中華大學生物資訊學系系統開發專題報告

愛因斯坦謎題的創新研究

Innovative Research on Einstein's Riddle

專題組員:許富翔、劉禮婷

專題編號：107004

指導老師:侯玉松老師

1. 摘要

愛因斯坦謎題據說是路易斯·卡羅所做的，而路易斯·卡羅正是數學家查爾斯·路特維奇·道奇森，玩家們可以設計一張表格，依循、綜理線索，採用消去法排除已知結果，最後取得結論，再將推論結果填入對應的表格。為了讓玩家可以在有新穎的愛因斯坦謎題可解，我們將研究如何創造新謎題，讓玩法更豐富。我們採用的是回溯法來設計解題器，回溯法中有深度優先搜尋法(DFS)跟限制式檢查函式(Bounding Function)。為了產生新題目，從多條限制式中拿掉其中一條，使得限制變少，解題器會產生多組解，再以人工的方式來檢查，找出獨一無二的特徵，並想出新的限制式以及寫入限制式檢查函式中，使題目有個新答案。所有條件經由更換、互相調換物件、分開、合併，並保留原本的物件數量，最後原本的15個限制式全都換成了全新的15個限制式。做這項專題的目的是為了能利用解題器去作題目的變化。在未來發展方面，能夠更深入的利用解題器來開發新題目的方式，應用在其他推理題目上。

2. 簡介

愛因斯坦謎題據傳說並不確定是愛因斯坦本人所設計，也有人說是《愛麗絲夢遊仙境》的作者路易斯·卡羅所作，而路易斯·卡羅正是數學家查爾斯·路特維奇·道奇森[1]。題目敘述為有間水族館裡的一尾稀有的魚被偷走了，警方循線報找到一條有五棟一模一樣房子的街道，但是他們怕若找錯了房子就會打草驚蛇。還好有一位擅長偵查的警探，而且他的邏輯推理能力很強。當他到達現場時，警察向他說明目前的情況：

> 「每一間房子的屋主，各有不同的國籍，喝不一樣的飲料，抽不一樣品牌的雪茄。每一間房屋外表的顏色都不相同。每一個屋主，分別養不同的寵物，其中之一就是我們在尋找的被偷走的魚。」

還好精明的警探經過幾個小時仔細的偵查，收集到了15個線索。

針對這些的謎題，玩家們可以設計一張表格，依循、綜理線索，採用消去法排除已知結果，最後取得結論，再將推論結果填入對應的表格。

先前學長姐創造了專門解這類題目的解題器，實行結果也是非常成功。

[2]

我們在網路上看到有一篇自稱是類似愛因斯坦的謎題，這個根本就不是創新題目，只是物件的名詞更換而已，像是房屋顏色只是換成包間名稱而已，裡面的像對位置都沒有改過，整個換湯不換藥。[3]

為了讓玩家可以在有新穎的愛因斯坦謎題可解，我們將研究如何創造新謎題，讓玩法更豐富。

3. 專題進行方式

若是以傳統的暴力搜尋法來解愛因斯坦謎題，將會有5!的五次方，也就是大約248億個可行解，如果檢查一種解需要1秒鐘，就要花費約789年的時間。

我們將採用回溯法進行解題，回溯法的進行方法如下：

回溯法

回溯法是一種可以找出所有解的一般性演算法，尤其特別適用於限制式滿足問題，在解決限制式滿足問題時，要逐步構造更多的候選解，並在確定某一些候選解後，逐一檢查並刪去不適合的候選解，最後取得唯一解。[4]

回溯法中有深度優先搜尋法(DFS)跟限制式檢查函式(Bounding Function)，以愛因斯坦謎題的縮小版為舉例，先畫出3*3的表格，房屋顏色有紅色、綠色、白色，屋主國籍有英國、瑞典、丹麥，飼養寵物有魚、狗、貓。

先列出五個限制式：

<1>.綠色屋主養貓

<2>.瑞典人住在白屋子隔壁

<3>.英國人住在紅屋子左邊一格

<4>.丹麥人養狗

<5>.英國人左邊住瑞典人

3*3表格是從上而下、從左至右填寫，先從房屋顏色的位置從左至右寫下「紅色、綠色、白色」，如圖：

	0	1	2
房屋顏色	紅色	綠色	白色
國籍			
寵物			

接下來在國籍的第一個寫下「英國」，用限制式進行檢查，可發現到第3個限制式不符合題意，所以將「英國」換為「瑞典」，但不符合第2條限制式，最後換上「丹麥」，因為沒有限制式可以檢查，所以往右邊寫下「英國」，卻不符合限制式，所以返回至房屋顏色那行，將「白色」去除，但是已經沒有其他可以放置的物件，所以「綠色」也一起清除，有左至右寫入「白色」、「綠色」，下方國籍從「英國」開始用限制式檢查，如圖：

	0	1	2
房屋顏色	紅色	白色	綠色
國籍	英國		
寵物			

可以發現還是無法找出答案，應此房屋顏色的第一個「紅色」更換為「綠色」，其後兩格填入「紅色」及「白色」，下行國籍先寫下「英國」，檢查限制式並沒有產生衝突，可以往下一格邁進，填進「瑞典」卻不符合限制式，第二格換為「丹麥」，後面寫上「瑞典」，如圖：

	0	1	2
房屋顏色	綠色	紅色	白色
國籍	英國	丹麥	瑞典
寵物			

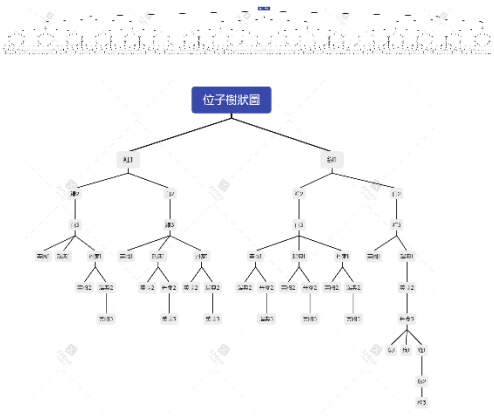
其後的第一格「瑞典」跟「丹麥」都不符合限制式，所以返回到房屋顏色，將「紅色」及「白色」對調，並重複檢查國籍排列是否符合，直到「瑞典、英國、丹麥」才完全符合限制式，如圖：

	0	1	2
房屋顏色	綠色	白色	紅色
國籍	瑞典	英國	丹麥
寵物			

最後在寵物那行的第一格寫下「魚」並開始檢查，發現不符合第1條限制式，再來換「狗」，不符合第4條，換成「貓」的時候，符合限制式，再寫下「魚」跟「狗」，最後得出答案。

以下為範例的答案以及所有可能與經過 DFS 的樹狀圖：

	0	1	2
房屋顏色	綠色	白色	紅色
國籍	瑞典	英國	丹麥
寵物	貓	魚	狗



為了產生新題目，從多條限制式中拿掉其中一條，使得限制變少，解題器會產生多組解，再以人工的方式來檢查，找出獨一無二的特徵，並想出新的限制式以及寫入限制式檢查函式中，使題目有個新答案。

時程與人員配置

以下為此專題之進度及工作分配表

職責分配	討論主題	題目設定	人工推算	報告製作	PPT製作	海報製作	影片製作	總整理
許富翔	○	○	○	○			○	○
劉禮婷	○		○		○	○	○	○

職責分配	討論主題	題目設定	人工推算	報告製作	PPT製作	海報製作	影片製作	總整理
3月	○							
4月	○							
5月	○							
6月	○							
7月	○							
8月	○							
9月	○	○						
10月		○						
11月		○						

12月		○	○	○	○	○	○	○

實作平台

RAM: 8.00 GB

處理器名稱: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

作業系統:Windows11 64 位元

技術: :logic.c-Dev-C++5.11

測試方式

以下是原題目的15條限制式以及答案：

(1).英國人住在紅色房屋裏。

(2).瑞典人養狗。

(3).丹麥人喝茶。

(4).綠色的房屋在白色房屋的左邊。

(5).綠色房屋的屋主喝咖啡。

(6).抽 Pall Mall 雪茄的屋主養鳥。

(7).黃色屋主抽 Dunhill 雪茄。

(8).位於最中間的屋主喝牛奶。

(9).挪威人住在第一間房屋裏。

(10).抽 Blend 的人住在養貓人家的隔壁.

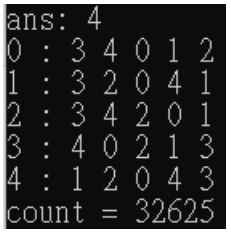
(11).養馬的屋主隔壁是抽 Dunhill 雪茄。

(12).抽 Blue Master 雪茄的屋主喝啤酒。

(13).德國人抽 Prince 雪茄。

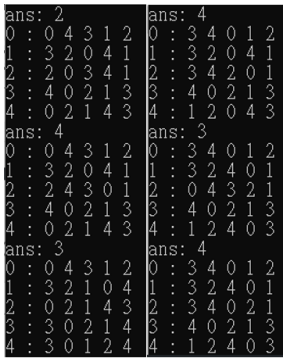
(14).挪威人住在藍色房屋隔壁。

(15).只喝開水的人家住在抽 Blend 雪茄的隔壁。

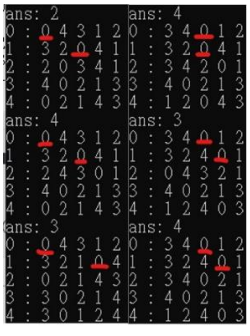


	0	1	2	3	4
房屋顏色	黃色	藍色	紅色	綠色	白色
屋主國籍	挪威	丹麥	英國	德國	瑞典
飼養寵物	貓	馬	鳥	魚	狗
飲料種類	白開水	茶	牛奶	咖啡	啤酒
香煙品牌	Dunhill	Blend	PallMall	Prince	BlueMaster

範例：我們將第一條限制式「英國人住在紅色房屋裏」抽掉並執行，會出現6種可能的答案，如圖：



「紅色」為0-0、「英國」為1-0，將他們所在位置畫底線以易辨識，如圖：



可以看到除了左下的答案以外，其他五個不是位置重複就是與原來的限制式一樣，所以採用左下的答案，並想出新的限制式，因而得出「英國人住紅色房屋的右邊三間」並寫入其程式碼，得出上圖左下的答案。

所有條件經由更換、互相調換物件、分開、合併，並保留原本的物件數量，最後原本的15條限制式都換成了全新的15條限制式。

以下為更改後的15條限制式：

{1}.英國人住紅色房屋的右邊三間

{2}.德國人養狗

{3}.丹麥人住第二間房屋

{4}.喝茶的人家住在最左邊

{5}.白色房屋在綠色房屋左邊一格

{6}.喝咖啡的人在綠屋子的左邊三間

{7}.抽 BlueMaster 雪茄的屋主養鳥

{8}.黃色房屋跟抽 Dunhill 雪茄的人分別住在最左及最右

{9}.挪威人喝牛奶

{10}.抽 Blend 雪茄的人住在養貓的人右3間

{11}.抽 Dunhill 雪茄的人養馬

{12}.抽 Pallmall 雪茄的屋主喝啤酒

{13}.瑞典人抽 Prince 雪茄

{14}.挪威人抽 Blend 雪茄

{15}.只喝開水的人家住在藍色房屋右邊兩格

困難與解決之道

我們有嘗試在15條限制式中去掉其中兩樣限制式，會產生很多解，最多有20多組解，還要用人工查找，真的很花心思，而且在原本的15條限制式中，「魚」以外的其他物件都有出現過，抽換的物件都要至少出現過一次，連程式碼都要打正確，不然執行結果不是只有 Count，就是跑出超多種可能。

4. 主要成果

下圖為更改前後的15條限制式對比

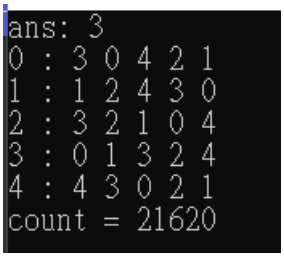
更改前	更改後
1. 英國人住在紅色房屋裏	1. 英國人住紅色房屋的右邊三間
2. 瑞典人養狗	2. 德國人養狗
3. 丹麥人喝茶	3. 丹麥人住第二間房屋
4. 綠色的房屋在白色房屋的左邊	4. 喝茶的人家住在最左邊
5. 綠色房屋的屋主喝咖啡	5. 白色房屋在綠色房屋左邊一格
6. 抽 Pall Mall 雪茄的屋主養鳥	6. 喝咖啡的人在綠屋子的左邊三間
7. 黃色屋主抽 Dunhill 雪茄	7. 抽 BlueMaster 雪茄的屋主養鳥
8. 位於最中間的屋主喝牛奶	8. 黃色房屋跟抽 Dunhill 雪茄的人分別住在最左及最右
9. 挪威人住在第一間房屋裏	9. 挪威人喝牛奶
10. 抽 Blend 的人住在養貓人家的隔壁	10. 抽 Blend 雪茄的人住在養貓的人右 3 間
11. 養馬的屋主隔壁是抽 Dunhill 雪茄	11. 抽 Dunhill 雪茄的人養馬
12. 抽 Blue Master 雪茄的屋主喝啤酒	12. 抽 Pallmall 雪茄的屋主喝啤酒
13. 德國人抽 Prince 雪茄	13. 瑞典人抽 Prince 雪茄
14. 挪威人住在藍色房屋隔壁	14. 挪威人抽 Blend 雪茄
15. 只喝開水的人家住在抽 Blend 雪茄的隔壁	15. 只喝開水的人家住在藍色房屋右邊兩格

在新的限制式套上程式碼之後，如下圖所示。

```
int x[5][5];
int count = 0;
//int found = 0;
int place(int k, int i)
{
    int j, r, c; // r: row, c: column

    count++;
    r = k/5;
    c = k%5;
    for (j=0; j<c; j++)
        if (x[r][j] == i) return(0); //同列數字不得相同
    //if (r==1 && i==0 && x[0][c] != 0) return(0); //英國人住紅色房
    if (r==1 && i==0 && c<3) return(0); //英國人住123
    if (r==1 && i==0 && c>=3 && x[0][c-3]!=0) return(0);  //英國人住紅色房的右3間

    //if (r==2 && i==1 && x[1][c] != 1) return(0); //瑞典人養狗
    if (r==2 && i==1 && x[1][c] != 4) return(0); //德國人養狗

    //if (r==3 && i==0 && x[1][c] != 2) return(0); //丹麥人喝茶
    if (r==1 && i==2 && c != 1) return(0);//丹麥人住第二間

    if (r==3 && i==0 && c !=0) return(0);//喝茶人家住在最左邊


    //if (r==0 && i==2 && c==0) return(0); //白色屋在最左邊
    //if (r==0 && i==2 && x[0][c-1] != 1) return(0); //綠屋在白屋左邊
    if (r==0 && i==1 && c==0) return(0); //綠屋子在最左邊
    if (r==0 && i==1 && x[0][c-1]!=2) return(0); //白屋子在綠屋子左邊一格

    //if (r==3 && i==1 && x[0][c] != 1) return(0); //綠屋主喝咖啡
    if (r==3 && i==1 && c>2) return(0); //喝咖啡的在345位
    if (r==3 && i==1 && c<=2 && x[0][c+3]!=1) return(0);  //喝咖啡的在綠屋子的左3間

    //if (r==4 && i==0 && x[2][c] != 2) return(0); //抽 Pall 屋主養鳥
    if (r==4 && i==3 && x[2][c] != 2) return(0); //抽 BlueMaster 屋主養鳥

    //if (r==4 && i==1 && x[0][c] != 3) return(0); //黃屋主抽 Dunhill
    if (r==0 && i==3 && c != 0) return(0);//黃屋最左
    if (r==4 && i==1 && c != 4) return(0);//Dunhill最右

    //if (r==3 && i==2 && c != 2) return(0); //中間屋主喝牛奶
    //if (r==1 && i==3 && c != 0) return(0); //挪威人住第一間房
    if (r==3 && i==2 && x[1][c] != 3) return(0); //合成挪威人喝牛奶
```

```
    //if (r==4 && i==2 && c==0 && x[2][1] != 3) return(0); //抽Blend住最左, 右鄰養貓
    //if (r==4 && i==2 && c==4 && x[2][3] != 3) return(0); //抽Blend住最右, 左鄰養貓
    //if (r==4 && i==2 && c>0 && c<4 && x[2][c-1] != 3 && x[2][c+1] !=3) return(0);
    if (r==4 && i==2 && c<3) return(0); //Blend住123
    if (r==4 && i==2 && c>=3 && x[2][c-3]!=3) return(0);  //Blend住養貓的右3間

    //if (r==4 && i==1 && c==0 && x[2][1] != 4) return(0); //抽Dunhill住最左, 右鄰養馬
    //if (r==4 && i==1 && c==4 && x[2][3] != 4) return(0); //抽Dunhill住最右, 左鄰養馬
    //if (r==4 && i==1 && c>0 && c<4 && x[2][c-1] != 4 && x[2][c+1] !=4) return(0);
    if (r==4 && i==1 && x[2][c] != 4) return(0); //抽 Dunhill 屋主養馬

    //if (r==4 && i==3 && x[3][c] != 3) return(0); //抽BlueMaster屋主喝啤酒
    if (r==4 && i==0 && x[3][c] != 3) return(0); //抽 Pall 屋主喝啤酒

    //if (r==4 && i==4 && x[1][c] != 4) return(0); //德國屋主抽Prince
    if (r==4 && i==4 && x[1][c] != 1) return(0); //瑞典屋主抽Prince

    //if (r==1 && i==3 && c==0 && x[0][1] != 4) return(0); //挪威屋主住最左, 右鄰藍色
    //if (r==1 && i==3 && c==4 && x[0][3] != 4) return(0); //挪威屋主住最右, 左鄰藍色
    //if (r==1 && i==3 && c>0 && c<4 && x[0][c-1] != 4 && x[0][c+1] !=4) return(0);
    if (r==4 && i==2 && x[1][c] != 3) return(0);//挪威人抽Blend


    //if (r==4 && i==2 && c==0 && x[3][1] != 4) return(0); //抽Blend住最左, 右鄰喝水
    //if (r==4 && i==2 && c==4 && x[3][3] != 4) return(0); //抽Blend住最右, 左鄰喝水
    //if (r==4 && i==2 && c>0 && c<4 && x[3][c-1] != 4 && x[3][c+1] !=4) return(0);
    if (r==3 && i==4 && c<3) return(0); //喝水的在45位
    if (r==3 && i==4 && c>=3 && x[0][c-2]!=4) return(0); //喝水人家住藍屋子右兩格
```

以下為執行結果與位置圖



	0	1	2	3	4
房屋顏色	黃色	紅色	藍色	白色	綠色
屋主國籍	瑞典	丹麥	德國	挪威	英國
飼養寵物	貓	鳥	狗	魚	馬
飲料種類	茶	咖啡	啤酒	牛奶	白開水
香煙品牌	Prince	Bluemaster	PallMall	Blend	Dunhill

5. 評估與展望

做這項專題的目的是為了能利用解題器去作題目的變化。在未來發展方面，能夠更深入的利用解題器來開發新題目的方式，應用在其他推理題目上。

這次專題的心得，我們了解到原來程式還可以解謎題，在十二多億個可能中找出唯一的可能。這道謎題已被學長姐們拿來做許多延伸，程式碼不斷地變化，甚至發展出新題目，讓我們知道程式有多種用途，或許還能應用在別的謎題上，像是數獨這方面需要動腦的題目。透過這次專題，讓我們學習到如何用解題器創造新題目，也增加了對程式應用的能力，還有解決問題的能力。正當還在苦惱時，我們決定詢問老師的意見，在老師的指點迷津下，最後終於順利跑出預想結果。

6. 結語

這次作品不論是運用人工解題，或是程式執行都能夠有個標準答案。解一道題目看似容易，但出一道新的題目卻是寸步難行，而利用解題器出題的過程，則是大大地降低其中的難度。這道謎題不論是用回溯法，或是用程式(解題器)解題，希望能透過這種方式讓大家認識這些方法的運作，對程式有更深一步的了解。

7. 銘謝

感謝**侯玉松**老師的指導，在百忙中抽出時間與我們討論專題的問題及解決方法，還有哪些需改進的地方。也很感謝組員的互相協助，能願意抽出時間思考問題所在，以及提出對程式上建議。

8. 參考文獻

[1] 昌爸工作坊， 如何利用排除消去法解謎題 (http://www.mathland.idv.tw/fun/riddle.htm)

[2] 楊宇傑、陳昱蓁、劉冠廷，解謎程式與活用 (https://bio.chu.edu.tw/var/file/39/1039/img/105/199509904.pdf)

[3] 潘多拉の泪，[逻辑推理] 类似爱因斯坦的谜题 (http://www.tuilixy.net/thread-24147-1-1.html)

[4] 維基百科，回溯法 (https://zh.wikipedia.org/zh-tw/%E5%9B%9E%E6%BA%AF%E6%B3%95)