ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CƠ SỞ TRÍ TUỆ NHÂN TẠO

| CHỦ ĐỀ |

PROJECT: GEM HUNTER

| GIẢNG VIÊN HƯỚNG DẪN |

Nguyễn Ngọc Đức

Nguyễn Thị Thu Hằng

Nguyễn Trần Duy Minh

| CÁC THÀNH VIÊN NHÓM |

21120247 – Nguyễn Văn Quang Hưng

21120298 – Chiêm Bỉnh Nguyên

21120262 – Nguyễn Huỳnh Hữu Khang

21120254 – Lưu Chấn Huy

Thành phố Hồ Chí Minh - 2024

MỤC LỤC

1. Đánh giá mức độ hoàn thành

STT	Nội dung	Mức độ hoàn thành
1	Solution description: Describe the correct logical principles for generating CNFs.	100%
2	Generate CNFs automatically	100%
3	Use pysat library to solve CNFs correctly	100%
4	Implement an optimal algorithm to solve CNFs without a library	100%
5	Program brute-force algorithm to compare with your chosen algorithm(speed) Program backtracking algorithm to compare with your chosen algorithm (speed)	100%
6	Thoroughness in analysis and experimentation Give at least 5 test cases with different sizes (5x5, 9x9, 11x11, 15x15, 20x20. . .) to check your solution Comparing results and performance	100%

2. Giải quyết vấn đề

2.1 Mô tả những quy tắc logic chính xác để tạo CNF

- Kiểm tra các ô lân cận: Việc xác định các ô lân cận có thể là bẫy hay không là một bước quan trọng để từ đó tạo ra các rang buộc logic cho bài toán. Chỉ xét những ô là ô rỗng (‘_’) vì chỉ những ô này mới có khả năng là bẫy chưa được xác định. Tiến hành lưu các ô này vào một list để sử dụng tạo ràng buộc.
- Tiến hành tạo CNF:
    - Xét điều kiện một ô là bẫy: Đối với mỗi ô số, tạo CNF từ điền kiện rằng không có nhiều hơn số bẫy cho phép bằng cách xem xét tất cả tổ hợp có thể của n + 1 ô từ các ô lân cận với n là số lượng bẫy được hiển thị trên ô số. Mỗi tổ hợp này sẽ được biển diễn dưới dạng một mệnh đề phủ định, thể hiện rnafg không thể cùng lúc có nhiều hơn n ô là bẫy trong các ô lân cận này.
    - Xét điều kiện một ô không là bẫy: Đảm bảo rằng có ít nhất n ô là bẫy trong các ô lân cận bằng cách chọn các tổ hợp của m – n + 1 với m là tổng số lân cận và tạo các mệnh đề khẳng định, chỉ ra rằng không thể có ít hơn n ô là bẫy trong các ô này.
- Tiếp theo tiến hành chuyển đổi các tổ hợp này thành mệnh đề CNF và xử lí

2.2 Giải thích các file trong source code

- main.py: Đây là hàm khởi chạy trò chơi
- Gem_Hunter.py:
    - GemHunter class: khởi tạo đối tượng với tên file input và khởi chạy trò chơi
    - getAnswer: Chạy thuật toán được chọn và trả về kết quả.
    - convertResult: Chuyển đổi kết quả từ mô hình logic sang định dạng bảng trò chơi.
- cnf.py:
    - GenerateTruthTable: Tạo bảng chân lý từ các mệnh đề DNF.
    - GenerateCNF: Hàm này chuyển đổi từ bảng chân lý và DNF sang CNF.
    - de_morgan: Áp dụng quy tắc De Morgan để chuyển đổi các mệnh đề.
    - generate_clauses: Hàm này tạo ra CNF từ ma trận đầu vào của trò chơi.

- optimal.py:
    - walksat: Hàm này thực hiện thuật toán WalkSAT, một biến thể của thuật toán SAT solver.
    - satisfies: Kiểm tra xem một phân bổ có thỏa mãn một mệnh đề cụ thể không.
    - choose_var: Chọn biến để đảo ngược giá trị nhằm cải thiện tình trạng thỏa mãn của CNF.
    - optimal: Sử dụng WalkSAT để tìm giải pháp tối ưu cho bảng.
- bruteforce.py:
    - brute_force_sat: Hàm này thực hiện thuật toán vét cạn trên CNF, tạo ra tất cả các phân bổ có thể và kiểm tra xem có thỏa mãn CNF hay không.
    - bruteForce: Hàm này chuẩn bị dữ liệu và gọi brute_force_sat để tìm giải pháp cho bảng trò chơi.
- backtrack.py:
    - backtracking: Đây là hàm triển khai thuật toán quay lui cho bài toán SAT, sử dụng CNF làm đầu vào. Hàm này kiểm tra xem có thể tìm thấy giải pháp thỏa mãn CNF hay không và trả về mô hình nếu tìm thấy.
    - getNewClauses: Hàm này tạo ra một CNF mới bằng cách loại bỏ hoặc cập nhật các mệnh đề sau khi một biến đã được gán giá trị.
    - backtrack: Hàm này khởi tạo và bắt đầu quá trình quay lui bằng cách tạo CNF từ bảng và gọi hàm backtracking.

So sánh:

Optimal và backtracking có tốc độ chạy nhanh hơn rất nhiều so với Bruteforce, với kích thước 5x5 Bruteforce mất tận 151s trong khi 2 giải thuật kia lại tốn chưa đến 1s,

Và Optimal cũng tốt hơn so với Backtracking khi dữ liệu càng lớn

3. Kết quả chạy chương trình

Kết quả chạy chương trình nằm trong video này:
https://youtu.be/cFkiSQb2uOA

4. TÀI LIỆU THAM KHẢO

LoiNguyennn/Gem-hunter (github.com)