# SUMMARY LOGRAM

1.1. Mở đầu

Logram sử dụng từ điển n-gram để lưu trữ tần suất xuất hiện của các n-gram trong log và tận dụng các từ điển để trích xuất mẫu tĩnh và biến động trong log. Các n-gram xuất hiện thường xuyên có nhiều khả năng đại diện cho mẫu tĩnh, trong khi các n-gram hiếm có nhiều khả năng là các biến động.

Các từ điển n-gram được xây dựng và truy vấn một cách hiệu quả, với độ phức tạp lần lượt là O(n) và O(1).

+ Xây dựng:

+ Truy vấn:

1.2. Đánh giá hiệu quả

+ Logram đạt độ chính xác cao hơn so với các phương pháp tốt nhất hiện có, đồng thời vượt trội hơn về hiệu suất, với tốc độ phân tích nhanh hơn từ 1,8 đến 5,1 lần so với phương pháp nhanh thứ hai.

+ Các từ điển n-gram được xây dựng song song và tổng hợp hiệu quả. Logram có khả năng mở rộng cao khi triển khai trong môi trường đa lõi (ví dụ: cụm Spark) mà không làm giảm độ chính xác của quá trình phân tích.

+ Logram có thể hỗ trợ hiệu quả việc phân tích log trực tuyến bằng cách cập nhật liên tục từ điển n-gram khi các log mới được thêm vào theo phương thức streaming.

1.3. Đóng góp chính

+ Trình bày thiết kế chi tiết của phương pháp Logra, tận dụng từ điển n-gram để đạt được độ chính xác và hiệu suất cao.

+ So sánh hiệu suất của Logram với các phương pháp phân tích log tiên tiến khác thông qua đánh giá trên 16 tập dữ liệu log.

+ Triển khai Logram trên Spark và chứng minh rằng Logram có thể mở rộng hiệu quả khi số lượng node trong Spark tăng lên mà không ảnh hưởng đến độ chính xác phân tích.

+ Chứng minh Logram có thể hỗ trợ hiệu quả quá trình phân tích log trực tuyến, đạt được kết quả phân tích và hiệu suất tương đương với chế độ ngoại tuyến.

+ Logram có thể tự động xác định ngưỡng xuất hiện của n-gram để phân biệt giữa phần tĩnh và phần động trong các thông điệp log.

1.4. n-gram

Mục đích của phân tích log là phân tách các biến động có trong log để tạo thành các mẫu (template, event) tĩnh.

- N-gram là một chuỗi liên tiếp gồm phần tử (ký tự, từ, hoặc âm tiết) xuất hiện trong một đoạn văn bản hoặc chuỗi dữ liệu nào đó.

Logram sử dụng n-gram theo token:

+ Bigrams: "The cow", "cow jumps", "jumps over", "over the", "the moon".

+ Trigrams: "The cow jumps", "cow jumps over", "jumps over the", "over the moon".

Logram trích xuất các n-gram từ dữ liệu log và lưu trữ tần suất của từng n-gram trong n-gram dictionary. Việc tìm tất cả n-gram trong một chuỗi được thực hiện hiệu quả chỉ với một lần quét qua chuỗi đó (độ phức tạp tuyến tính: ).

Ngoài ra, cấu trúc của từ điển n-gram cho phép xây dựng song song, chẳng hạn bằng cách tạo từ điển riêng cho các phần khác nhau của log, sau đó tổng hợp chúng lại. Hơn nữa, từ điển n-gram có thể được cập nhật trực tuyến khi có thêm log mới.

1.5. Các công trình liên quan

Các phương pháp phân tích nhật ký hiện có có thể được nhóm thành 03 loại:

+ Phương pháp dựa trên tập luật (Rule-based)

+ Phương pháp dựa trên mã nguồn (Source code-based)

+ Phương pháp dựa trên khai thác dữ liệu (Data mining-based).

1.5.1. Phương pháp dựa trên tập luật (Rule-based)

Các công cụ quản lý và xử lý log thuộc loại này hỗ trợ người dùng xác định các quy tắc tùy chỉnh để phân tích log. Tuy nhiên, các công cụ này đòi hỏi nhiều công sức từ con người để thiết kế và duy trì các quy tắc khi định dạng log thay đổi theo thời gian. Việc sử dụng các định dạng log tiêu chuẩn hóa có thể giúp giảm bớt gánh nặng trên. Tuy nhiên, trên thực tế, các định dạng log tiêu chuẩn chưa bao giờ được sử dụng rộng rãi.

1.5.2. Phương pháp dựa trên mã nguồn (Source code-based)

Mỗi sự kiện log tương ứng với một câu lệnh log cụ thể trong mã nguồn. Các phương pháp này sử dụng phân tích tĩnh để trích xuất mẫu log từ các câu lệnh log trong mã nguồn. Sau đó, chúng tự động tạo biểu thức chính quy để khớp với các thông điệp log tương ứng với từng mẫu log được trích xuất. Tuy nhiên, phương pháp này có những hạn chế đáng kể:

+ Mã nguồn thường không có sẵn để phân tích log, đặc biệt khi log được tạo ra bởi phần mềm đóng gói hoặc thư viện của bên thứ ba.

+ Ngay cả khi có mã nguồn, quá trình phân tích tĩnh vẫn rất phức tạp, do cần xử lý nhiều thư viện logging khác nhau và các ngôn ngữ lập trình khác nhau.

1.5.3. Phương pháp dựa trên khai thác dữ liệu (Data mining-based).

Một số phương pháp phân tích log tự động không yêu cầu mã nguồn mà thay vào đó sử dụng các kỹ thuật khai phá dữ liệu.

+ SLCT, LogCluster, và LFA: Phân tích log bằng cách khai phá các token xuất hiện thường xuyên trong thông điệp log. Kỹ thuật đếm tần suất xuất hiện của các token và sử dụng ngưỡng xác định trước để phân biệt phần tĩnh và động.

+ LKE, LogSig, LogMine, SHISO, LenMa: Xem phân tích log như một bài toán phân cụm (clustering) và sử dụng các phương pháp khác nhau để đo lường độ tương đồng giữa hai thông điệp log. Ví dụ, LKE nhóm các thông điệp log thành các nhóm sự kiện dựa trên khoảng cách chỉnh sửa (edit distance) có trọng số theo vị trí của các token.

+ AEL: Sử dụng các quy tắc dựa trên tri thức miền, ví dụ: nhận diện các biến động bằng cách phát hiện token đứng sau ký hiệu "=". Nếu hai thông điệp log có cùng thành phần tĩnh và động, chúng sẽ được nhóm vào cùng một sự kiện.

+ Spell: Phân tích log dựa trên thuật toán chuỗi con chung dài nhất (longest common subsequence - LCS), với giả định rằng phần tĩnh của log chính là chuỗi con chung dài nhất giữa hai thông điệp log.

+ IPLoM: Phân cụm thông điệp log theo từng bước, đầu tiên dựa trên số lượng token, sau đó theo vị trí token, và cuối cùng dựa trên mối quan hệ giữa các cặp token.

+ Drain: Sử dụng cây phân cấp có độ sâu cố định để biểu diễn mối quan hệ giữa các thông điệp log. Mỗi lớp trong cây xác định một quy tắc để nhóm log (ví dụ: độ dài thông điệp log, token phía trước, độ tương đồng giữa các token).

# KIẾN TRÚC LOGRAM

Phương pháp Logram bao gồm 02 bước:

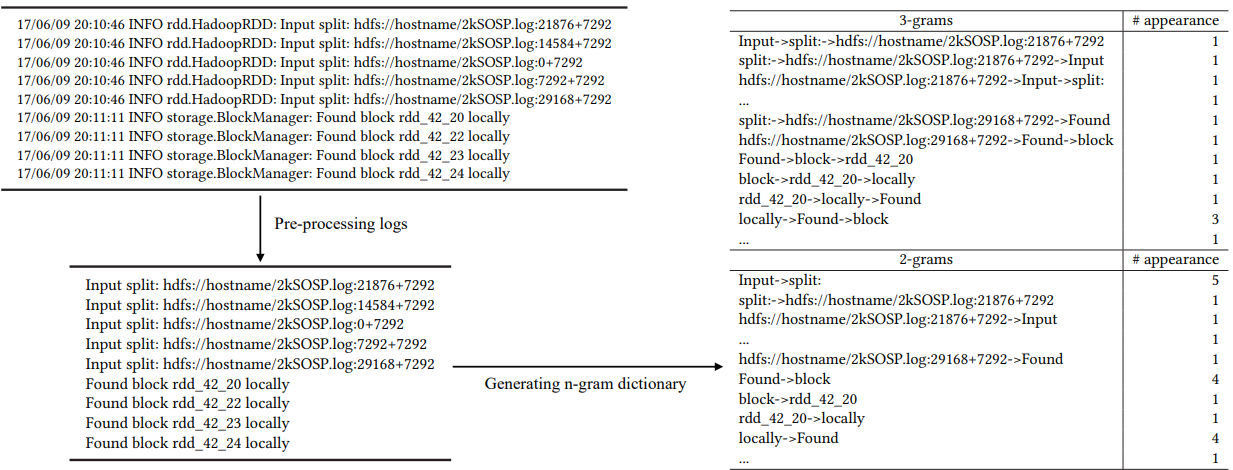
+ Bước 1. Tạo từ điển n-gram dictionary: Trích xuất các n-gram từ thông điệp log và tính số lần xuất hiện của mỗi n-gram.

+ Bước 2. Phân tích thông điệp log bằng từ điển n-gram: Chuyển đổi từng thông điệp log thành các n-gram. Dựa vào số lần xuất hiện của từng n-gram để tự động phân tách các thành phần tĩnh và biến động.

2.1. Tạo từ điển n-gram dictionary

2.1.1. Tiền xử lý log

Trước hết, trích xuất nội dung của message log dựa trên các biểu thức chính quy xác định trước như: “<Date> <Time>,<Pid> <Level> …”.



Sau khi trích xuất nội dung, thu được nội dung chính, message log sẽ được tách thành các token bằng các ký tự khoảng trắng (tab hoặc space).

Một số loại thông tin động có định dạng đặc biệt trong log, chẳng hạn như địa chỉ IP và địa chỉ email.

Để đảm bảo so sánh công bằng với các trình phân tích log khác trong bộ dữ liệu LogPai (xem Mục 5), bài báo sử dụng các biểu thức chính quy mở từ bộ dữ liệu LogPai để nhận diện các thông tin động này.

2.1.2. Tạo từ điển n-grams

Mỗi thông điệp log được chia thành các token (từ hoặc cụm từ) để tạo ra các n-grams. Tuy nhiên, không thể tạo n-gram với bằng tổng số token trong log, vì điều đó sẽ khiến n-gram gần như giống toàn bộ thông điệp log và không còn hữu ích.

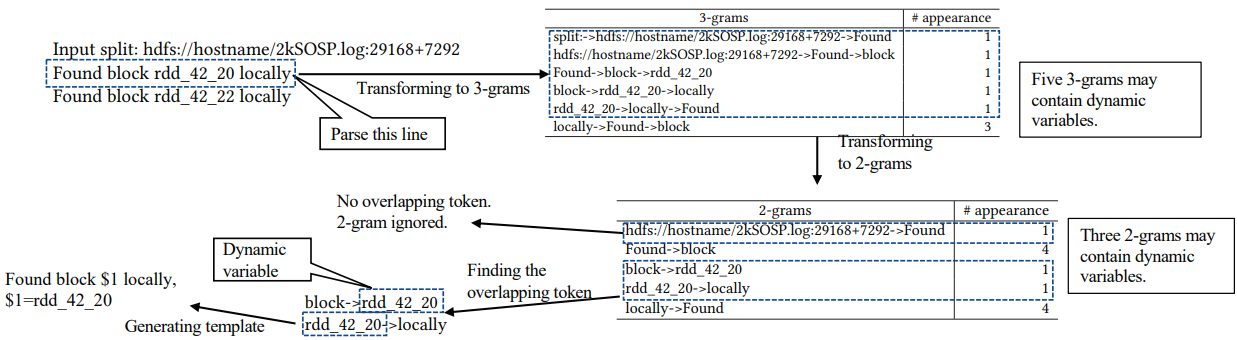
Nếu quá nhỏ, các n-gram có thể xuất hiện nhiều lần ngay cả khi chúng chứa biến động, làm giảm độ chính xác. Một nghiên cứu cho thấy khi n ≤ 3, tần suất xuất hiện của n-gram trở nên ổn định. Vì vậy, phương pháp sử dụng 2-grams và 3-grams để tạo từ điển.

Hạn chế của xử lý log độc lập: Nếu mỗi log được xử lý riêng lẻ, sẽ có hai vấn đề:

+ Một số event log có thể trải dài trên nhiều dòng.

+ Các token ở đầu và cuối log có thể không nằm trong số lượng n-grams tương tự như các token khác, làm sai lệch việc xác định phần tĩnh và động, dẫn đến thiên lệch trong xác định các biến động.

Để giải quyết vấn đề trên, Logram kết hợp các token cuối của message log trước và các token đầu của message log sau khi tạo n-gram. Ví dụ, nếu n = 3, ta sẽ sử dụng hai token cuối của log trước và hai token đầu của log sau để đảm bảo tính chính xác khi nhận diện các n-gram. Ngoài ra, Logram sẽ tính toán số lần xuất hiện của mỗi n-gram trong từ điển.



Như được minh họa trong Hình 2, từ 09 dòng log, một từ điển chứa 3-grams và 2-grams được tạo ra. Trong đó, chỉ có một 3-gram (“locally->Found->block”) xuất hiện nhiều lần; 03 2-grams (“Found->block”, “Input->split:” và “locally->Found”) xuất hiện từ 4 đến 5 lần.

Đặc biệt, một số n-gram (ví dụ: “locally->Found->block”) được tạo bằng cách kết hợp phần cuối và đầu của hai message log. Nếu không thực hiện cách này, các token như “input”, “Found” và “locally” sẽ có tần suất xuất hiện thấp hơn trong từ điển.

2.2. Phân tích message log bằng cách sử dụng từ điển n-gram

2.2.1. Xác định các n-gram chứa các giá trị biến

Trong bước này, Logram chuyển đổi mỗi message log thành n-grams và kiểm tra số lần xuất hiện của chúng trong từ điển n-gram đã tạo từ bước trước. Nếu số lần xuất hiện của một n-gram nhỏ hơn một ngưỡng tự động xác định (Mục 2.2.3), Logram giả định rằng n-gram đó có thể chứa token được tạo từ các giá trị biến.

Quy trình:

+ Bước 1. Xác định tất cả các n-grams có số lần xuất hiện thấp

+ Bước 2. Chuyển đổi các n-grams này thành (n-1)-grams, kiểm tra số lần xuất hiện của chúng trong từ điển. Mục đích: Thu hẹp phạm vi tìm kiếm các token động

+ Bước 3. Thực hiện đệ quy cho đến khi ta thu được danh sách 2-grams xuất hiện ít, mỗi 2-gram có thể chứa một hoặc hai token động.

Ví dụ: Trong hình minh họa trên, message log được chuyển thành 3-gram. Cả hai 3-grams này chỉ xuất hiện một lần trong từ điển, do đó, chúng có thể chứa giá trị biến. Sau đó, mỗi 3-gram được tách thành ba 2-grams. Một trong số đó, "Found->block", xuất hiện 04 lần → Đây có thể là một phần tĩnh của log. Hai 2-grams còn lại, "block->rdd 42 20" và "rdd 42 20->locally", chỉ xuất hiện một lần, do đó có thể chứa các giá trị biến.

2.2.2. Xác định các token được tạo động và tĩnh

1. Xác định token động trong 2-grams xuất hiện ít

Như vậy, ta thu được danh sách các 2-grams có số lần xuất hiện thấp. Tuy nhiên, không phải tất cả các token trong đó đều là các token biến. Một số 2-grams có thể chứa một token tĩnh và một token động. Trong những trường hợp này, token động phải xuất hiện trong hai 2-grams khác nhau, trong đó một 2-gram kết thúc bằng token động và một 2-gram bắt đầu bằng token động. Đối với tất cả các token khác, bao gồm cả những token được chọn trong bước này hoặc bước trước, ta coi chúng là token tĩnh.

2. Xử lý các token ở đầu và cuối log

Tuy nhiên, một trường hợp đặc biệt trong bước này là các token bắt đầu và kết thúc của mỗi message log (2.1.1). Mỗi token này chỉ xuất hiện trong một số lượng nhỏ n-gram. Ví dụ, token đầu tiên của một message log chỉ xuất hiện trong một 2-gram duy nhất. Điều này có thể dẫn đến việc hiểu nhầm token tĩnh là biến động.

Để giải quyết vấn đề này, đối với các token bắt đầu và kết thúc của mỗi message log, ta tạo thêm các n-gram bằng cách xem xét các token kết thúc từ message log trước đó; và đối với các token kết thúc của mỗi message log, ta tạo thêm các n-gram bằng cách xem xét các token bắt đầu từ message log tiếp theo.

Trong ví dụ đang xét ở Hình 3, “rdd\_42\_20” được tạo từ một biến động và nó xuất hiện trong 02 2-gram (“block->rdd\_42\_20” và “rdd\_42\_20->locally”). Do đó, ta có thể xác định token “rdd\_42\_20” là một biến động, trong khi “block” và “locally” là token tĩnh. Ngược lại, vì “hdfs://hostname/2kSOSP.log:29168+7292->Found” chỉ xuất hiện mà không có token nào trùng lặp với các 2-gram khác, nên ta bỏ qua 2-gram này khi xác định token động.

2.2.3. Xác định ngưỡng xuất hiện của n-gram một cách tự động

Việc xác định các token tĩnh và động phụ thuộc vào một ngưỡng số lần xuất hiện của n-gram. Mục tiêu là tránh việc người dùng phải tự xác định ngưỡng bằng tay, Logram đề xuất một cách tiếp cận tự động để ước lượng ngưỡng phù hợp.

Ý tưởng chính: Hầu hết n-grams tĩnh sẽ có số lần xuất hiện cao hơn, n-grams động sẽ có số lần xuất hiện ít hơn. Do đó, giữa chúng có thể tồn tại một khoảng trống rõ ràng về tần suất xuất hiện, giúp tự động xác định ngưỡng.

Các bước thực hiện

+ Bước 1. Đo lường số lần xuất hiện của mỗi n-gram.

+ Bước 2. Với mỗi giá trị xuất hiện, đếm số lượng n-grams có chính xác số lần xuất hiện đó.

+ Bước 3. Logram sử dụng biểu diễn dữ liệu trên hệ tọa độ hai chiều, với

* Trục x: Giá trị xuất hiện của n-gram.
* Trục y: Số lượng n-grams có giá trị xuất hiện đó.

+ Bước 4. Sau đó, Logram áp dụng hàm làm mượt Loess để điều chỉnh Y-values và tính đạo hàm của Y theo X. Sau khi có đạo hàm, Logram dùng thuật toán phân cụm Ckmeans.1d.dp để tìm điểm phân tách dữ liệu thành hai nhóm:

* Nhóm n-grams tĩnh (có số lần xuất hiện cao).
* Nhóm n-grams động (có số lần xuất hiện thấp).

Điểm phân tách này sẽ được sử dụng làm ngưỡng tự động.

2.2. 4. Tạo log template

Sau khi phân loại các token tĩnh và động, bước cuối cùng là tạo mẫu log (log template). Mẫu log được tạo theo định dạng chuẩn của LogPai benchmark.

Ví dụ từ Hình 3: Log gốc: "Found block rdd 42 20 locally", hệ thống phân tích và thay thế phần động "rdd\_42\_20" bằng một biến $1: "Found block $1 locally", trong đó, $1 = rdd\_42\_20.

2.3. Đánh giá

Trong phần này, chúng tôi trình bày đánh giá về phương pháp của mình. Chúng tôi đánh giá phương pháp bằng cách phân tích các log từ bộ dữ liệu LogPai benchmark [21]. Chúng tôi so sánh Logram với năm phương pháp phân tích log tự động khác, bao gồm Drain [22], Spell [59], AEL [23], Lenma [58] và IPLoM [60]. Đây đều là những phương pháp từ các nghiên cứu trước và đã được đưa vào LogPai benchmark.Chúng tôi chọn năm phương pháp này vì một nghiên cứu trước đó [21] nhận thấy rằng chúng có độ chính xác và hiệu suất cao nhất trong số các trình phân tích log đã được đánh giá. Cụ thể, chúng tôi đánh giá phương pháp của mình dựa trên bốn khía cạnh:

+ Độ chính xác (Accuracy): Độ chính xác của một trình phân tích log đo lường khả năng xác định chính xác văn bản tĩnh và các biến động trong thông điệp log, từ đó ánh xạ chính xác thông điệp log với sự kiện log tương ứng. Một nghiên cứu trước đây [20] đã chỉ ra rằng độ chính xác cao trong phân tích log là rất quan trọng, vì độ chính xác thấp có thể dẫn đến kết quả không chính xác (ví dụ: dương tính giả) trong phân tích log.

+ Hiệu suất (Efficiency): Các hệ thống phần mềm lớn thường tạo ra một lượng lớn log trong quá trình chạy [64]. Vì phân tích log thường là bước đầu tiên trong quy trình xử lý log, hiệu suất thấp có thể làm tăng chi phí xử lý và gây ra độ trễ trong việc trích xuất thông tin quan trọng từ log.

+ Độ ổn định (Ease of stabilisation): Trình phân tích log thường học từ dữ liệu log có sẵn để xác định các thành phần tĩnh và động trong thông điệp log. Càng có nhiều log đầu vào, kết quả phân tích càng chính xác. Tuy nhiên, một trình phân tích log lý tưởng nên có kết quả ổn định ngay cả khi chỉ học từ một lượng nhỏ dữ liệu log, cho phép phân tích log thời gian thực mà không cần cập nhật kiến thức liên tục.

+ Khả năng mở rộng (Scalability): Với lượng dữ liệu log lớn, người dùng có thể cân nhắc sử dụng các hệ thống xử lý song song, như Hadoop và Spark, để hỗ trợ phân tích log [65]. Tuy nhiên, nếu một phương pháp phân tích log khó mở rộng, nó có thể không được áp dụng rộng rãi trong thực tế.

# THUẬT TOÁN Ckmeans.1d.dp

Tham khảo: <https://journal.r-project.org/archive/2011/RJ-2011-015/RJ-2011-015.pdf>

3.1. Quy trình thuật toán ckmeans.1d.dp

*Yêu cầu bài toán:* Cho là một mảng đầu vào gồm số được sắp xếp theo thứ tự không giảm. Bài toán phân cụm k-means trong 1D được định nghĩa là gán các phần tử của mảng đầu vào vào cụm sao cho tổng bình phương khoảng cách trong cụm từ mỗi phần tử đến trung bình cụm tương ứng được tối thiểu hóa. Ta gọi tổng này là tổng bình phương trong cụm, viết tắt là withinss (Within-cluster sum of squares).

Phương pháp sử dụng lập trình động để đảm bảo tính tối ưu của phân cụm 1D. Định nghĩa một bài toán con là tìm *withinss* tối thiểu khi phân cụm thành cụm. Ta lưu giá trị withinss tối thiểu này vào mục của ma trận . Do đó, là giá trị withinss tối thiểu của bài toán gốc.

Trong lời giải tối ưu, giả sử là chỉ số của phần tử nhỏ nhất trong cụm trong một nghiệm tối ưu của . Khi đó, rõ ràng phải là withinss tối ưu cho phần tử đầu tiên trong cụm. Nếu không, ta có thể tìm một nghiệm tốt hơn cho . Điều này thiết lập cấu trúc con tối ưu cho lập trình động và dẫn đến phương trình truy hồi:

với

Trong đó, là tổng bình phương khoảng cách từ các phần tử đến trung bình của chúng. Ma trận được khởi tạo với .

Sử dụng phương trình truy hồi trên, ta có thể tìm được , là withinss tối thiểu có thể đạt được bằng cách phân cụm dữ liệu cho trước. Đồng thời, cho ta biết giá trị withinss tối thiểu nếu tất cả phần tử được phân thành cụm. Theo định nghĩa, mỗi phần tử trong yêu cầu thời gian để tính toán nếu được tính trong thời gian tuyến tính, dẫn đến tổng thời gian chạy là .

Tuy nhiên, có thể được tính trong thời gian hằng số bằng cách tính toán dần dần dựa trên giá trị . Cụ thể, với chỉ số từ 1 đến , ta có thể tính toán:

trong đó, là trung bình của phần tử đầu tiên. Việc tính toán này giúp giảm tổng thời gian chạy của thuật toán lập trình động xuống còn .

Để tìm ra một phân cụm có withinss tối thiểu , ta định nghĩa một ma trận phụ để ghi lại chỉ số của phần tử nhỏ nhất trong cụm :

với

Sau đó, ta truy vết ngược từ để lấy ra chỉ số bắt đầu và kết thúc của tất cả các cụm, từ đó tạo ra một nghiệm tối ưu cho bài toán k-means. Quá trình truy vết ngược này thực hiện trong thời gian .

Về độ phức tạp không gian, thuật toán lập trình động sử dụng do cần một ma trận để lưu withinss tối thiểu và một ma trận để truy vết ngược.

★ Truy vết ngược

Sau khi tính toán ma trận (chứa giá trị withinss tối ưu) và ma trận (ghi lại điểm bắt đầu của mỗi cụm trong nghiệm tối ưu), ta cần truy vết ngược để xác định các cụm trong phân cụm tối ưu. lưu chỉ số , tức là chỉ số nhỏ nhất trong cụm thứ khi phân cụm phần tử thành cụm tối ưu.

Bắt đầu từ , ta truy vết ngược để xác định chỉ số bắt đầu và kết thúc của từng cụm

+ Bước 1. Khởi tạo: Bắt đầu từ chỉ mục cuối cùng , .

+ Bước 2. Truy vết ngược các chỉ số cụm: Với mỗi cụm thứ , chỉ số đầu tiên của cụm là . Ghi nhận khoảng . Sau đó, cập nhật và giảm số cụm . Lặp lại quá trình cho đến khi tất cả các cụm được xác định.

# CÁC CHÚ Ý TRONG THỜI KHÓA BIỂU

1. Các cảnh báo

+ Xanh lá: Cảnh báo trùng lớp

+ Đỏ: Cảnh báo trùng giảng viên

+ Vàng: Cảnh báo trùng phòng học

2. Các thao tác với thời khóa biểu

+ Xóa: Ctrl + X

+ Sao chép: Ctrl + C

+ Dán: Ctrl + V

+ Quay lại: Ctrl + Z

Chú ý: Khi các số tiết đã đủ đối với một môn học nào đó thì không thể thực hiện việc paste mặc dù dữ liệu đó đã được sao chép

+ Chọn nhiều ô: Nhấn giữ Ctrl và thực hiện click chuột vào các ô cần chọn. Sau đó, tùy thuộc vào mục đích mà ta có thể thực hiện các thao tác khác.

+ Chọn một bảng nhiều ô: Muốn chọn một vùng hình chữ nhật các ô giá trị cần thao tác. Đầu tiên, chọn vào ô góc trên bên trái của vùng cần chọn (Phải hiện lên viền đỏ nhấp nháy tại ô đó), sau đó, giữ phím Shift và chọn vào ô góc dưới bên phải, như vậy ta đã chọn được vùng hình chữ nhật cần thao tác.

+ Kiểm tra thông tin: Di chuột vào ô cần kiểm tra, ô sẽ hiện lên hai biểu tượng, chọn biểu tượng info (biểu tượng thứ 2 từ trên xuống). Click chuột vào sẽ thấy thông tin lịch giảng dạy, các dấu hiệu trùng phòng, trùng giảng viên có thể có tại ô đó.

+ Nếu muốn di chuyển một ô từ ô này sang ô khác có thể thực hiện kéo thả

3. Chức năng rải lịch

Ta thực hiện rải lịch tự động bằng cách chọn vào nút “Rải lịch” ở bên trên, một cửa sổ làm việc hiện ra bên trái. Tiếp theo, chọn vào môn học cần rải lịch, tuần bắt đầu thực hiện rải lịch, số tiết cần rải trên một ô. Sau đó, ta chọn vào bảng ma trận bao gồm các ô tương ứng với ngày và buổi muốn đặt lịch dạy cho môn đó. Cuối cùng, Click vào button “Rải lịch” để rải lịch, chương trình tự động rải lịch theo yêu cầu.