TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

**KHOA CÔNG NGHỆ THÔNG TIN**



**Trương Anh Tuấn – 21120589**

**BÁO CÁO LAB02: LOGIC**

**Lớp: Cơ sở trí tuệ nhân tạo – 21\_22**

MỤC LỤC

[**GIẢI THÍCH HÀM** 3](#_Toc151579916)

[**def readInput(filename)** 3](#_Toc151579917)

[**def appendNegateAlphaToKB(alpha, KB)** 4](#_Toc151579918)

[**def are\_two\_opposite\_literals(l1, l2)** 4](#_Toc151579919)

[**def resolveTwoClauses(clause\_one, clause\_two)** 4](#_Toc151579920)

[**def PL\_RESOLUTION(clauses)** 4](#_Toc151579921)

[**def writeOutput(isSuccess, new\_clauses\_array)** 5](#_Toc151579922)

[**THAM KHẢO** 5](#_Toc151579923)

# **GIẢI THÍCH HÀM**

## **def readInput(filename)**

- Hàm này để đọc file input, các file input nằm trong thư mục input. Ví dụ tập tin “input1.txt” sẽ có đường dẫn tương đối là “input/input1.txt”

- Đọc dòng dầu tiên để lấy 𝛼, sau đó tách các literal trong 𝛼 thành các phần tử riêng của một mảng.

- Đọc dòng tiếp theo để lấy N mệnh đề trong KB (N là số lượng mệnh đề trong KB).

- Đọc N dòng tiếp theo, mỗi dòng sẽ lấy một mệnh đề, sau đó tách các literal trong mệnh đó thành các phần tử riêng của một mảng.

- Ví dụ file input có định dạng sau:

A screenshot of a computer program

Description automatically generated

- Sau khi đọc ta được kết quả:

**A black background with white text

Description automatically generated**

## **def appendNegateAlphaToKB(alpha, KB)**

- Hàm này để nối negate 𝛼 vào trong KB, tạo ra các mệnh đề mới (bao gồm cả KB), ta được kết quả.

## **def are\_two\_opposite\_literals(l1, l2)**

- Hàm này để kiểm xem hai literal có đối ngẫu hay không, ví dụ ‘B’ và ‘-B’ là đối ngẫu, ‘B’ và ‘C’ là không đối ngẫu, ‘B’ và ‘B’ là không đối ngẫu.

## **def resolveTwoClauses(clause\_one, clause\_two)**

- Hàm này để hợp giải hai mệnh đề, và kiểm tra xem có tạo ra một mệnh đề mới hay không, và sắp xếp các literal trong mệnh đó theo thứ tự bảng chữ cái.

- Ví dụ hợp giải hai mệnh đề ‘-R OR -T OR -Z’ và ‘R OR S OR -T, ta được một mệnh đề mới:



## **def PL\_RESOLUTION(clauses)**

- Hàm này sẽ chạy hai vòng lặp lồng nhau để hợp giải hai mệnh đề trong mảng ‘clauses’ được tạo ra từ hàm def appendNegateAlphaToKB(alpha, KB).

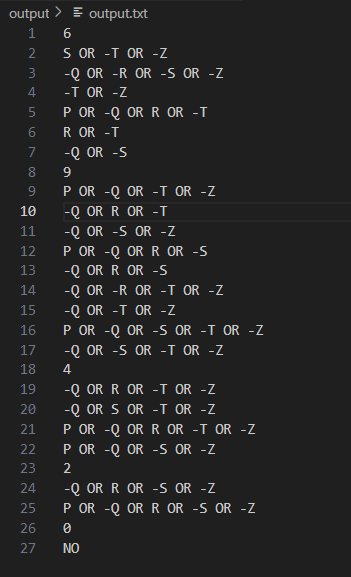
- Nếu trong lúc chạy hai vòng lặp này mà sinh ra một mệnh đề rỗng thì KB entails 𝛼.

- Nếu chạy xong hai vòng lặp này mà không được thêm mệnh đề nào thì KB not entail 𝛼.

- Nếu chạy xong hai vòng lặp này sinh ra các mệnh đề mới không rỗng, ta sẽ nối các mệnh đề đó vào mảng ‘clauses’, rồi tiếp tục chạy hai vòng lặp lồng nhau để hợp giải hai mệnh đề trong mảng ‘clauses’ và kiểm tra các điều kiện trên.

## **def writeOutput(isSuccess, new\_clauses\_array)**

- Hàm này để viết kết quả ra file output với cấu trúc theo đề yêu cầu, ví dụ với file input như trên thì khi chạy chương trình sẽ được file output như sau (file có đường dẫn tương đối là “output/output.txt”:



# **THAM KHẢO**

* Grace Nkurikiyinka. *Understanding Search Algorithms in AI*, December 16, 2021. URL: <https://www.section.io/engineering-education/understanding-search-algorithms-in-ai/>
* Hisham Touma, Ryan Kunkel. *Search Algorithms*, Updated Apr 1, 2023*.* URL: <https://www.codecademy.com/resources/docs/ai/search-algorithms>
* Online. *Difference between Informed and Uninformed Search in AI*, Last Updated: 16 Feb, 2023. URL: <https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/>
* Karleigh Moore, Ken Jennison, and Jimin Khim. *Depth-First Search (DFS*), October 18, 2023. URL: <https://brilliant.org/wiki/depth-first-search-dfs/>
* Reducible channel. *Depth First Search (DFS) Explained: Algorithm, Examples, and Code*, Jul 6, 2020. URL: <https://www.youtube.com/watch?v=PMMc4VsIacU&t=849s>
* Online. *Search Algorithms in AI*, Last Updated: 22 Mar, 2023. URL: <https://www.geeksforgeeks.org/search-algorithms-in-ai/>
* Online. *A\* – thuật toán tìm kiếm A\**. URL: <https://bloglaptrinh2016.wordpress.com/2016/06/09/a-thuat-toan-tim-kiem-a/>
* Graham Cox. *A\* Pathfinding Algorithm*. URL: <https://www.baeldung.com/cs/a-star-algorithm>
* [Gang Wu](https://www.baeldung.com/cs/author/gangwu). *Comparison Between Uniform-Cost Search and Dijkstra’s Algorithm*. URL: <https://www.baeldung.com/cs/uniform-cost-search-vs-dijkstras>
* Online. *Greedy Best first search algorithm*, Last Updated: 04 Apr, 2023. URL: <https://www.geeksforgeeks.org/greedy-best-first-search-algorithm/>