**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A blue and white logo

Description automatically generated~~~~~~\*~~~~~~**

**PROJECT: GEM HUNTER**

**Cơ Sở Trí Tuệ Nhân Tạo**

|  |  |
| --- | --- |
| **MSSV** | **Họ và tên** |
| **21120083** | Phan Văn Khải |
| **21120126** | Nguyễn Tấn Hoàng Sa |
| **21120197** | Cao Nguyễn Tuấn Anh |
| **21120318** | Lê Minh Quý |

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:** | Nguyễn Ngọc Đức |
|  | Nguyễn Trần Duy Minh |

**HỒ CHÍ MINH – 2024**

1. Đánh giá mức độ hoàn thành, hướng dẫn chạy, đóng góp của mỗi thành viên:
2. Đánh giá mức độ hoàn thành:
3. Hướng dẫn chạy đồ án:
4. Đóng góp của mỗi thành viên: mỗi người 25% nhé
5. Giải thích cấu trúc file, thư mục source

|  |  |
| --- | --- |
| Hình ảnh thư mục | Giải thích ý nghĩa từng file |

1. Giải thích chi tiết cách thực hiện:

Ở đây chúng ta giả sử file input.txt đầu vào là đúng với định dạng cho trước để khi đọc file thì dữ liệu file được chính xác.

1. Mô tả nguyên tắc tạo CNFs:
2. Khởi tạo CNF tự động:
3. Sử dụng pysat giải quyết CNFs:
4. Giải quyết CNF mà không sử dụng thư viện:

* Sử dụng thuật giải DPLL để tìm bộ nghiệm thỏa yêu cầu để bài:

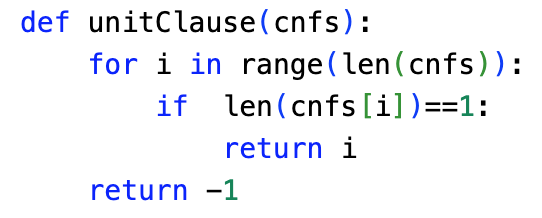
|  |
| --- |
|  |

1. Input/Output:

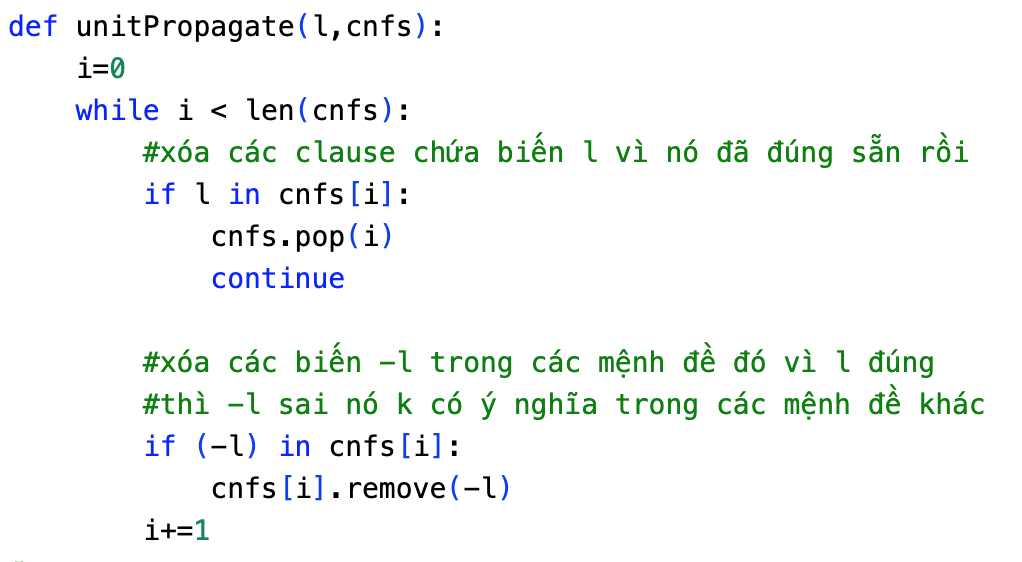
* Input:
  + cnfs: Danh sách các biểu thức logic đã chuyển về dạng CNF lấy được từ các hàm ở trên.
  + puzzle: Danh sách đọc được từ file gồm các Trap và Gem chưa được điền.
* Output: Trả về 1 trong 2 ý sau
  + True và danh sách là puzzle đã được điền thành công Trap và Gem.
  + False, None (biểu thị cho không thể điền puzzle hợp lí)

1. Chi tiết thuật toán:

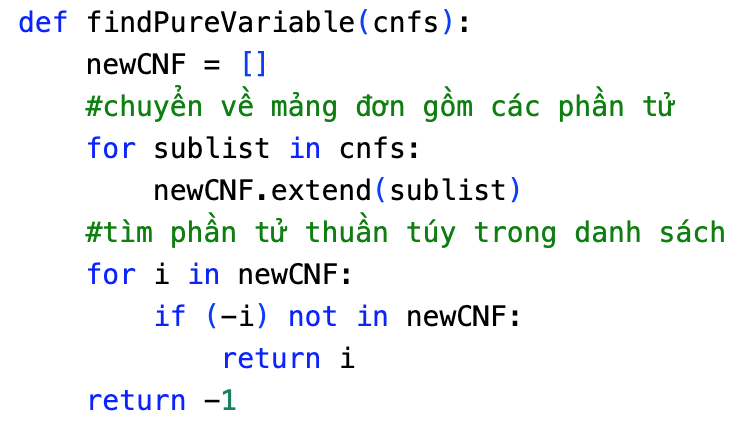
* Hàm **unitClause(cnfs):** Kiểm tra trong cnfs có còn mệnh đề nào là mệnh đề đơn hay không, nếu có trả về vị trí của mệnh đề đơn đó, ngược lại trả về -1.



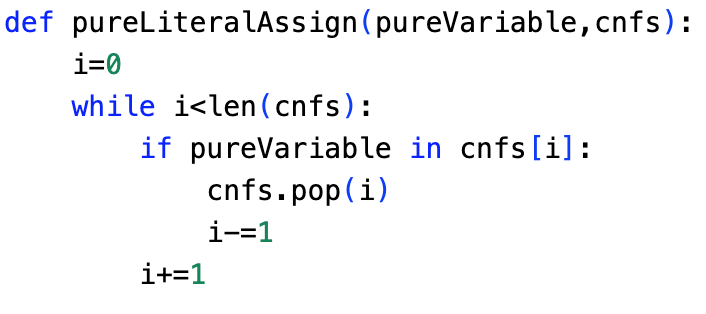
* Hàm **unitPropagate(l, cnfs):**
  + Xóa các mệnh đề chứa biến đơn l.
  + Xóa các biến là -l trong các mệnh đề khác.



* Hàm **findPureVariable(cnfs):** Tìm biến thuần túy trong cnfs
  + Biến thuần túy là biến chỉ tồn tại nó trong cnfs mà không tồn tại bù của nó trong cnfs.



* Hàm **pureLiteralAssign(pureVariable, cnfs)**: Xóa các mệnh đề mà chứa biến thuần túy ra khỏi cnfs.



* Hàm **DPLL(originalCNFs, listPuzzle)**: Thực hiện chính DPLL theo thuật giải DPLL trình bày ở trên. Tham số đầu vào là originalCNFs là cnfs ban đầu, listPuzzle là mảng 1 chiều của mảng puzzle ban đầu. Trả về listPuzzle sau khi điền Trap, Gem nếu True và ngược lại trả về Sai.
* Hàm **doDPLL(puzzle\_origin)**: Hàm này được gọi ở hàm main để thực hiện. Nhiệm vụ chuyển listPuzzle trả về từ DPLL() ở trên thành dạng mảng 2 chiều và trả về cho main.

1. Thuật toán Brute-force:
2. Thuật toán Backtracking:
3. Testcase và so sánh kết quả:
4. Thoroughness in analysis and experimentation:
5. Testcase input/ouput:

Nhớ làm input/ kết quả dự kiến, output làm ra

1. Testcase 1:
2. Testcase 2: Ma trận 9x9

|  |  |
| --- | --- |
| input2.txt |  |
| Kết quả dự kiến |  |
| output2.txt |  |
| Kết luận: Ta thấy kết quả dự kiến giống với kết quả chúng ta chạy ra trên output2.txt | |

1. Dsd
2. De
3. Efef
4. Gddfd
5. Gfgfg
6. Testcase 8: Ma trận 20x20:

|  |  |
| --- | --- |
| input8.txt |  |
| Kết quả dự kiến | False |
| output8.txt | False |
| Kết luận: Ta thấy kết quả dự kiến giống với kết quả chúng ta chạy ra trên output8.txt | |

1. So sánh kết quả và hiệu suất:
2. Tài liệu liên quan:
3. Tài liệu tham khảo:

DPLL: <https://en.wikipedia.org/wiki/DPLL_algorithm>

1. Link video demo:

I,II,IV mỗi người một phần thứ 6 họp chia ra.(Quý lm phần 3 chủ yếu)