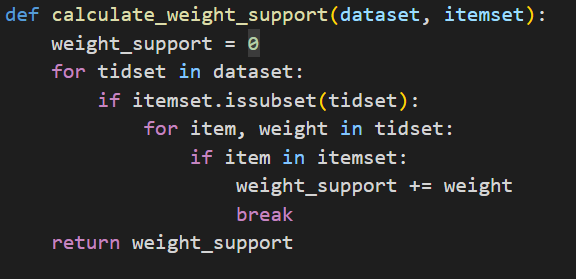
* 1. **Giải thích**

****

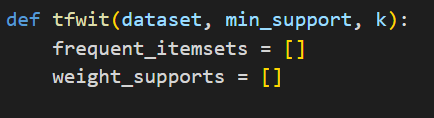
* Hàm generate\_candidates nhận vào danh sách các tập hợp mục phổ biến frequent\_itemsets và giá trị k (kích thước mong muốn của các tập hợp ứng viên). Hàm này tạo ra các tập hợp ứng viên bằng cách kết hợp các cặp tập hợp mục phổ biến từ danh sách đầu vào. Nó kiểm tra độ dài của mỗi tập hợp ứng viên và thêm nó vào danh sách candidates nếu độ dài bằng k. Hàm trả về danh sách các tập hợp ứng viên (candidates).



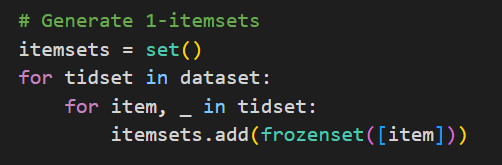
* Hàm calculate\_support nhận vào một tập dữ liệu (danh sách các tập giao dịch) và một tập hợp mục. Nó tính toán số lần xuất hiện của tập hợp mục đó trong tập dữ liệu, tức là số lượng tập giao dịch trong tập dữ liệu chứa tập hợp mục đó. Hàm lặp qua mỗi tập giao dịch trong tập dữ liệu và kiểm tra xem tập hợp mục có là tập con của tập giao dịch không. Nếu phải, nó tăng giá trị support lên. Hàm trả về giá trị support.



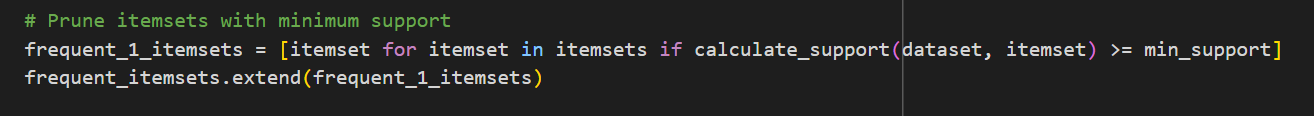
* Hàm calculate\_weight\_support nhận vào một tập dữ liệu và một tập hợp mục. Nó tính toán trọng số hỗ trợ của tập hợp mục đó, tức là tổng trọng số của các giao dịch chứa tập hợp mục đó. Hàm lặp qua mỗi tập giao dịch trong tập dữ liệu và kiểm tra xem tập hợp mục có là tập con của tập giao dịch không. Nếu phải, nó lặp qua các mục và trọng số trong tập giao dịch và thêm trọng số vào weight\_support nếu mục đó thuộc tập hợp mục. Hàm trả về trọng số hỗ trợ.



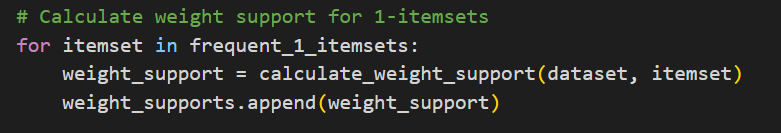
* Hàm chính tfwit thực hiện thuật toán TFWIT. Nó nhận vào một tập dữ liệu, ngưỡng hỗ trợ tối thiểu min\_support và kích thước ban đầu của các tập hợp mục k.



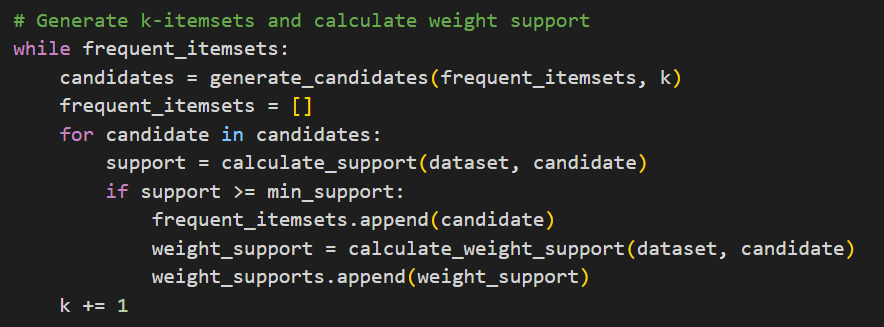
* Hàm bắt đầu bằng cách tạo ra các tập hợp mục 1-itemsets từ tập dữ liệu. Nó lặp qua mỗi tập giao dịch trong tập dữ liệu và thêm từng mục vào tập hợp mục itemsets.



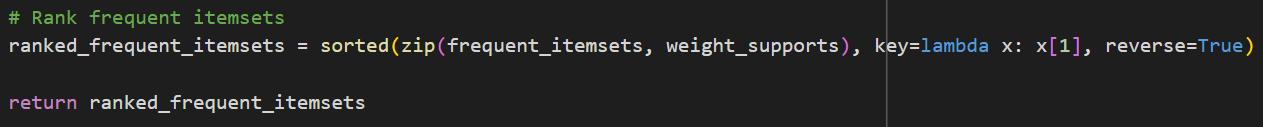
* Tiếp theo, nó loại bỏ các tập hợp mục 1-itemsets không đạt ngưỡng hỗ trợ tối thiểu bằng cách tính toán số lần xuất hiện của chúng bằng cách sử dụng hàm calculate\_support và chỉ giữ lại các tập hợp mục có số lần xuất hiện lớn hơn hoặc bằng ngưỡng min\_support. Các tập hợp mục phổ biến này được thêm vào danh sách frequent\_itemsets.



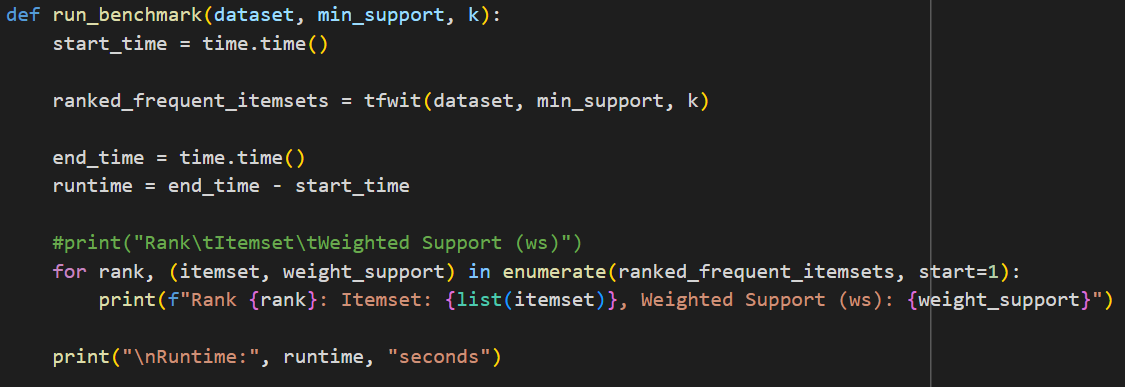
* Sau đó, nó tính toán trọng số hỗ trợ cho mỗi tập hợp mục phổ biến 1-itemsets bằng cách sử dụng hàm calculate\_weight\_support và lưu giá trị trọng số hỗ trợ vào danh sách weight\_supports.



* Mã chạy vào một vòng lặp while để tạo ra các tập hợp mục k-itemsets và tính toán trọng số hỗ trợ cho chúng. Trong vòng lặp này, nó sử dụng hàm generate\_candidates để tạo ra các tập hợp ứng viên từ các tập hợp mục phổ biến hiện tại. Sau đó, nó lặp qua từng ứng viên, tính toán hỗ trợ sử dụng hàm calculate\_support và chỉ giữ lại các tập hợp mục có hỗ trợ lớn hơn hoặc bằng ngưỡng min\_support. Các tập hợp mục phổ biến này được thêm vào danh sách frequent\_itemsets, và trọng số hỗ trợ được tính toán bằng cách sử dụng hàm calculate\_weight\_support và được thêm vào danh sách weight\_supports. Kích thước k được tăng lên sau mỗi vòng lặp để tạo ra các tập hợp mục k-itemsets.



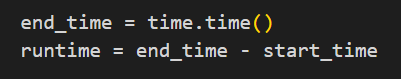
* Cuối cùng, các tập hợp mục phổ biến được sắp xếp theo trọng số hỗ trợ giảm dần và được trả về dưới dạng danh sách các cặp (tập hợp mục, trọng số hỗ trợ). Điều này được thực hiện bằng cách sử dụng hàm sorted với tham số key=lambda x: x[1] để sắp xếp theo trọng số hỗ trợ và reverse=True để đảo ngược thứ tự.



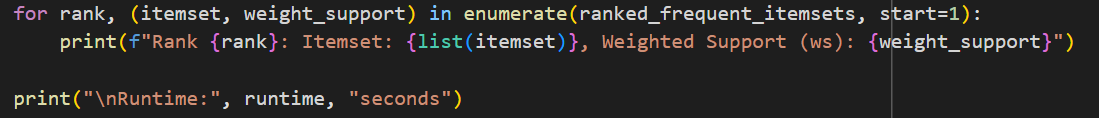
* Hàm run\_benchmark được thêm vào để đo thời gian thực thi của thuật toán tfwit. Nó bắt đầu bằng cách ghi lại thời gian hiện tại bằng cách sử dụng time.time() để đánh dấu thời điểm bắt đầu thực thi.



* Tiếp theo, nó gọi hàm tfwit với dữ liệu, ngưỡng hỗ trợ tối thiểu và kích thước ban đầu để nhận danh sách các tập hợp mục phổ biến.

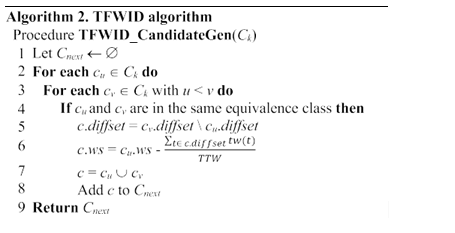


* Sau khi nhận được danh sách các tập hợp mục phổ biến, nó tính thời gian thực thi bằng cách trừ thời điểm bắt đầu từ thời gian hiện tại.



* Cuối cùng, nó lặp qua các tập hợp mục phổ biến đã được sắp xếp và in ra thứ hạng, tập hợp mục và giá trị trọng số hỗ trợ.

1. **Thuật toán TFWID**



* 1. **Xác định input và output của từng phương thức trong lớp TFWID**

**Class TFWID:**

* **Phương thức \_\_init\_\_(self, min\_support):**

1. Đầu vào: min\_support là một số thực (float) đại diện cho ngưỡng hỗ trợ tối thiểu của itemset (có thể là None nếu không có ngưỡng).
2. Đầu ra: Không có.

* **Phương thức check\_same\_equivalence(self, c\_i, c\_j):**

1. Đầu vào: c\_i và c\_j là hai đối tượng của lớp TFWID cần so sánh.
2. Đầu ra: Trả về True nếu c\_i và c\_j có cùng tính tương đương, ngược lại trả về False.

* **Phương thức diffset\_combination(self, a, b, hash\_tw\_of\_trans, sum\_tw):**

1. Đầu vào: a và b là hai từ điển (dict) biểu diễn các itemset. hash\_tw\_of\_trans là một từ điển biểu diễn bảng băm của trọng số giao dịch. sum\_tw là một số thực (float) đại diện cho tổng trọng số của các itemset.
2. Đầu ra: Trả về một từ điển biểu diễn sự kết hợp (diffset combination) của a và b, cùng với sum\_tw đã được cập nhật.

* **Phương thức add\_transaction(self, transaction):**

1. Đầu vào: transaction là một danh sách các cặp (item, weight) đại diện cho một giao dịch. item là một đối tượng (item) và weight là trọng số tương ứng của item trong giao dịch.
2. Đầu ra: Không có.

* **Phương thức generate\_itemsets(self):**

1. Đầu vào: Không có.
2. Đầu ra: Trả về một danh sách chứa các itemset phổ biến (frequent itemsets) cùng với hỗ trợ tương ứng, đã được sắp xếp theo hỗ trợ giảm dần.

* **Phương thức get\_top\_k\_itemsets(self, k):**

1. Đầu vào: k là một số nguyên (int) đại diện cho số lượng itemset hàng đầu cần trả về.
2. Đầu ra: Trả về danh sách chứa k itemset phổ biến hàng đầu, đã được sắp xếp theo hỗ trợ giảm dần.

**Hàm tfwid\_cadidate\_generation:**

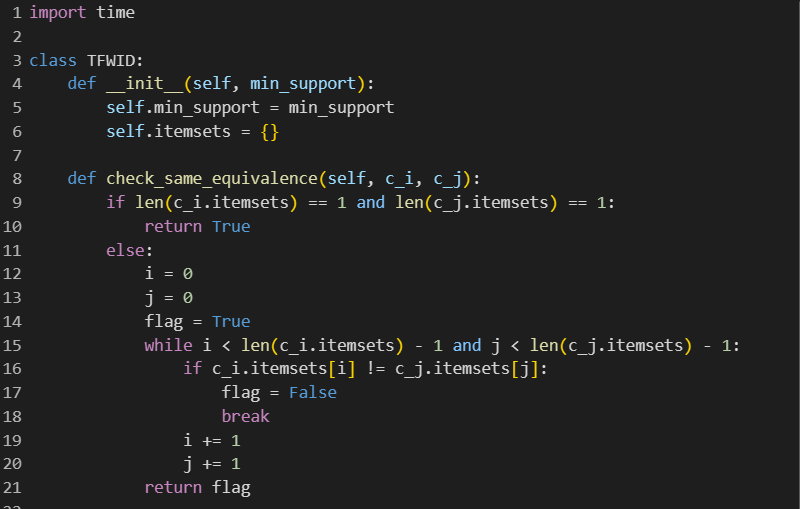
Đầu vào:

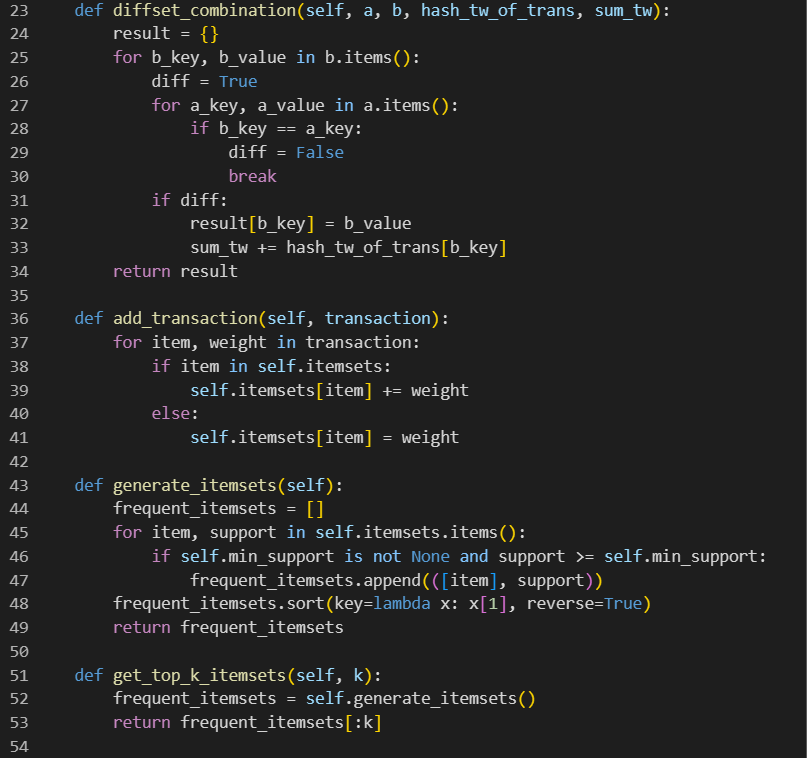
1. candidate\_k là một danh sách (List) các đối tượng của lớp TFWID. Mỗi đối tượng đại diện cho một ứng cử viên itemset.
2. hash\_tw\_of\_trans là một từ điển (Dict) biểu diễn bảng băm của trọng số giao dịch. Khóa (key) là một mục (item), và giá trị (value) là trọng số tương ứng của mục trong các giao dịch.
3. ttw là một số thực (float) đại diện cho tổng trọng số của tất cả các giao dịch.

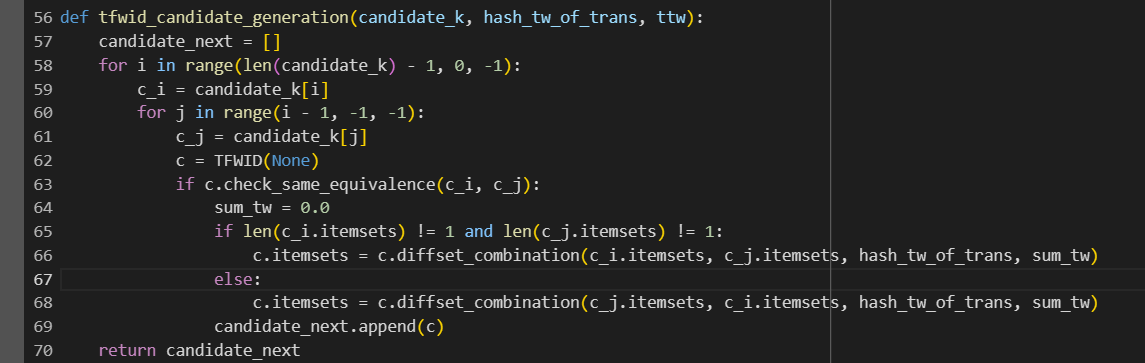
Đầu ra:

1. candidate\_next là một danh sách (List) các đối tượng của lớp TFWID. Đối tượng này đại diện cho các ứng cử viên itemset tiếp theo được tạo ra từ candidate\_k theo quy tắc của thuật toán.

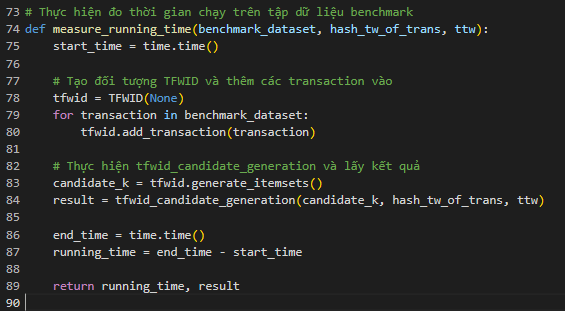
**2.2 Thực hiện giải thuật TFWID:**

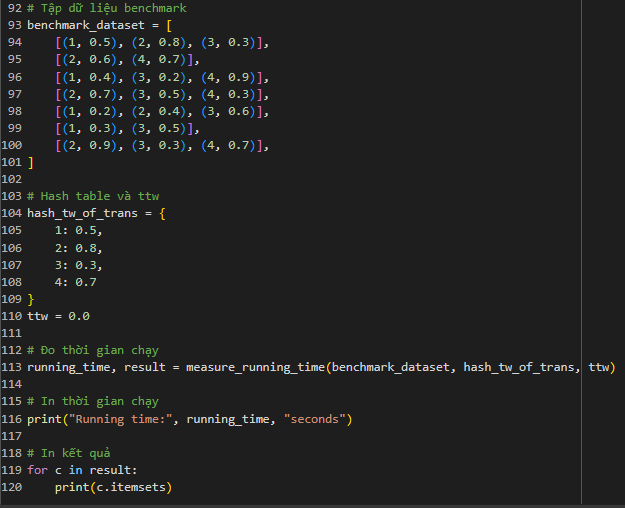






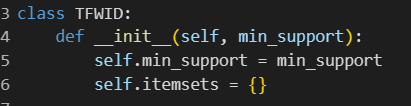
4. Đo thời gian chạy cho bộ dữ liệu điểm chuẩn:



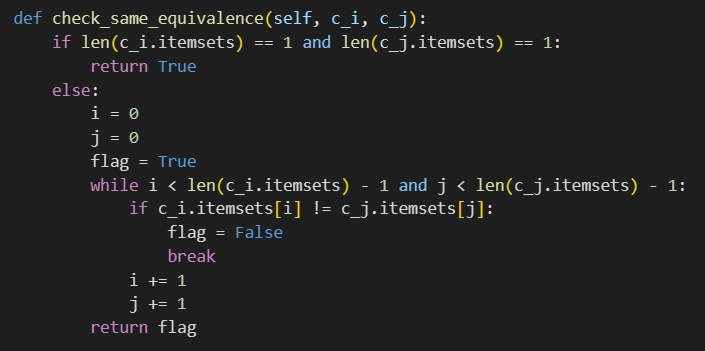


**2.3.3 Giải thích**

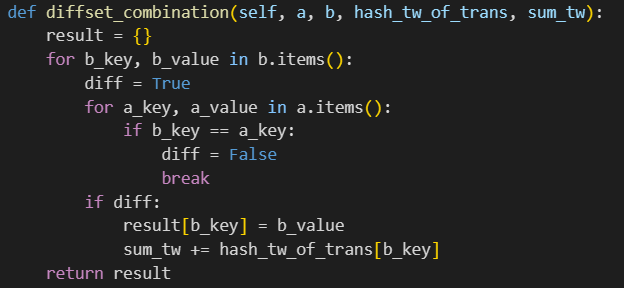
1. Giải thuật Class TFWID và hàm tfwid\_cadidate\_generation



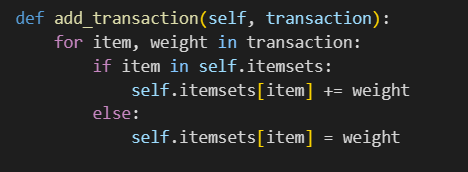
* \_\_init\_\_(self, min\_support): Phương thức khởi tạo của lớp TFWID. Nhận một tham số min\_support để đặt giá trị ngưỡng hỗ trợ tối thiểu cho itemset. self.min\_support lưu trữ giá trị này.
* self.itemsets là một từ điển để lưu trữ các itemset và trọng số tương ứng.



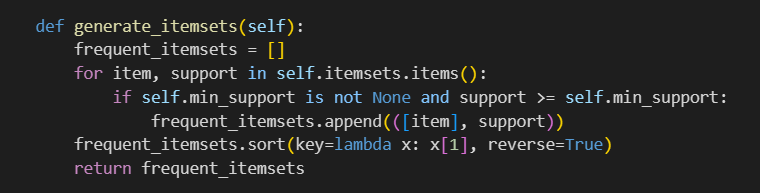
* check\_same\_equivalence(self, c\_i, c\_j): Phương thức kiểm tra tính tương đương giữa hai đối tượng c\_i và c\_j của lớp TFWID. Nếu cả c\_i và c\_j chỉ chứa một itemset, phương thức trả về True.
* Trong trường hợp khác, phương thức so sánh từng itemset của c\_i và c\_j để kiểm tra tính tương đương. Phương thức trả về True nếu các itemset trong c\_i và c\_j tương đương, ngược lại trả về False.



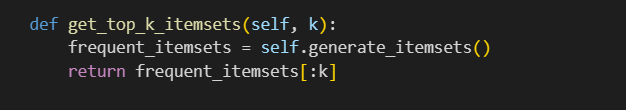
* diffset\_combination(self, a, b, hash\_tw\_of\_trans, sum\_tw): Phương thức thực hiện kết hợp (diffset combination) giữa hai itemset a và b trong lớp TFWID. Phương thức sử dụng các từ điển a và b để so sánh các itemset và tạo ra một từ điển mới result chứa các phần tử chỉ có trong b mà không có trong a. Trong quá trình kết hợp, phương thức cập nhật giá trị sum\_tw bằng tổng trọng số của các itemset trong hash\_tw\_of\_trans.



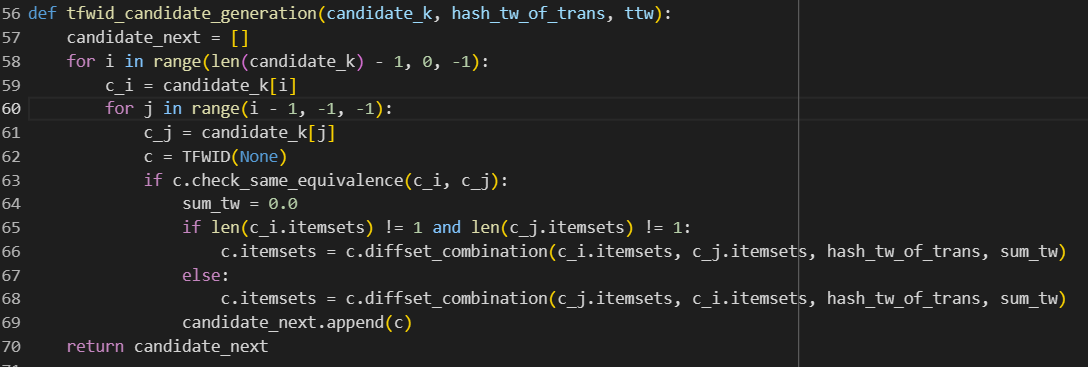
* add\_transaction(self, transaction): Phương thức thêm một giao dịch mới vào lớp TFWID. Giao dịch được biểu diễn dưới dạng danh sách các cặp (item, weight), trong đó item là một đối tượng (item) và weight là trọng số tương ứng của item trong giao dịch. Phương thức cập nhật giá trị trọng số của item trong self.itemsets. Nếu item đã tồn tại trong self.itemsets, trọng số của nó được cộng thêm vào, ngược lại, item mới sẽ được thêm vào self.itemsets với trọng số tương ứng.



* generate\_itemsets(self): Phương thức tạo ra các itemset phổ biến từ self.itemsets. Phương thức duyệt qua từng cặp (item, support) trong self.itemsets và kiểm tra xem trọng số support có lớn hơn hoặc bằng self.min\_support hay không. Nếu có, itemset đó được thêm vào danh sách frequent\_itemsets dưới dạng một cặp ([item], support). Cuối cùng, danh sách frequent\_itemsets được sắp xếp theo trật tự giảm dần của trọng số (support) và trả về.

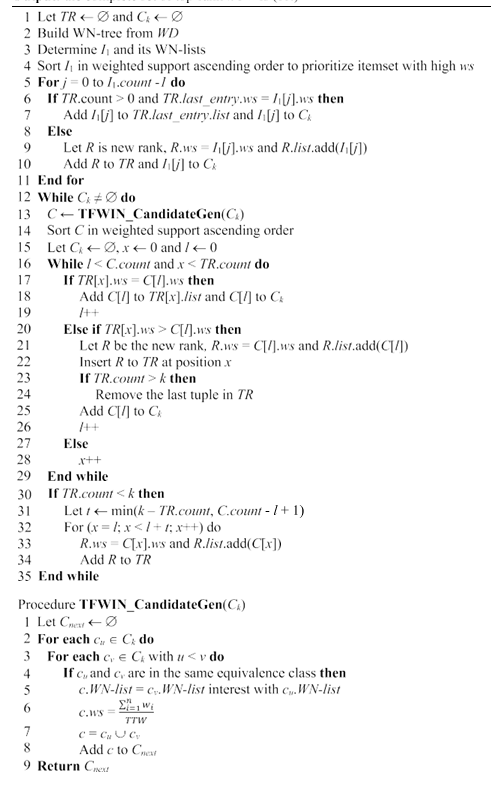


* get\_top\_k\_itemsets(self, k): Phương thức trả về k itemset phổ biến nhất từ self.itemsets. Phương thức sử dụng phương thức generate\_itemsets để tạo ra danh sách frequent\_itemsets chứa tất cả các itemset phổ biến, sau đó trả về k itemset đầu tiên của danh sách frequent\_itemsets.



1. tfwid\_candidate\_generation(candidate\_k, hash\_tw\_of\_trans, ttw): Hàm tạo ra các ứng viên itemset tiếp theo từ danh sách các ứng viên candidate\_k dựa trên từ điển hash\_tw\_of\_trans và tổng trọng số giao dịch ttw. Hàm trả về danh sách các ứng viên itemset tiếp theo.
2. candidate\_next = []: Khởi tạo danh sách rỗng để lưu trữ các ứng viên itemset tiếp theo.
3. for i in range(len(candidate\_k) - 1, 0, -1): Vòng lặp chạy từ len(candidate\_k) - 1 đến 1 (bước -1). Vòng lặp này lặp qua các chỉ số i của candidate\_k từ cuối danh sách đến đầu danh sách.
4. c\_i = candidate\_k[i]: Gán giá trị của ứng viên itemset tại chỉ số i trong candidate\_k cho biến c\_i.
5. for j in range(i - 1, -1, -1): Vòng lặp chạy từ i - 1 đến 0 (bước -1). Vòng lặp này lặp qua các chỉ số j của candidate\_k từ chỉ số i - 1 đến đầu danh sách.
6. c\_j = candidate\_k[j]: Gán giá trị của ứng viên itemset tại chỉ số j trong candidate\_k cho biến c\_j.
7. c = TFWID(None): Khởi tạo một đối tượng c từ lớp TFWID với min\_support được đặt là None.
8. if c.check\_same\_equivalence(c\_i, c\_j): Kiểm tra tính tương đương giữa c\_i và c\_j bằng cách gọi phương thức check\_same\_equivalence của đối tượng c. Nếu chúng tương đương, tiếp tục thực hiện các bước tiếp theo.
9. sum\_tw = 0.0: Khởi tạo biến sum\_tw với giá trị 0.0.
10. if len(c\_i.itemsets) != 1 and len(c\_j.itemsets) != 1: Kiểm tra nếu cả c\_i và c\_j không chỉ chứa một itemset (có độ dài lớn hơn 1), thực hiện các bước trong khối này.
11. c.itemsets = c.diffset\_combination(c\_i.itemsets, c\_j.itemsets, hash\_tw\_of\_trans, sum\_tw): Gán giá trị của kết hợp diffset giữa c\_i.itemsets và c\_j.itemsets cho c.itemsets bằng cách gọi phương thức diffset\_combination của đối tượng c.
12. else: Nếu cả c\_i và c\_j chỉ chứa một itemset (độ dài bằng 1), thực hiện các bước trong khối này.
13. c.itemsets = c.diffset\_combination(c\_j.itemsets, c\_ti.itemsets, hash\_tw\_of\_trans, sum\_tw): Gán giá trị của kết hợp diffset giữa c\_j.itemsets và c\_i.itemsets cho c.itemsets bằng cách gọi phương thức diffset\_combination của đối tượng c.
14. candidate\_next.append(c): Thêm đối tượng c vào danh sách candidate\_next.
15. return candidate\_next: Trả về danh sách các ứng viên itemset tiếp theo.

**3. Thuật toán TFWIN**



**3.1 Xác định input và output của từng phương thức trong lớp TFWID**

* **def create\_item\_wnlists(items, weights, transactions):**

Đầu vào:

- items: Danh sách các item duy nhất trong tập dữ liệu.

- weights: Từ điển với trọng số tương ứng cho mỗi item.

- transactions: Danh sách các giao dịch, mỗi giao dịch là một danh sách các item.

Đầu ra:

- wnlists: Từ điển chứa WN-list cho mỗi item.

* **def intersect\_wnlists(wnlist1, wnlist2):**

Đầu vào:

- wnlist1, wnlist2: Các WN-list cần tìm giao.

Đầu ra:

- intersect: WN-list là kết quả của phép giao giữa wnlist1 và wnlist2.

* **def calc\_ws(wnlist, total\_weight):**

Đầu vào:

- wnlist: WN-list của một pattern.

- total\_weight: Tổng trọng số của tất cả các item trong tập dữ liệu.

Đầu ra:

- ws: Trọng số chuẩn hóa (weighted support) của pattern.

* **def tfwin(transactions, weights, k):**

Đầu vào:

- transactions: Danh sách các giao dịch, mỗi giao dịch là một danh sách các item.

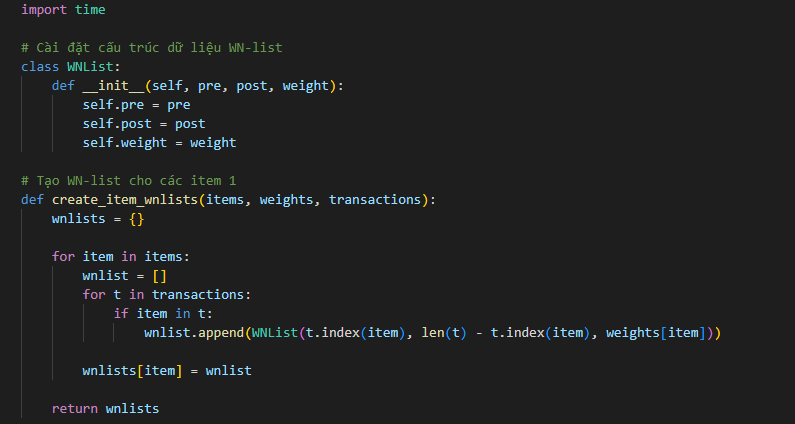
- weights: Từ điển với trọng số tương ứng cho mỗi item.

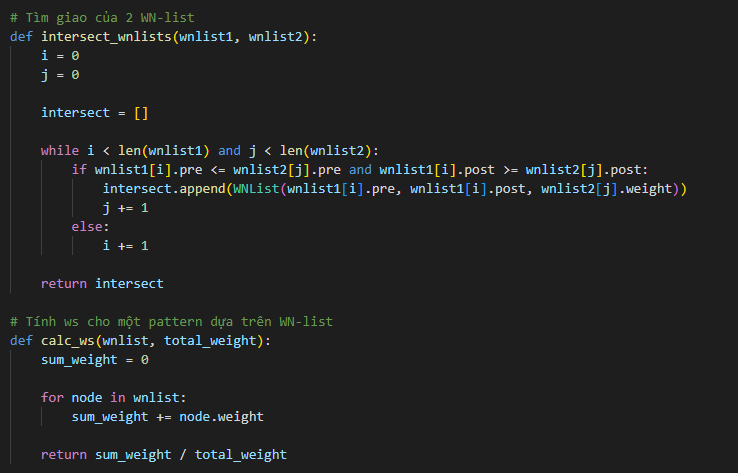
- k: Số lượng itemset phổ biến cần trích xuất.

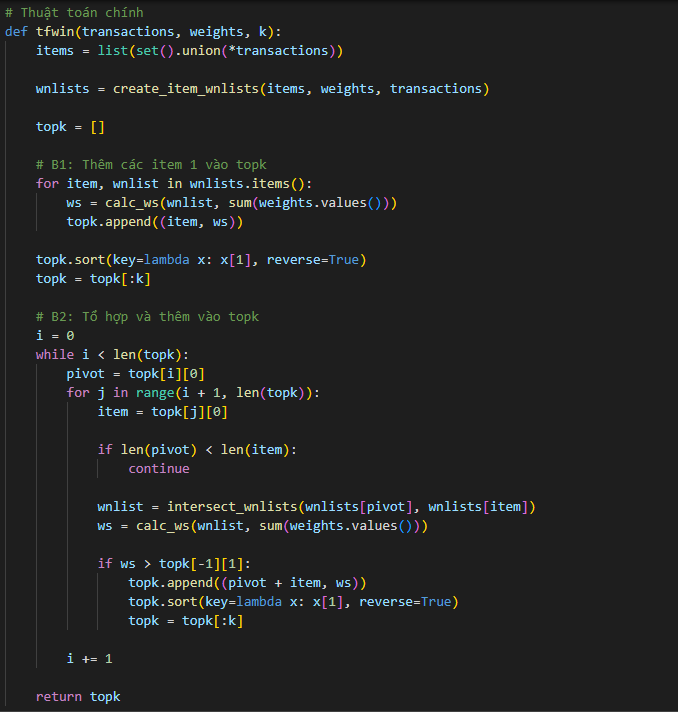
Đầu ra:

- topk: Danh sách chứa các itemset phổ biến hàng đầu cùng với trọng số của chúng.

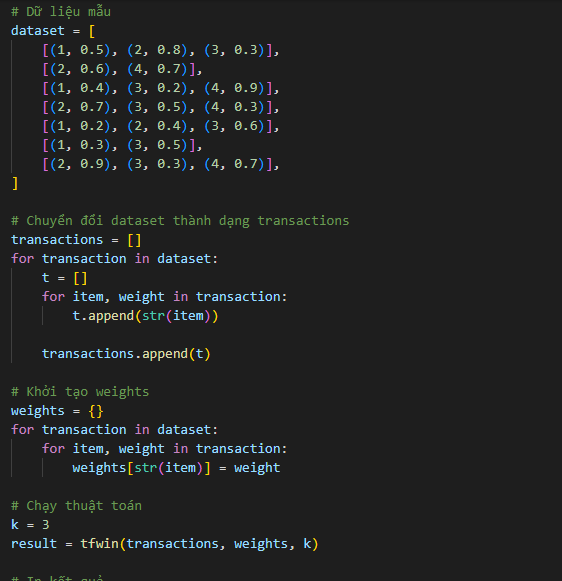
**3.2 Thực hiện giải thuật TFWID**

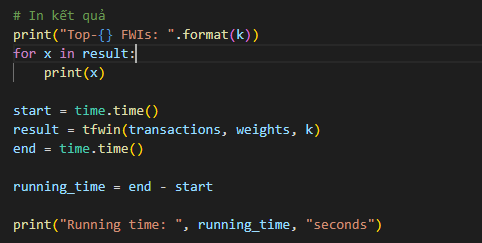


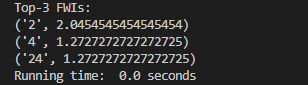




**Đo thời gian chạy cho bộ dữ liệu điểm chuẩn:**







**3.3 Giải thích**

* **create\_item\_wnlists(items, weights, transactions):** Hàm này tạo danh sách WN-list cho mỗi phần tử trong tập dữ liệu. Nó duyệt qua mỗi giao dịch và tạo danh sách WN-list cho mỗi phần tử trong giao dịch dựa trên vị trí và trọng số của nó.
* **intersect\_wnlists(wnlist1, wnlist2):** Hàm này tính toán sự giao nhau của hai danh sách WN-list. Nó duyệt qua các phần tử của danh sách WN-list và tạo một danh sách mới chứa các phần tử chỉ xuất hiện cả trong danh sách thứ nhất và danh sách thứ hai.
* **calc\_ws(wnlist, total\_weight):** Hàm này tính toán trọng số của mẫu dựa trên danh sách WN-list. Nó tính tổng trọng số của tất cả các nút trong danh sách WN và chia cho tổng trọng số của tất cả các phần tử.
* **tfwin(transactions, weights, k):** Hàm chính thực hiện thuật toán TFWIN trên tập dữ liệu đầu vào. Nó bao gồm việc tạo danh sách WN-list cho các mục, tính toán trọng số và tìm ra các mẫu phổ biến nhất dựa trên thuật toán TFWIN.

Dưới đây là một giải thích cụ thể hơn về cách thuật toán TFWIN hoạt động, theo từng bước:

**Chuẩn bị dữ liệu:**

Dữ liệu đầu vào là dataset, một tập dữ liệu gồm nhiều giao dịch (transactions).

Mỗi giao dịch chứa các mục (items) được gắn kèm với trọng số (weights). Dữ liệu này được chuyển đổi thành biểu diễn transactions để thuận tiện xử lý.

**Tạo danh sách trọng số (WNList) cho từng mục:**

Đối với mỗi mục, tạo một danh sách trọng số (WNList) ghi lại vị trí xuất hiện đầu tiên (pre), vị trí cuối cùng (post) và trọng số tương ứng trong mỗi giao dịch.

**Tính toán trọng số hỗ trợ (WS) cho mỗi mục đơn (item 1):**

Tính tổng trọng số của mỗi mục đơn dựa trên thông tin từ các danh sách trọng số (WNList) tương ứng.

**Sắp xếp và chọn Top-k mục đơn với WS cao nhất:**

Sắp xếp các mục đơn theo thứ tự giảm dần của trọng số hỗ trợ (WS).

Chọn ra top-k mục đơn có WS cao nhất.

**Tạo và mở rộng các mẫu:**

Lấy từng mục trong top-k và kết hợp với nhau để tạo ra các mẫu mới có độ dài lớn hơn.

Đảm bảo tính hợp lý của mẫu bằng cách tìm giao giữa các danh sách trọng số (WNList) của các mục.

**Lặp lại quá trình cho đến khi đạt được kết quả Top-k cuối cùng hoặc không còn mẫu nào có thể mở rộng được nữa.**

Lặp lại quá trình của bước 5 để tìm kiếm mẫu có trọng số cao nhất cho đến khi đạt được số lượng top-k hoặc không còn mẫu nào có thể mở rộng được.

**Kết quả:**

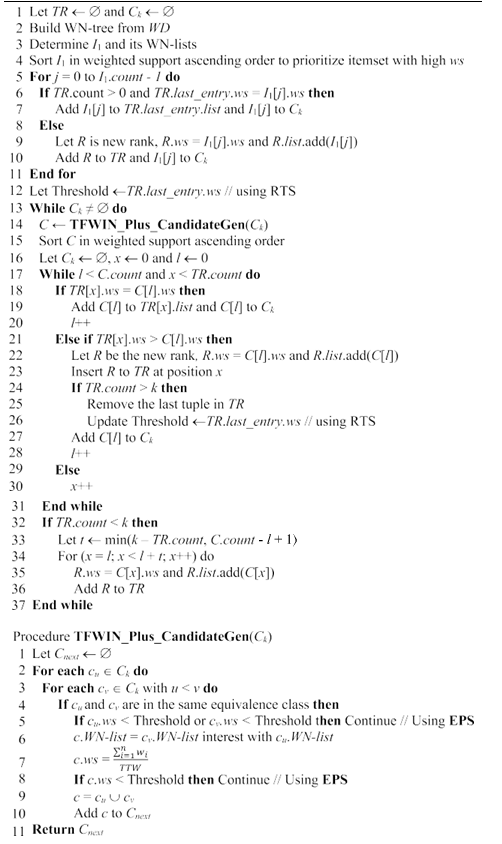
Xuất ra top-k mẫu có trọng số hỗ trợ cao nhất trong dữ liệu.

**Hiển thị kết quả và thời gian chạy:**

In ra top-k mẫu cùng với trọng số hỗ trợ tương ứng.

Đo lường thời gian chạy của thuật toán để đánh giá hiệu suất.

**4. Thuật toán TFWINPLUS**



**4.1 Xác định input và output của từng phương thức trong lớp TFWINPLUS**

* **class Nodecode:**

1. Đầu vào:

* post\_order: Một danh sách đại diện cho việc duyệt cây theo thứ tự sau.
* pre\_order: Một danh sách đại diện cho việc duyệt cây theo thứ tự trước.
* tw: Trọng số được liên kết với nút.

1. Đầu ra: Một thể hiện của lớp NodeCode với các thuộc tính được cung cấp.

* **Class TR:**

1. Đầu vào : không có
2. Đầu ra: Một thể hiện của lớp TR với các thuộc tính sau:

* fwi\_list: Một danh sách để lưu trữ các đối tượng FWI (Frequent Weight Itemset).
* ws: Trọng số được liên kết với đối tượng TR.
* **Class FWI:**

1. Đầu vào : không có
2. Đầu ra: Một thể hiện của lớp FWI với các thuộc tính sau:

* n\_cs: Một danh sách để lưu trữ các kết hợp mã nút.
* ws: Trọng số được liên kết với đối tượng FWI.
* **tfwinplus\_candidate\_generation(candidate\_k, threshold, ttw):**

1. Đầu vào:

* candidate\_k: Một danh sách các đối tượng FWI đại diện cho các frequent weight itemset ứng cử viên hiện tại.
* threshold: Một giá trị ngưỡng cho trọng số.
* ttw: Tổng trọng số giao dịch.

1. Đầu ra: Một danh sách các đối tượng FWI đại diện cho các frequent weight itemset ứng cử viên tiếp theo.

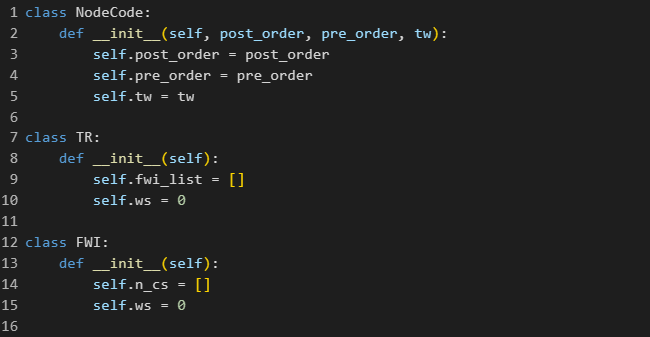
* **tfwinplus(benchmark\_dataset, threshold, ttw):**

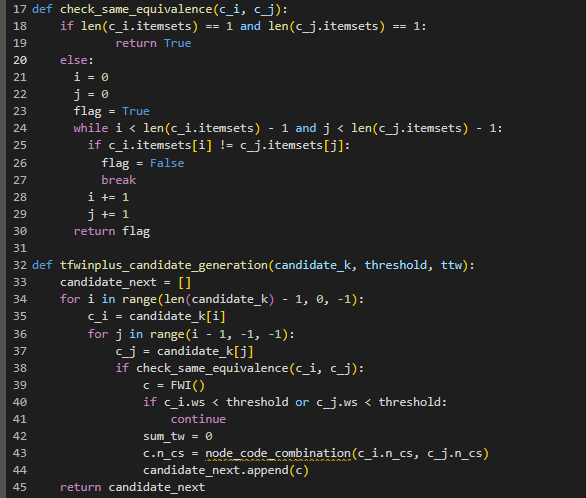
1. Đầu vào:

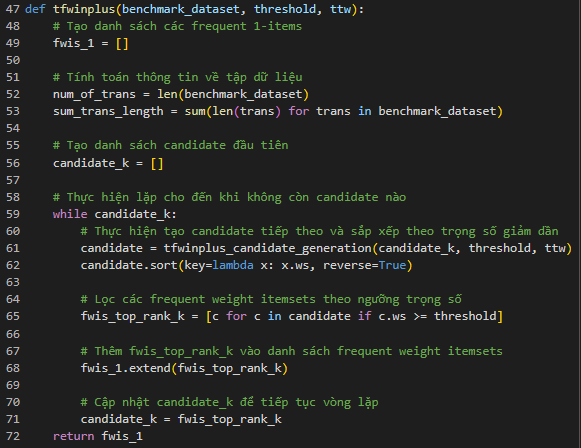
* benchmark\_dataset: Một tập dữ liệu được sử dụng để khai thác frequent weight itemset.
* threshold: Một giá trị ngưỡng cho trọng số.
* ttw: Tổng trọng số giao dịch.

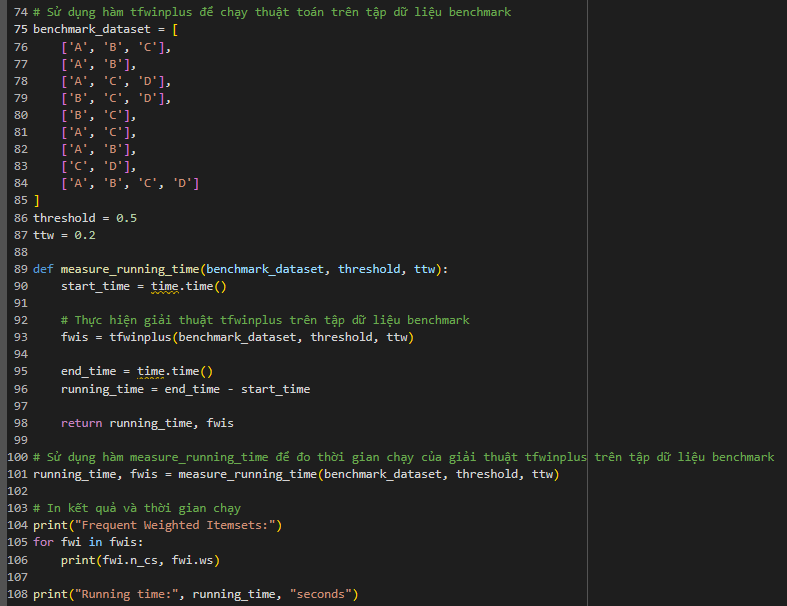
1. Đầu ra: Một danh sách các đối tượng FWI đại diện cho các frequent weight itemset.

**4.2 Thực hiện giải thuật TFWINPLUS**



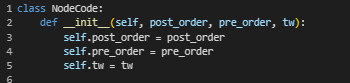






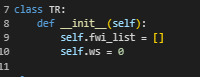
**4.3 Giải thích**

1. lớp NodeCode:



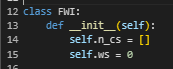
class NodeCode: Lớp này đại diện cho một đối tượng NodeCode. Nó có các thuộc tính post\_order, pre\_order và tw để lưu thông tin về thứ tự sau, thứ tự trước và trọng số của nút.

2. lớp TR:



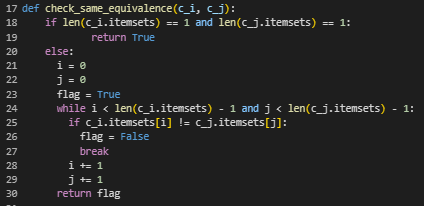
class TR: Lớp này đại diện cho một đối tượng TR. Nó có các thuộc tính fwi\_list và ws để lưu danh sách các FWI và tổng trọng số.

3. lớp FWI:



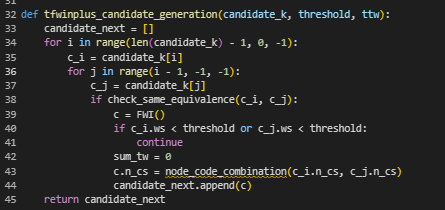
Lớp này đại diện cho một đối tượng FWI. Nó có thuộc tính n\_cs để lưu danh sách các NodeCode và ws để lưu trọng số.

4. hàm check\_same\_equivalence(c\_i, c\_j):



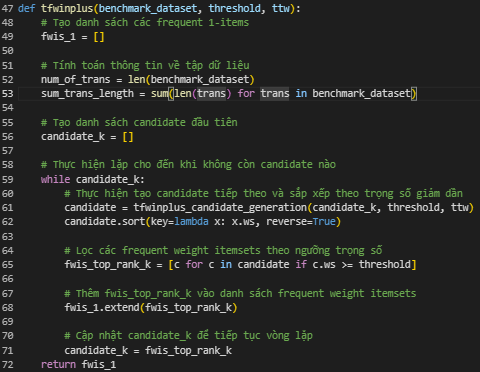
Hàm này được sử dụng để kiểm tra tính đồng đẳng giữa hai FWI c\_i và c\_j. Nếu cả hai chỉ có một itemset và itemset đó giống nhau, hàm trả về True. Trong trường hợp khác, hàm so sánh từng itemset của c\_i và c\_j để kiểm tra tính đồng đẳng.

5. tfwinplus\_candidate\_generation(candidate\_k, threshold, ttw):



Hàm này thực hiện sinh ra các candidate mới từ danh sách candidate\_k theo thuật toán TFWINPlus. Nó lặp qua danh sách candidate\_k và thực hiện kiểm tra tính đồng đẳng giữa các cặp FWI. Nếu cả hai FWI đều có trọng số lớn hơn ngưỡng, sẽ tạo ra một FWI mới với danh sách NodeCode kết hợp của hai FWI và thêm vào danh sách candidate\_next.

6. tfwinplus(benchmark\_dataset, threshold, ttw):



Hàm này là hàm chính thực hiện thuật toán TFWINPlus trên tập dữ liệu benchmark\_dataset với ngưỡng threshold và tổng trọng số ttw. Đầu tiên, nó tạo danh sách fwis\_1 để lưu các frequent weight itemsets. Sau đó, nó tính toán thông tin về tập dữ liệu như tổng số giao dịch và tổng độ dài giao dịch. Tiếp theo, nó tạo danh sách candidate\_k ban đầu và thực hiện vòng lặp để sinh ra các frequent weight itemsets. Cuối cùng, nó trả về danh sách fwis\_1 kết quả.

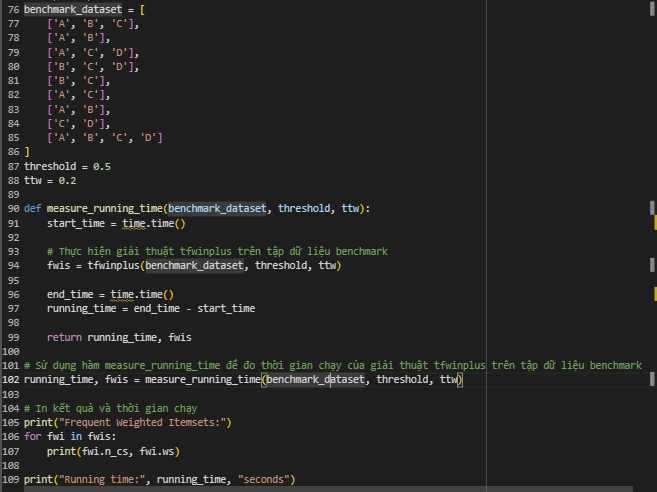
Dữ liệu đầu vào:

* benchmark\_dataset: Một danh sách các giao dịch, trong đó mỗi giao dịch được biểu diễn dưới dạng một danh sách các mục.
* threshold: Ngưỡng trọng số tối thiểu để coi một tập hợp mục là phổ biến.
* ttw: Tổng trọng số giao dịch.

Kết quả đầu ra:

* fwis\_1: Một danh sách các tập hợp mục có trọng số phổ biến, trong đó mỗi tập hợp mục được biểu diễn dưới dạng một đối tượng FWI.

Ví dụ:



**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

**Tiếng Anh**

**1. Bài báo** [Mining top-rank-k frequent weighted itemsets using WN-list structures and an early pruning strategy Bay Vo a , Huong Bui b,c , Thanh Vo d , Tuong Le e,f,∗](https://sci-hub.se/10.1016/j.knosys.2020.106064)

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn