TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-TPHCM
KHOA CÔNG NGHỆ THÔNG TIN

MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO

ĐỒ ÁN 1 – Cryptarithmetic Problem in AI

Sinh viên thực hiện:

Hà Minh Khang – 21127623

Kuo Yung Sheng – 21127684

Nguyễn Đình Vinh – 21127724

Giảng viên hướng dẫn: Lê Ngọc Thành

MỤC LỤC

1. Mô tả

1.1. Yêu cầu

Giải quyết vấn đề **cryptarithmetic** bằng thuật toán trong môn học AI. Các vấn đề được chia thành 4 cấp độ.

Level 1: Giải quyết với 2 toán hạng và 1 toán tử (+ hoặc -)

Level 2: Giải quyết với đa toán hạng và toán tử (+ hoặc -)

Level 3: Giải quyết với đa toán hạng và toán tử (+ hoặc -)kèm theo dấu "()" đại diện cho ưu tiên tính toán.

Level 4: Giải quyết level 3 kèm theo phép nhân 2 toán hạng và 1 toán tử (*).

Một vài phương trình sẽ có nhiều kết quả khác nhau hoặc không có kết quả.

Lưu ý: Tất cả các phương trình đưa ra đều phải tuân theo định dạng hợp lệ và không được để chữ số 0 đứng đầu con số (VD: 0123).

1.2. Môi trường lập trình

Ngôn ngữ lập trình: python

Môi trường thực thi: terminal

1.3. Input

Đầu vào là một text file bao gồm một dòng duy nhất để diễn tả thuật toán. Các kí tự hợp lệ bao gồm:

Kí tự in hoa A-Z

Toán tử: +, - , *

Dấu ngoặc đơn: ()

Dấu bằng: =

Ví dụ: SEND+MORE=MONEY

1.4. Output

Đầu ra là một file text với giá trị chữ số tượng trưng cho chữ cái đã được gán. Các chữ số này tuân theo các chữ cái đã được sắp xếp theo bảng chữ cái. Không có khoảng trống giữa các chữ số.

VD: SEND+MORE=MONEY, kết quả là DEMNORSY = 75160892, output sẽ là 75160892.

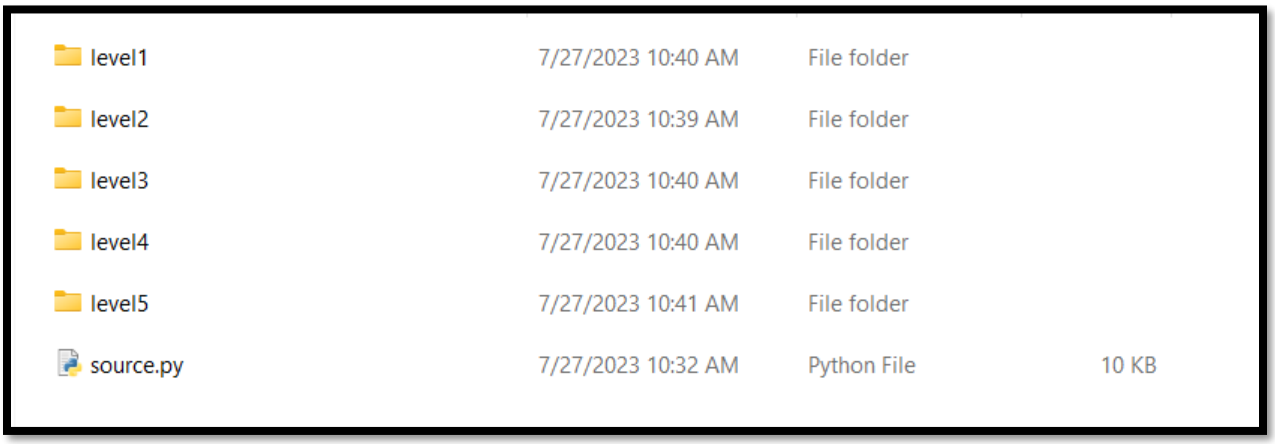
2. Cài đặt chương trình

Chương trình bao gồm một file source.py duy nhất để giải quyết 4 cấp độ bài toán yêu cầu.

2.1. Xử lý input

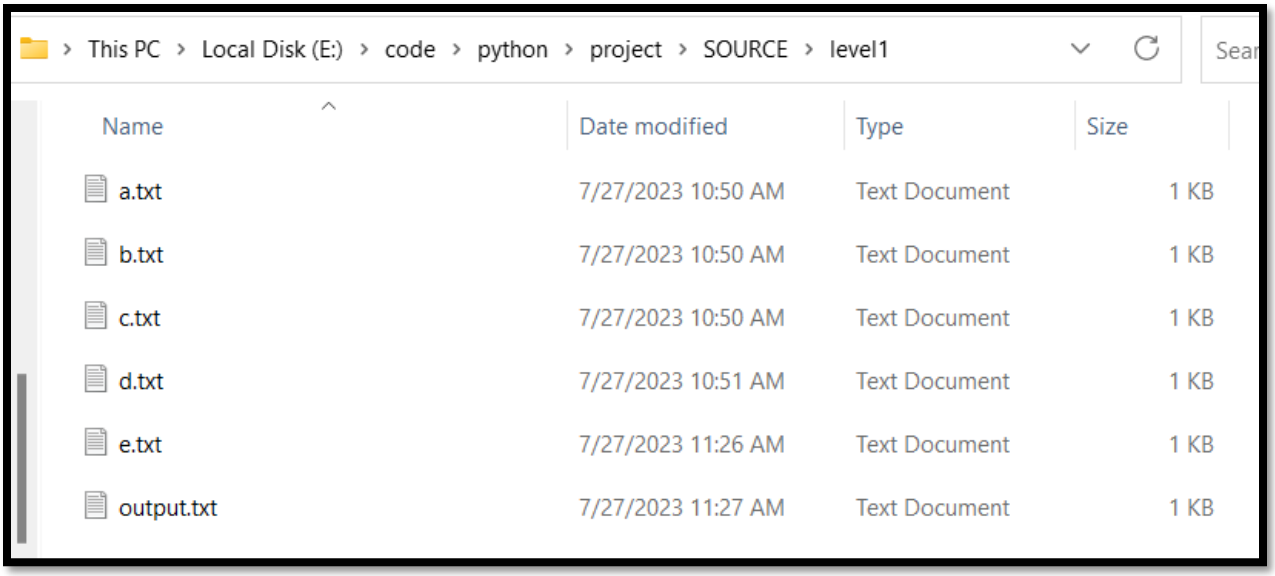
Hàm xử lý input sẽ yêu cầu người dùng nhập vào tên folder và tên file, trả về kết quả là một chuỗi trong file và một đường dẫn output.

Do thuật toán được thiết kế cho tất cả level nên thư mục được tổ chức thành 1 file source code nằm ở ngoài và các file text nằm bên trong các thư mục.



Tổ chức cây thư mục

Trong mỗi thư mục, các file text có dạng như sau:



Các file text trong thư mục con

Trong folder level2, level3, level4 và level5, các file text cũng có tên tương tự như level 1. Các file text chứa test case bao gồm: a.txt, b.txt, c.txt, d.txt, e.txt.

Các file text (a, b, c, d, e) chứa test case sẽ tuân theo thứ tự của các bộ test case cho từng level (được liệt kê ở phần 4 bên dưới).

Ví dụ minh họa cho cách nhập:

```
PS E:\code\python\project\SOURCE> & "C:/Users/My Computer/AppData/Local/Programs/Python/Python310/python.exe"
Enter folder name: level3
Enter file name: c.txt
```

Ví dụ 1

Chỉ cần nhập folder chứa tên file và tên của file kèm phần mở rộng là có thể thực thi được, trong ví dụ trên là folder level3 và file là c.txt.

2.2. Xử lý output

Đầu vào là đường dẫn output được lấy từ xử lý input.

Sau khi chọn đường dẫn cho input ở trên thì file output sẽ được xuất hiện trong thư mục vừa chọn. Theo như ví dụ 1 ở trên thì khi chọn folder là level3 thì file output.txt sẽ được xuất ở thư mục level3.

Nếu tìm được solution, trong assignment có dạng sau. Ví dụ:

Input: TWO+TWO=FOUR

Sau khi tìm được solution trong assignment: {'F': 1, 'O': 4, 'R': 8, 'W': 3, 'U': 6, 'T': 7}

Lúc này tạo một mảng để lưu các chữ cái và sắp xếp theo thứ tự bảng chữ cái. VD:

['F', 'O', 'R', 'T', 'U', 'W'].

Dò tìm sao cho các kí tự trong assigment giống với kí tự trong mảng được sắp xếp, nếu giống nhau thì gán giá trị chữ số của chữ cái đó vào chuỗi output. Output cho trường hợp trên là 148763.

2.3. Các class và hàm bên trong class

class csp:

Nodes: each character is a node.

Strings: the problem replaced by having character.

Equation: equation’s constraint extracted from problem.

class node:

character: character of that’s node.

domain=[0,1,2,3,4,5,6,7,8,9]

values=the values of that’s character.

Class cryptarithmetic:

Variable=[]: An array to store each characters (nodes) from string.

Assignment={}: Assign character to a number from domain.

Equations=[]: A list to store equations.

Hàm solution(self,problem)

Đầu vào: problem: truyền một vấn đề vào hàm.

Hàm sẽ tổng hợp các hàm sau đây (bằng cách gọi hàm xử lý và constraint) để đưa ra kết quả cuối cùng.

Hàm tackle_input(self, input_string)

Xử lý chuỗi đầu vào (input_string) và tách chuỗi ra thành phần trước dấu bằng và sau dấu bằng.

Hàm trả về: left_x: Mảng lưu các từ trước dấu "=".

result: lưu kết quả chuỗi sau dấu "=".

operator_input: Mảng lưu các toán tử trước dấu bằng "="

Ngoài ra, hàm còn trả về một chuỗi sau khi đã xử lý dấu "()".

VD: "TWO+TWO=FOUR"

left_x : ["TWO","TWO"]

result: "FOUR"

operator_input: "+"

Note: Ý tưởng để xử lý tách parentheses ().

List=[]

Duyệt từng phần tử của chuỗi nếu gặp "(" nếu kí tự trước đó là "+" thì thêm 1 vào list còn "-" thì thêm – 1.

Duyệt từng phần tử của chuỗi nếu gặp ")" thì xóa phần tử cuối của list.

Nếu tại phần tử đó là dấu thì check xem list có rỗng không. Nếu list là rỗng thì không thay đổi gì. Nếu list khác rỗng thì nhân tất cả phần tử trong list. Nếu kết quả < 0 thì đổi dấu tại phần tử đó.

Hàm removeNode(self,node_check)

Đầu vào: node_check: a node need to be removed from variable[]

Hàm dùng để xóa một character khỏi variable[] đồng thời xóa một giá trị trong domain[] sau khi character đó đã được gán với môt giá trị chữ số.

Hàm backTracking(self,assignment,csp)

Ý tưởng**:**

1. Thực hiện phép toán theo thứ tự từ phải sang trái.
2. Áp dụng backtracking cho mỗi character.domain đang được xét: Chạy vòng lặp cho từng giá trị trong domain của character chưa biết, mỗi vòng lặp thực hiện một phép backtracking với giá trị tại vòng lặp đó.
3. Nếu phương trình chưa biết được giá trị gán với 1 chữ (các chữ còn lại đều biết giá trị gán) thì tìm chữ còn lại dựa trên những chữ đã gán với giá trị.

4. Nếu các chữ của phương trình đã gán hết thì thay vào phương trình để xem đó là các phép gán đó là đúng hay sai.
5. Nếu sai thì trả về None nếu đúng thì backtracking cho Node tiếp theo.
6. Nếu tất cả chữ trong bài toán đều có thay vào bài toán để xem các chữ có thực hiện phép tính đúng không. Nếu không thì trả về None, đúng thì trả ra kết quả.

Kết quả trả về: assignment: kết quả cuối cùng khi gán xong

Hàm valid_Word(self,left_x,result)

Kiểm tra tính hợp lệ chữ cái đầu tiên của mỗi từ (chữ cái đầu phải khác 0).

Hàm sẽ xóa giá trị 0 trong domain[] của character trong variable[].

Đầu vào:

left_x: Mảng lưu các từ trước dấu "=".

result: lưu kết quả sau dấu "=".

2.4. Các ràng buộc cho problem (constraints)

Hàm sameLen(self,left_x,result)

Đầu vào:

left_x: Mảng lưu các từ trước dấu "=".

result: lưu kết quả sau dấu "=".

Hàm sẽ xử lý cho ràng buộc với phép toán (2 toán hạng và 1 toán tử).

Nếu 2 toán hạng (left_x[0] và left_x[1]) có chung một độ dài a mà kết quả (result) có độ dài a+1 thì mặc định gán kí tự đầu của result là 1.

VD: TWO+TWO=FOUR thì gán F=1.

Hàm all_diff(self,left_x,result)

Đầu vào:

left_x: Mảng lưu các từ trước dấu "=".

result: lưu kết quả sau dấu "=".

Hàm dùng để tách các chữ cái riêng lẻ từ một từ (các chữ cái không được lặp lại) và lưu vào variable[].

Hàm getEquation(self,left_x,right_x,operators_input)

Đầu vào:

left_x: left_x: Mảng lưu các từ trước dấu “=”.

right_x: lưu kết quả chuỗi sau dấu “=”.

operators_input: mảng lưu các toán tử.

Hàm dùng để biến từng cột trong phép tính thành phương trình rồi đưa nó vào array.

VD: “TWO+TWO=FOUR”

Array sẽ lưu “O+O=R”, “W+W=U”, “T+T=FO”.

3. Nhận xét thuật toán

Thuật toán sẽ ngày càng tối ưu khi tìm được càng nhiều constraints cho một biến (kí tự). Từ đó, các giá trị trong domain của một kí tự đó sẽ được loại bỏ dần, từ đó ta có thể dễ dàng backtracking lấy được giá trị của kí tự đó.

Tuy nhiên, các constraints đa phần chỉ áp dụng được với dạng toán (+, -) hai toán hạng (level 1). Vì rất khó để tìm được các constraints cho các level khác.

Kiểm nghiệm ("TWO+TWO=FOUR") cho thấy khi không cài đặt constraints thì mất trung bình 3 giây để tìm được kết quả cho level 1. Nhưng khi cài đặt constraints vào thì thời gian giải quyết một bài toán giảm đi rất đáng kể: chỉ với trung bình 0.3 giây đã giải được bài toán.

Vì vậy , càng tìm được nhiều constraint thì thời gian thực thi thuật toán càng nhanh. Vì thuật toán sẽ dựa vào constraints để loại bỏ một vài số trong domain[] của một character.

Tuy nhiên, trường hợp bài toán không có kết quả nó sẽ mất thêm thời gian để tính toán (do các constraint). Do đó, yêu cầu input đầu vào phải có tính đúng đắn của input.

Các test case càng có nhiều kí tự thì thời gian tìm ra đáp án sẽ lâu hơn những test case đơn giản.

4. Các bộ test case cho từng level kèm theo thời gian để thực thi và output

Lưu ý: Các kết quả được lấy số liệu từ trung bình của 3 lần thực thi. Có những test case nhỏ chênh lệch nhau (0.01-0.1s), những test case phức tạp thì chênh lệch nhau (1-30s). Do đó độ lệch có thể lớn. Nguồn tài nguyên của máy: RAM 16GB, 12GB available.

LEVEL 1				
STT	**Bộ test mẫu**	**Kết quả (chữ cái)**	**Kết quả (chữ số)**	**Thời gian thực thi (giây)**
1	ABCDE+FGHIK=HKBCE	ABCDEFGHJK	2478135690	≈ 54.7247
2	BASE+BALL=GAMES	ABEGLMS	4731598	≈ 0.971
3	YOUR+YOU=HEART	AEHORTUY	30146829	≈ 64.4
4	POR-RO=WC	COPRW	32157	≈ 0.57
5	COMES–SHE=HERE	CEHMORS	1493058	≈ 21.2

LEVEL 2				
STT	**Bộ test mẫu**	**Kết quả (chữ cái)**	**Kết quả (chữ số)**	**Thời gian thực thi (giây)**
1	ABC+DEF+GHI-FHK=FHH	ABCDEFGHIK	1203756984	≈ 25.6
2	SIX+SEVEN+SEVEN =TWENTY	EINSTVWXY	852617304	≈ 4.28
3	HALF+FIFTH+TENTH +TENTH+TENTH =WHOLE	AEFHILNOTW	7516403829	≈ 21
4	WEAR+A+MASK+WASH =SAFER	AEFHKMRSW	168543029	≈ 0.93
5	SO+MANY+MORE+MEN +SEEM+TO+SAY+THAT +THEY+MAY+SOON+TRY +TO+STAY+AT+HOME+SO +AS+TO+SEE+OR+HEAR+ THE+SAME+ONE+MAN+ TRY+TO+MEET+THE+TEAM +ON+THE+MOON+AS+HE	AEHMNORSTY	7052618394	≈ 13.8

	+HAS+AT+THE+OTHER+TEN =TESTS			

LEVEL 3				
STT	**Bộ test mẫu**	**Kết quả (chữ cái)**	**Kết quả (chữ số)**	**Thời gian thực thi (giây)**
1	SEND+(MORE+MONEY) -OR+DIE=NUOYI	DEIMNORSUY	1295630874	≈ 7
2	GHI+FHK-(ABC+DEF)=HIK	ABCDEFGHIK	4235978601	≈ 1.17
3	HOUDINI-(CADABRA+ABRA)-ABRA=CADABRA	ABCDHINORU	7419382650	≈ 390
4	TWO+TWO-(TWO+TWO)+TWO+ TWO+ONE=FIVE	EFINOTVW	02345971	≈ 35
5	BASE+BALL+GAMES-BASEBALL-(FUN+GAMES) +BASEBALLGAME =BASBUUFBLGGM	ABEFGLMNSU	1238456079	≈ 130

LEVEL 4				
STT	**Bộ test mẫu**	**Kết quả (chữ cái)**	**Kết quả (chữ số)**	**Thời gian thực thi (giây)**
1	GHI+FHK-(ABC+DEF)*B=CAA	ABCDEFGHIK	5431869702	≈ 5.1
2	TWO*TWO* (A-U)=SQUARE	AEOQRSTUW	724319658	≈ 95
3	(FISH+N)*S+CHIPS =SUPPER	CEFHINPRSU	9658734210	≈ 300

4	(COUPLE+COUPLE)*L -(COUPLE+COUPLE) =QUARTER	ACELOPQRTU	5289071436	≈ 250
5	(GHI+FHK- (ABC+DEF)*B+CAA)*K +(ABC+DEF+GHI+FHK +ABCDEFGHI)*K=K	ABCDEFGHIK	9853264710	≈ 46

5. Phân công công việc

STT	MSSV	Họ và tên	Nhiệm vụ phân công	Mức độ hoàn thành
1	21127623	Hà Minh Khang	Viết hàm xử lý các input, output Viết báo cáo	100%
2	21127684	Kuo Yung Sheng	Viết hàm xử lý chuỗi Viết hàm xử lý thuật toán Viết hàm xử lý constraint	100%
3	21127724	Nguyễn Đình Vinh	Viết hàm xử lý các constraint Tìm các bộ test case	100%

6. Đánh giá mức độ hoàn thành đồ án

STT	Tiêu chí	Mức độ hoàn thành
1	Hoàn thành level 1	100%
2	Hoàn thành level 2	100%
3	Hoàn thành level 3	100%
4	Hoàn thành level 4	80%
5	Khởi tạo ít nhất 5 bộ test case cho từng level	100%

Level 4 có thể có sai sót với một số bộ test đặc biệt. Nhóm em chỉ tìm được một hoặc hai trường hợp tính toán sai nên quyết định đánh giá mức độ hoàn thành cho level 4 là 80%.

7. Tài liệu tham khảo

File hướng dẫn đồ án: Project 1 – Constraint Satisfaction Problem do giảng viên thực hành cung cấp.

Google drive bài giảng Lecture06 – CSP do cô Nguyễn Ngọc Thảo cung cấp.