中華民國第 65 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

第二名

052508

以資訊熵策略解決 Nerdle Maxi 問題：尋找最少猜測次數的演算法

學校名稱： 臺北市立成功高級中學

作者：	指導老師：
高二 蔡孟平 高二 林政忠 高二 李丰萬	江俊毅

關鍵詞： Nerdle Maxi、資訊熵、啟發式演算法

摘要

本研究試圖探索如何將資訊熵應用於 Nerdle Maxi，Nerdle Maxi 是一款數學益智遊戲，類似於 Wordle，但要求玩家猜測出一個數學等式。遊戲的核心在於考驗玩家以有限的猜測次數，通過猜測獲得的回饋，縮小可能等式的範圍，最終猜出答案。此問題涉及對資訊的處理，讓我們聯想到資訊理論中關於不確定性的研究。根據 Shannon 的資訊理論，資訊熵可以用來衡量一個系統中不確定性，從而幫助我們找出減少不確定性的最佳策略。參考文獻後，我們發現有尚無人嘗試的作法，並實作、壓縮了猜測次數，保證我們能在六次內猜重。再將其與文獻中用在 Wordle 的做法融合後，壓縮平均猜測次數至 3.38 次。

壹、前言

一、研究動機

研究各種遊戲策略時，數學推理遊戲 Nerdle Maxi 映入我們眼前，對於 Nerdle Maxi ，我們產生了一個疑問：是否可以利用程式設計讓找出遊戲的正確解答的猜測次數減少？ Nerdle Maxi 要求玩家通過不斷猜測，最終推測出正確的數學等式，而我們發現隨著排列組合數量的增加，遊戲難度顯著上升。為了探討如何有效減少花費次數，我們開始研究各種解決此問題的理論與方法，最終發現資訊理論，特別是資訊熵，提供了一個量化不確定性的工具，能夠幫助我們針對不同的猜測策略進行分析與評估。這促使我們進一步探討如何應用資訊理論來設計演算法，希望最小化 Nerdle Maxi 的花費猜測次數。

二、研究目的

（一）使用多種不同的啟發式演算法，觀察其之間的花費猜測次數、資訊熵、計算時間

（二）比較不同的資訊熵方法，並比較其猜測次數、獲得的熵值

（三）藉由前面的實驗結果，找出能用最少猜測次數之演算法

三、文獻回顧

（一）回顧 Wordle 相關之研究

由於 Wordle 的遊戲機制與 Nerdle Maxi 非常相近，並且網路上相關資訊、研究較多，因此我們從 Wordle 相關研究下手。

Lokshtanov 與 Subercaseaux (2022) 指出是否存在一組猜測策略能在有限次數內保證解出 Wordle 的問題是 *NP-hard*，即使詞語長度為 5 也成立。Rosenbaum (2022) 進一步指出，尋找一個能保證勝利的策略是 *NP-complete*，並提供了在最壞情況下使用 g 次猜測的策略，其計算時間為 $N^{O(g)}$ ，這些都凸顯了 Wordle 的計算挑戰性。

我們發現有許很多研究都是從資訊理論的角度找出最好的猜測，能有效地減少可能的答案數量，例如：3Blue1Brown [3B1B] (2022) 指出，選擇能最大化資訊熵的起始詞（如 "SOARE" ）有助於提高解題效率。

（二）回顧 Nerdle Maxi 相關之研究

查找與 Nerdle Maxi 相關的研究時，我們找到了 Nerdle Maxi 的所有可能等式，並且分析了其每種符號在每個格子出現的機率分布，如下圖。

表一、每種符號在每個格子出現的機率分布表格（r/nerdlegame, 2023）

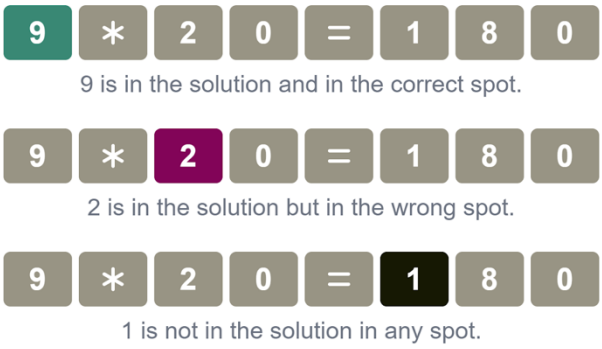
Symbol	% of equations with this symbol	% of equations with this symbol on slot 1	% of equations with this symbol on slot 2	% of equations with this symbol on slot 3	% of equations with this symbol on slot 4	% of equations with this symbol on slot 5	% of equations with this symbol on slot 6	% of equations with this symbol on slot 7	% of equations with this symbol on slot 8	% of equations with this symbol on slot 9	% of equations with this symbol on slot 10
=	100,000%	0,000%	0,000%	0,000%	0,002%	0,010%	1,093%	16,705%	45,079%	37,110%	0,000%
1	62,575%	14,557%	6,263%	7,505%	10,279%	7,966%	9,051%	9,358%	6,029%	8,821%	9,971%
-	62,087%	0,000%	6,864%	19,168%	13,279%	14,297%	13,673%	6,024%	0,000%	0,000%	0,000%
+	58,176%	0,000%	11,447%	19,002%	9,466%	13,569%	10,610%	4,093%	0,000%	0,000%	0,000%
2	57,026%	12,580%	6,419%	6,090%	8,659%	6,814%	7,961%	7,427%	5,814%	7,421%	10,178%
3	51,766%	11,151%	5,776%	5,100%	7,271%	5,880%	6,967%	6,441%	5,411%	6,833%	9,732%
4	51,361%	10,660%	6,172%	5,093%	7,004%	5,938%	6,594%	6,084%	5,379%	6,691%	10,212%
6	48,553%	9,799%	6,017%	4,689%	5,948%	5,595%	5,604%	5,468%	5,056%	6,339%	10,152%
5	47,638%	9,864%	5,843%	4,543%	6,071%	5,435%	5,831%	5,604%	5,040%	6,441%	9,836%
8	46,631%	9,328%	5,943%	4,493%	5,241%	5,382%	4,862%	5,176%	4,932%	6,120%	10,268%
7	45,270%	9,193%	5,543%	4,128%	5,111%	5,055%	4,891%	5,163%	4,781%	6,169%	9,671%
9	44,042%	8,917%	5,481%	4,156%	4,402%	4,830%	4,314%	5,046%	4,441%	6,080%	9,926%
*	30,845%	0,000%	9,709%	5,877%	5,081%	5,117%	4,822%	4,146%	0,000%	0,000%	0,000%
0	26,928%	0,000%	6,274%	2,150%	1,618%	2,670%	2,278%	2,423%	1,515%	1,975%	10,053%
/	23,448%	0,000%	2,958%	5,119%	4,361%	4,916%	5,388%	4,091%	0,000%	0,000%	0,000%
²	20,071%	0,000%	5,540%	0,927%	2,556%	3,205%	2,699%	3,127%	2,923%	0,000%	0,000%
³	12,638%	0,000%	3,665%	0,252%	1,674%	1,854%	1,883%	1,707%	1,909%	0,000%	0,000%
(	6,969%	3,951%	0,086%	1,062%	1,172%	0,480%	0,386%	0,000%	0,000%	0,000%	0,000%
)	6,969%	0,000%	0,000%	0,646%	0,806%	0,986%	1,093%	1,916%	1,691%	0,000%	0,000%

但是我們未找到有研究實作解決 Nerdle Maxi 問題的演算法，因此我們想嘗試設計演算法，並實作出來。

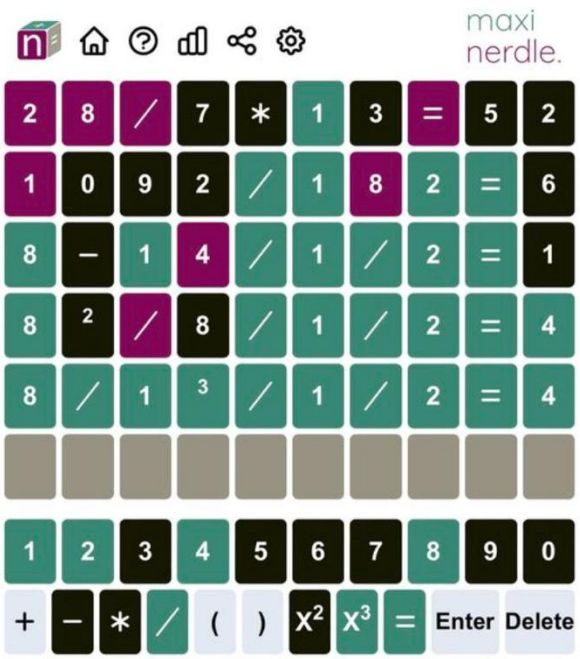
（三） Nerdle Maxi 數學邏輯遊戲介紹

1. 每個答案為十個字元組成的等式
2. 所有答案出現機率相等，在遊戲開始前已決定
3. 玩家總共有六次猜測機會，每次可以猜測一個長度為十個字元的等式
4. 可用字元： 0 1 2 3 4 5 6 7 8 9 + - * / =( ) ² ³
5. 答案一定包含一個「=」
6. 「=」右邊必須是個數字，不能是其他算式
7. 算式會依循算則，也就是「先乘除後加減」
8. 輸入猜測後，每格會變成該數字對應的顏色
9. 綠色：「字元」和位置都正確

10. 紫色：有這個「字元」但位置不對
11. 黑色：沒有這個「字元」
12. 第一個字元不使用零且沒有單獨的零，即使這些可能被接受為有效的猜測。因此，不會有這樣的答案：*0+5+5+5=15*，或是像這樣的答案：*01+2+1+1=5*，又或是像這樣的答案：*5+5+5+0=15* 然而，*0* 可以出現在等號右邊作為答案（例如，*10-5-3-2=0* 是被允許的）



圖一、回饋顏色示意圖（第二創作者製作）



圖二、Nerdle Maxi 遊戲示意圖（第二創作者製作）

（四）資訊量的量化

1. 資訊量公式： $Q = -k \log P$
2. P 為事件發生的機率， k 為常數
3. 期望獲得的資訊量（即資訊熵） $S = -k \sum_i P_i \log P_i$

貳、研究設備及器材

一、硬體設備

（一）RunPod GPU Cloud: NVIDIA L40S * 4 (48GB VRAM, 62GB RAM, 12 vCPUs)

（二） Intel® Core™ i5-8400

（三） Apple M1

二、軟體

（一）文字編輯 Visual Studio Code 1.100

（二） C++ Compiler (GNU GCC)

（三） NVIDIA CUDA Compiler (Driver NVCC 12.8)

（四） Python v2024.14.1

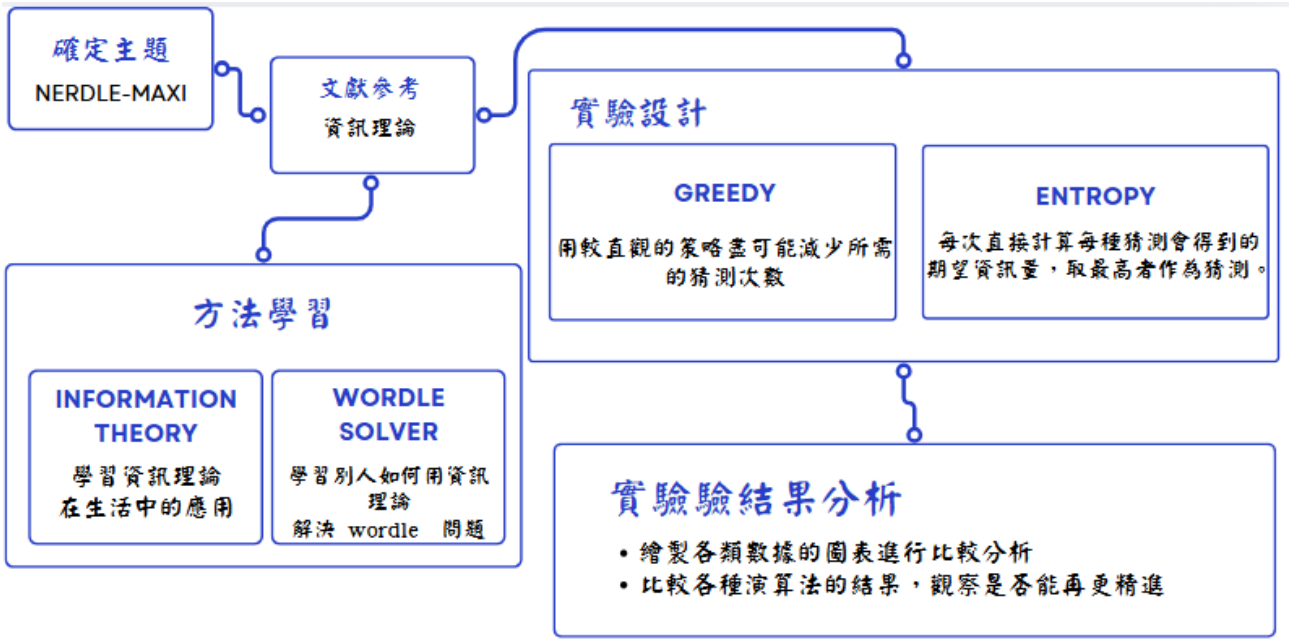
（五）作業系統 Windows 11 （64 位元作業系統，x64 型處理器）

（六）macOS Ventura Version 13.6.6 (22G630)

參、研究過程或方法

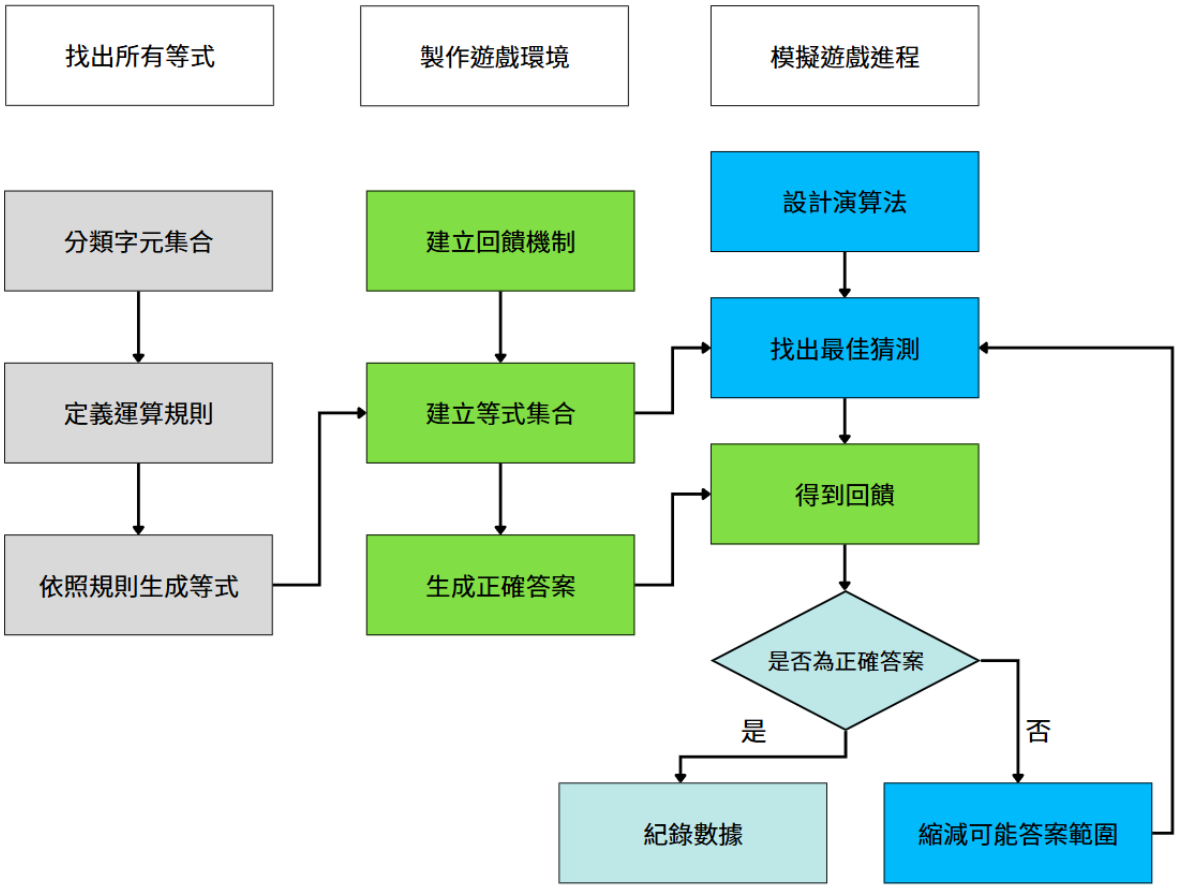
一、研究及實驗架構流程圖

（一）研究架構



圖三、研究架構圖（第三創作者製作）

（二）實驗流程圖



圖四、實驗流程圖（第三創作者製作）

二、找出所有符合遊戲規則之等式

只需生成等式左側。因為生成後，只需解析並計算其結果，檢查生成的等式是否符合 10 個字元的長度限制。左式還需小於或等於 8 個字元，因為至少需要 2 個字元用於等號和右式。左式是由左而右一格一格生成的，每添加一個字元時，下一個可添加的字元取決於上一個字元。

（一）考慮以下符號集合

1. 數字集合 $N = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
2. 運算符集合 $O = \{ -, +, *, / \}$
3. 次方符號集合 $F = \{ ^2, ^3 \}$
4. 左括號集合 $U = \{ (\}$
5. 右括號集合 $B = \{) \}$

（二）根據上一個字元的類型，決定接下來可以接的類型

1. 如果沒有上一個符號（表達式的開頭），則下一個字元可以是集合 N 、 U
2. 如果左側字元屬於集合 D，則下一個字元可以是集合 N、O、E
3. 如果左側字元屬於集合 O ，則下一個字元可以是集合 N、U
4. 如果左側字元屬於集合 E ，則下一個字元只能是集合 O
5. 如果左側字元屬於集合 U ，則下一個字元可以是集合 N、U 或負號（-）
6. 如果左側字元屬於集合 B ，則下一個字元只能是集合 O、E

使用此方法，在任何步驟中，如果沒有開啟的括號，則表達式始終有效。只需計算其結果是否為非負整數，且生成的等式是否符合所需長度。

由於此方法生成的每個表達式都是有效的，可以在生成時即計算表達式的值，更有效率的選擇下一個可能的符號。例如，當前表達式為「152^3」時，下一個運算子必為乘法或除法，因為其他運算子將使結果無法符合 10 個字元的限制。

枚舉所有可能的等式後，我們設計了五種不同的啟發式演算法，將其分別命名為 greedy1、greedy2、entropy1、entropy2、entropy3。

三、greedy1 演算法

定義 *cal_green*(*w*) 為，設等式 *w* 為猜測，枚舉所有可能的答案等式，並計算對應之回饋後，計算回饋中平均有多少格綠色（位置正確且符號或數字正確）。此演算法進行猜測時，先計算出所有可能答案等式的 *cal_green*() 值，並使用最大者作為猜測。

此方法主要是要模擬人類遊玩的直覺策略，也就是在每次猜測時，盡可能讓更多格子是正確的，並與其他的方法比對。

四、greedy2 演算法

進一步思考如何最小化花費的猜測次數時，發現應該要模擬所有可能答案，並利用所得到的回饋模擬縮減可能為答案的集合獲得線索。該演算法會在進行每次猜測時，目標是選擇可以平均讓答案可能數減少最多的等式，但由於使用平均可能會選中剩餘答案數標準差非常大之等式，因此最後決定選擇可以最小化剩餘可能答案的最大值（minimizing the maximum）之猜測 。

討論此演算法的過程中，我們認為此方法或許考慮的不夠全面，開始思考該如何量化猜測的回饋給的線索的價值。

五、entropy1 演算法

探討了其他有關 Wordle 的研究後，我們發現很多研究都是使用資訊熵為核心的演算法，每次會找出一個可以期望獲得最多資訊量的方法。

在每一輪的猜測時，會對於所有目前還可能是答案的等式（對於之前的猜測與回饋合法），計算此等式期望會獲得多少資訊量，最後以期望會獲得最多資訊量的等式作為猜測。

具體來說，定義 T、W 為兩個任意符合遊戲規則的字串，r 為回饋字串（由綠、紅、黑三種顏色組成）集合 $legal$ 為所有符合之前猜測的字串（可能為答案的字串）定義 $f(W,T)$ 為把 W 當作猜測，T 作為答案，的回饋字串。定義 $count(W,r)$ 為將 W 作為猜測，對於所有 T 屬於 $legal$，並且 $r = f(W,T)$，的數量。

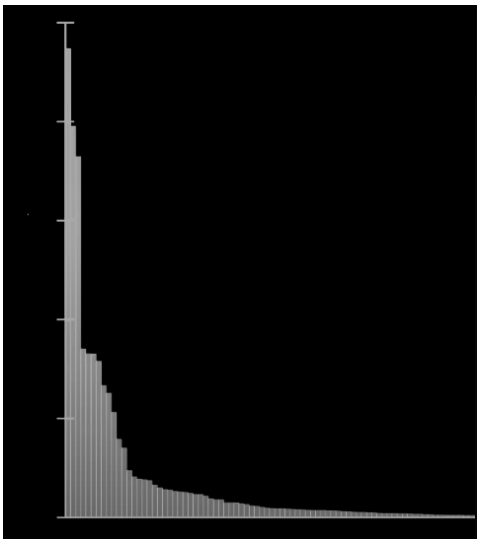
假設將 W 作為猜測問出，系統給的回饋字串為 r，答案可能的範圍，也就是 $legal$，就縮減成那些 $f(W,T) = r$ 的 T，答案可能種類數會從 $|legal|$ 變為 $count(W,r)$。因此根據資訊量定義，獲得了式一的資訊量，又因為回饋字串為 r 的機率為 ，因此可以用期望值公式算出式二，也就是以 W 作為猜測，加總所有情況獲得的資訊量乘以發生之機率，可以得到 W 的期望資訊量，因此我們會把 $P(W)$ 最大之等式作為猜測。

式一：$$\log_2\left(\frac{|legal|}{count(W,r)}\right)$$

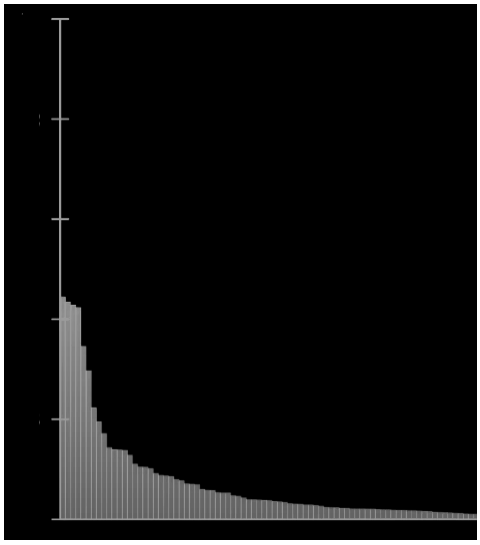
式二：$$P(W) = \sum_r \frac{count(W,r)}{|legal|} \cdot \log_2\left(\frac{|legal|}{count(W,r)}\right)$$

換句話說，從分佈圖的角度來看，由於答案由 10 個字元組成，因此回饋種類數有 3^{10} 種。計算這些可能回饋的分佈，如果這種方法將有利於猜測，則該猜測所獲得的每種回饋的機率應該有平坦的分布 。

以下有兩張示意圖，代表兩種不一樣的猜測獲得的回饋分佈圖，左到右分別稱作 A、B 兩圖，橫軸代表不同種類的回饋，縱軸代表對應回饋出現的次數。



圖五、回饋分布圖 A（第一創作者製作）



圖六、回饋分布圖 B（第一創作者製作）

圖 A 中，整體分布的不平坦，有些回饋的出現次數非常多，出現機率較高。當回饋是出現次數較多的回饋種類時，刪減後的 *legal* 會較大，也就是刪減的量較少，從而獲得較少的資訊量。因此圖 A 的猜測獲得的期望資訊量會比圖 B 的猜測差。

從上面的例子可以發現，回饋種類數量分布較為平坦的猜測，能獲得更多資訊量。因此二猜測中，圖 B 的猜測為較好的猜測。

六、entropy2 演算法

網路上大部分的研究都是以期望資訊量最大之猜測會出現在可能答案集合中為前提（或者加設差距非常小），並沒有實際將所有合法等式都列入猜測考慮，因此我們想要在實驗後了解此方法實際的表現會如何。

在每一輪的猜測時，會對於所有等式，計算此等式期望會獲得多少資訊量，最後以期望會獲得最多資訊量的等式作為猜測。

詳細算法與 entropy1 演算法相同差別在於 *legal* 集合換成以所有可能等式作計算，我們堆測這樣做可以在每次猜測時獲得更多資訊。因為原本如果有已經正確的格子（綠色），用 entropy1 時，下個猜測會繼承上面的綠色，有可能會浪費掉可以得到資訊的機會。

由於此方法所使用的計算量極大（是 entropy1 的千倍以上），我們將原本程式結合了 NVIDIA CUDA，藉由平行演算法與 GPU 的運算能力，將原本要跑一年以上的程式加速至 1 ~ 2 日就可以算完。

七、entropy3 演算法

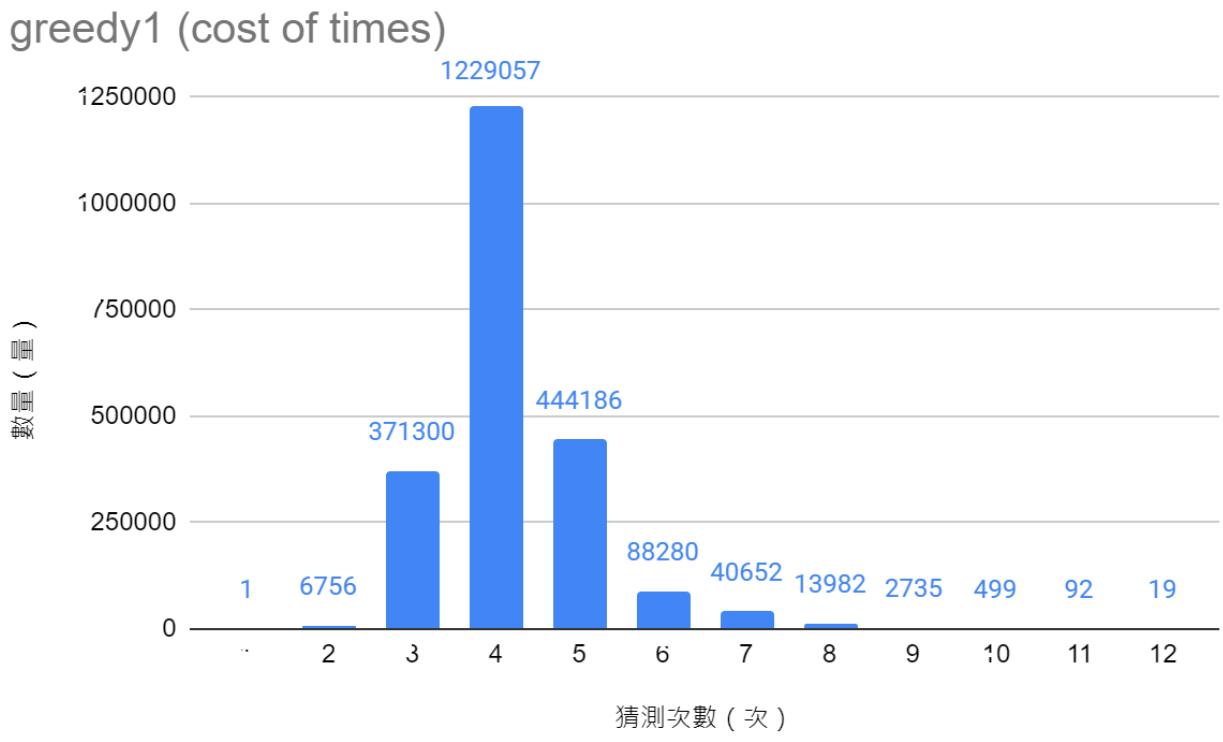
在 entropy2 實驗過後我們發現此演算法有缺陷，經常在第 3、4 次猜測時，即便將可能答案範圍縮減到個位數，還是猜不到正確答案，導致平均猜測次數較多。

但根據我們的推測，entropy2 可以在 1、2 次猜測將可能答案範圍縮至更小，因此我們想到融合 entropy1、entropy2 兩種方法，設計出了 entropy3：使用 entropy2 計算前兩次猜測，從第三次猜測開始，使用 entropy1。

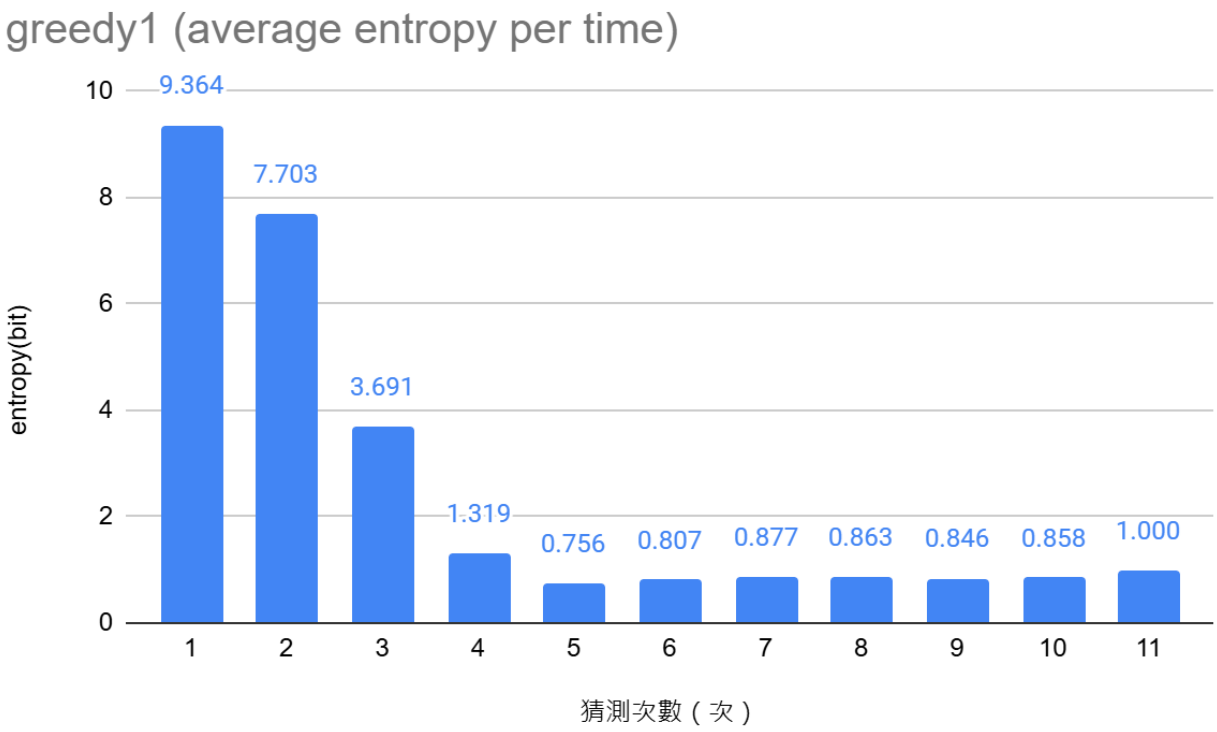
肆、研究結果

根據 Nerdle Maxi 的規則，我們生成了總共 2197559 種可能的等式。

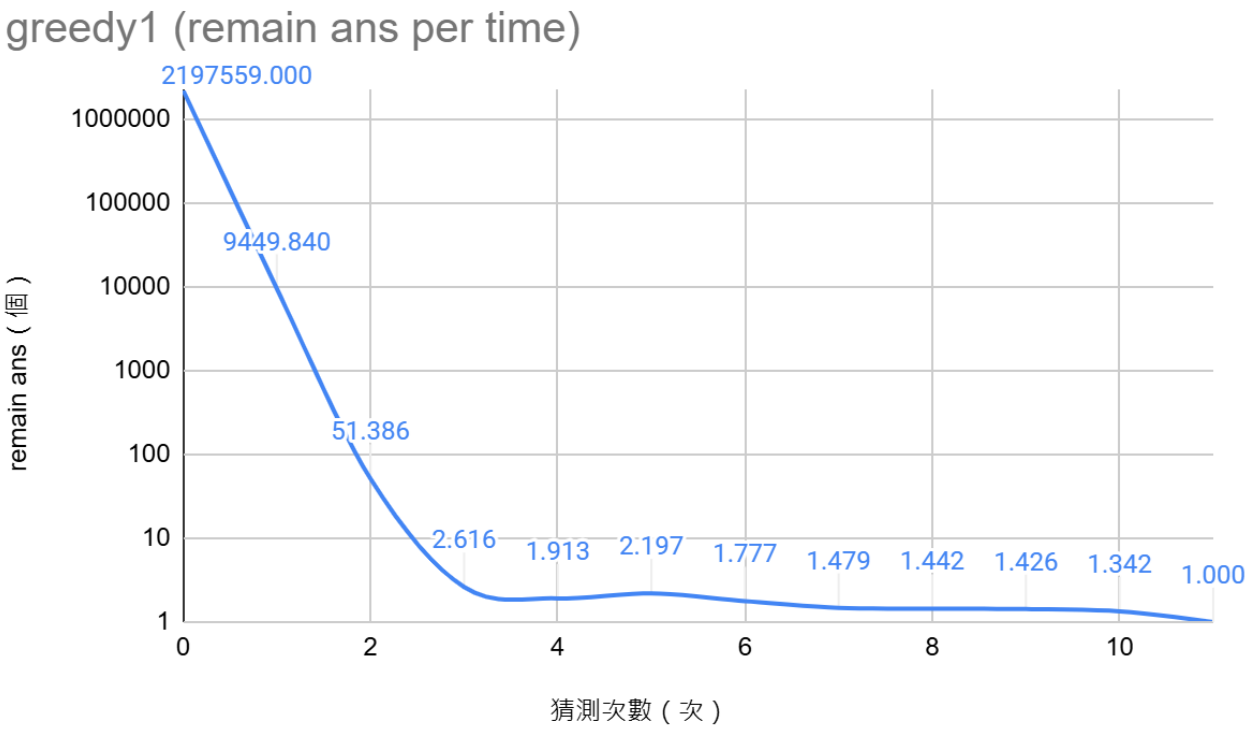
一、 greedy1 演算法



圖七、greedy1 所有答案情況猜測所需次數長條圖（第三作者製作）

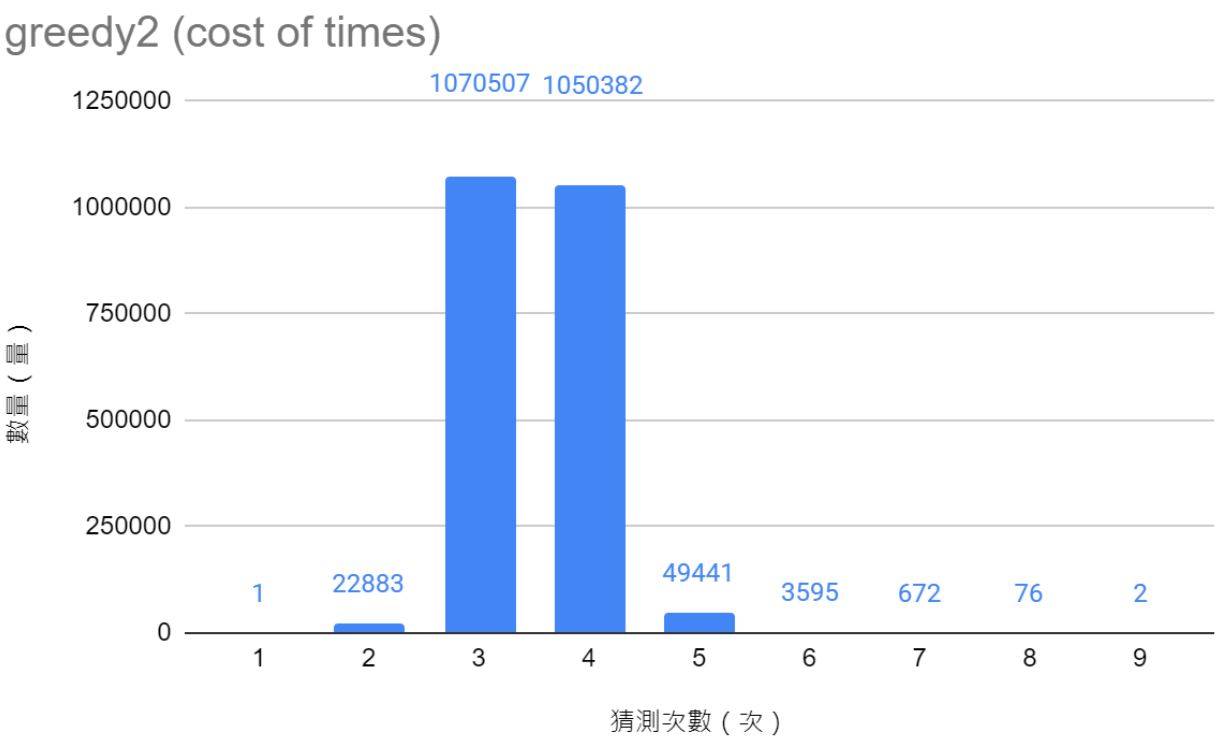


圖八、greedy1 平均每次猜測獲得資訊熵長條圖（第三作者製作）

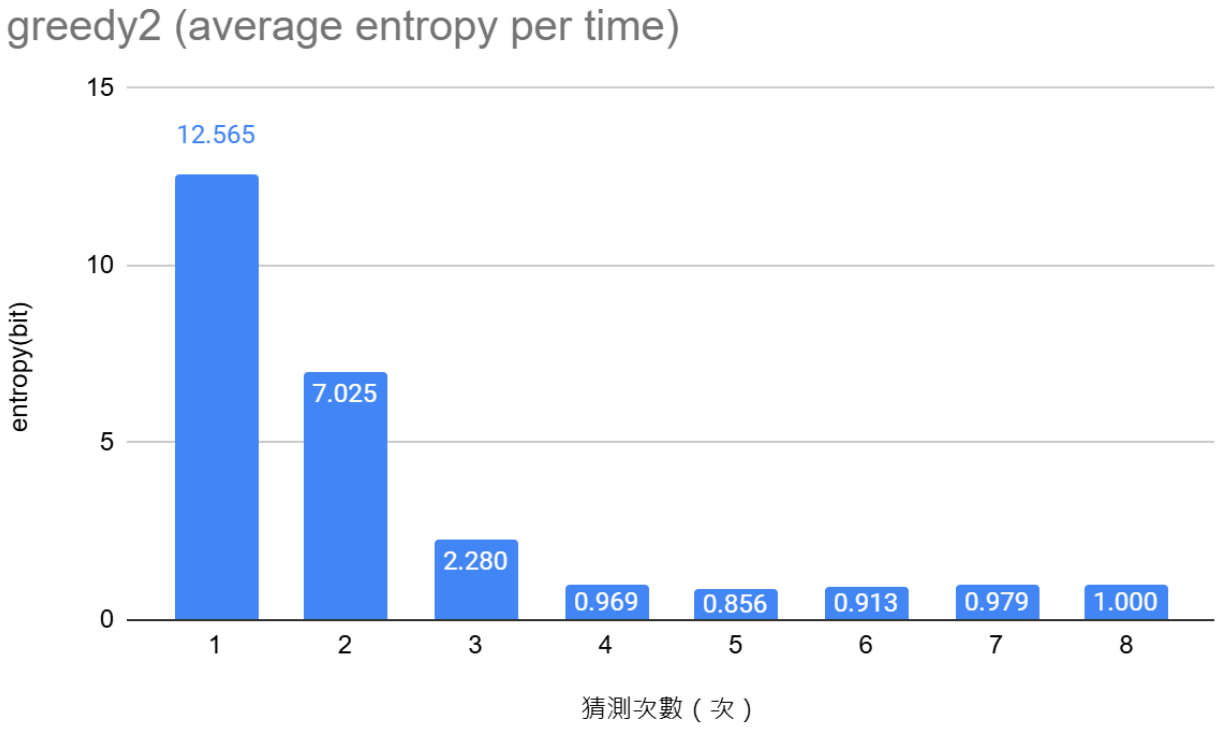


圖九、greedy1 平均每次猜測完剩餘答案數量長條圖（第三作者製作）

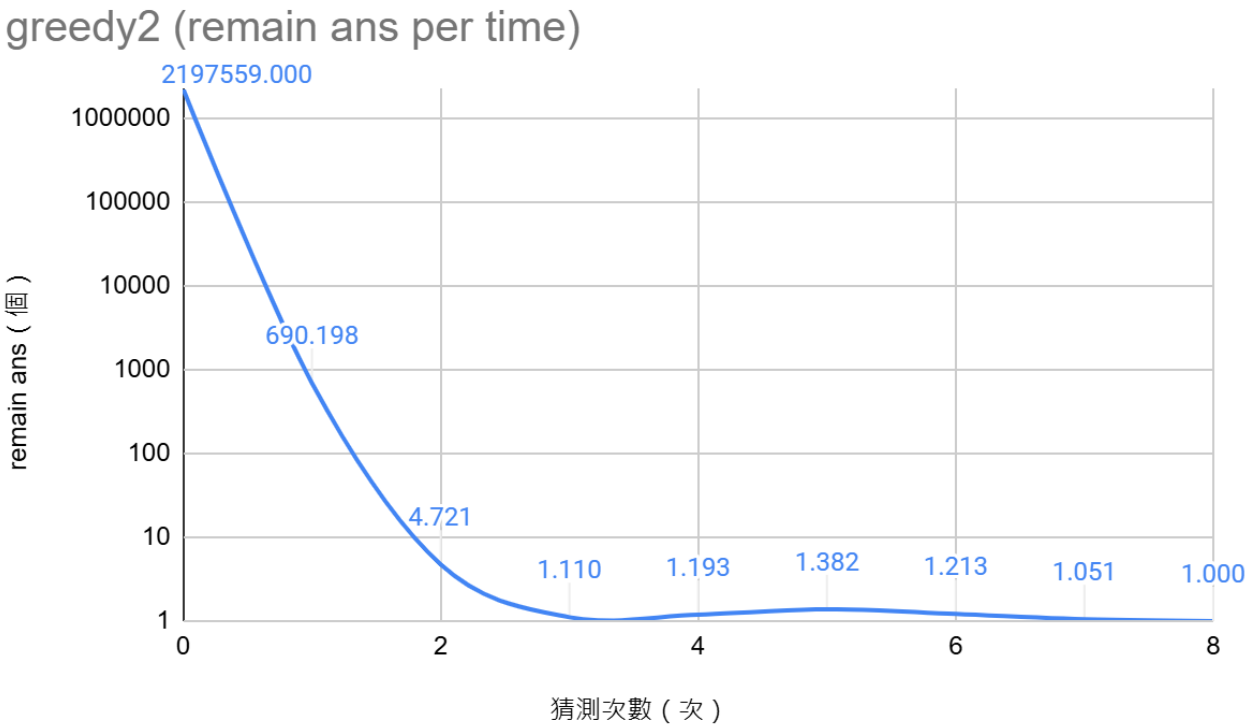
二、 greedy2 演算法



圖十、greedy2 所有答案情況猜測所需次數長條圖（第三作者製作）

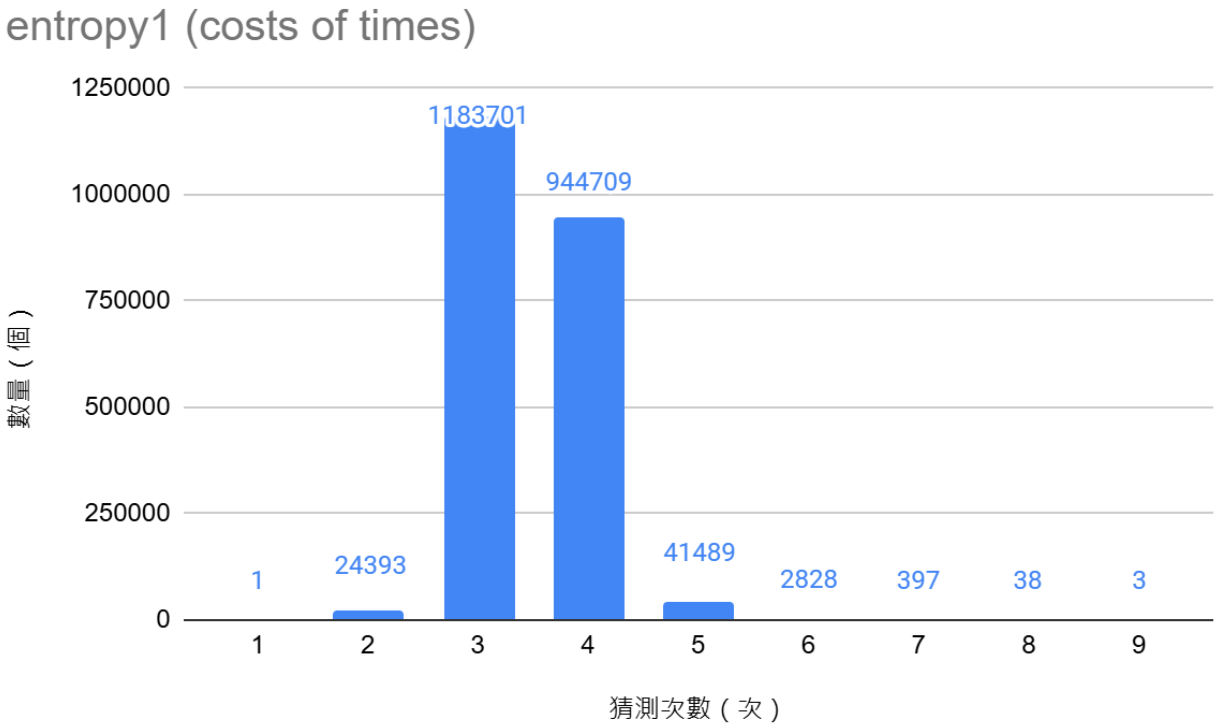


圖十一、greedy2 平均每次猜測獲得資訊熵長條圖（第三作者製作）

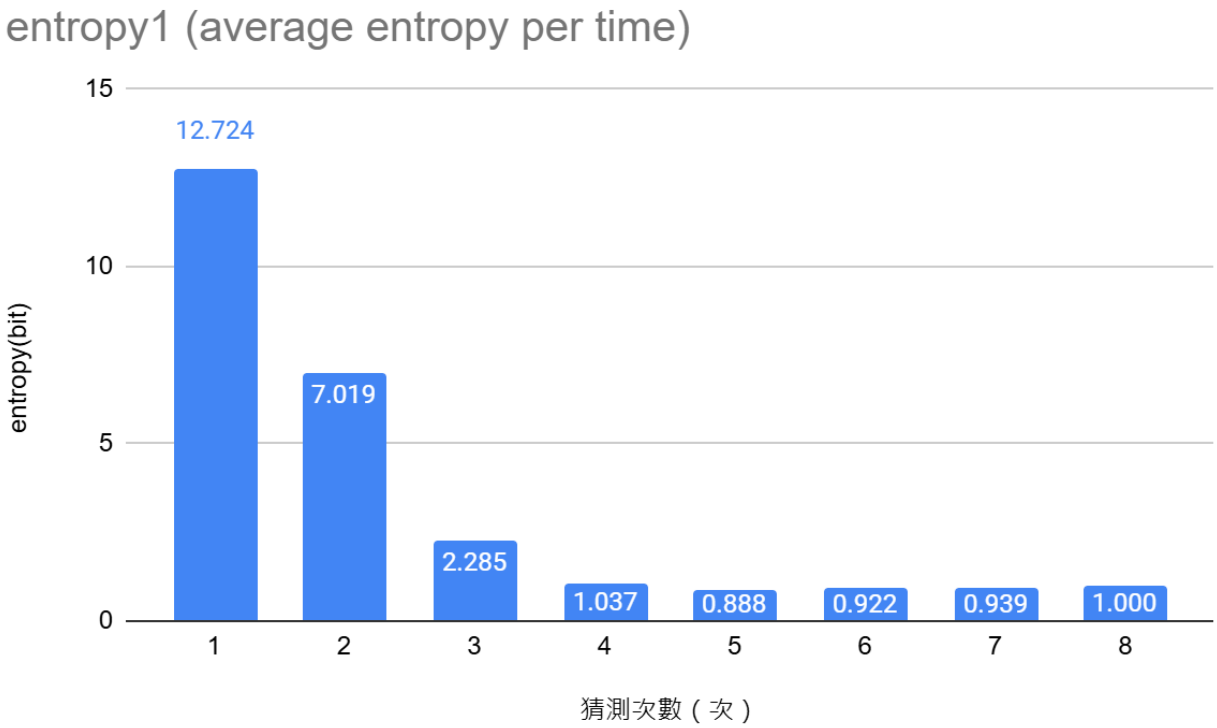


圖十二、greedy2 平均每次猜測完剩餘答案數量長條圖（第三作者製作）

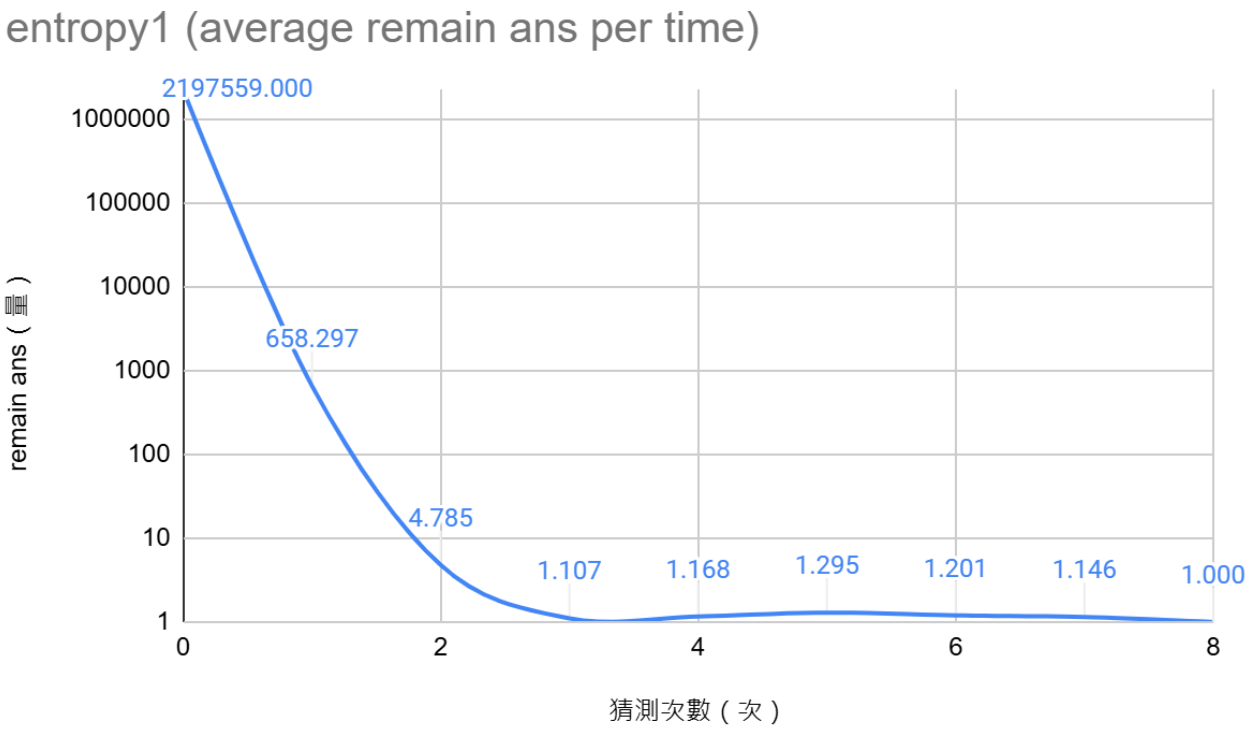
三、 entropy1 演算法



圖十三、entropy1 所有答案情況猜測所需次數長條圖（第三作者製作）

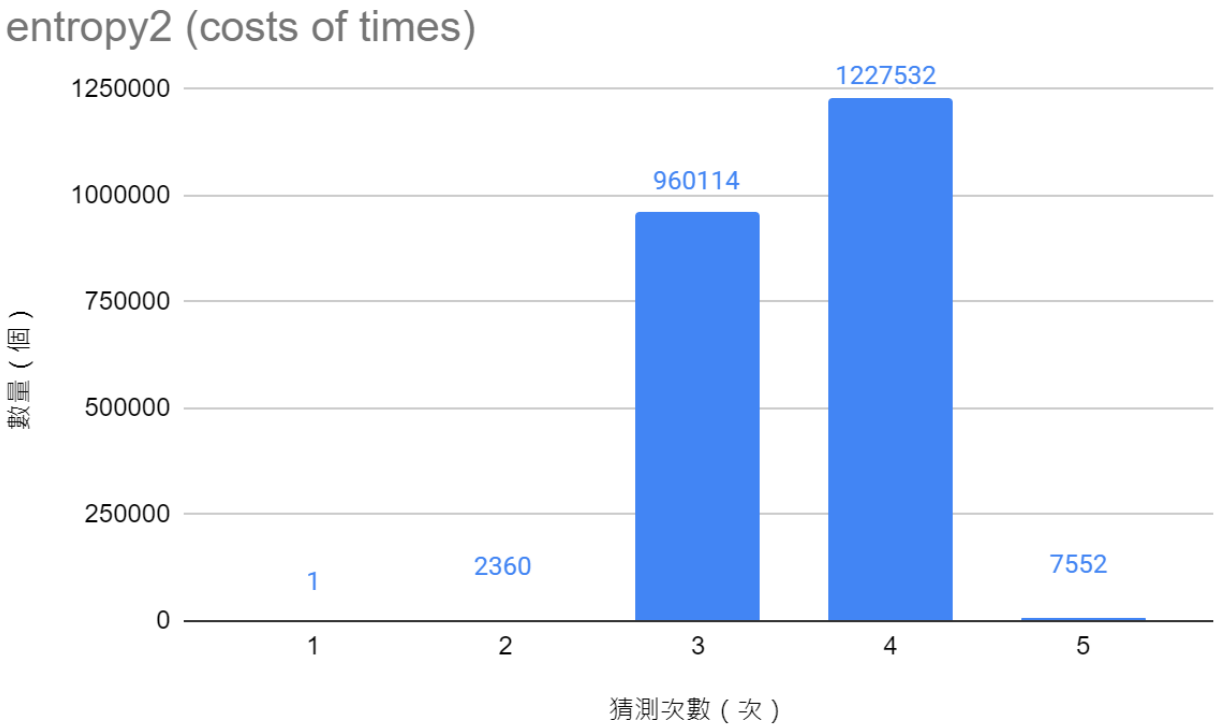


圖十四、entropy1 平均每次猜測獲得資訊熵長條圖（第三作者製作）

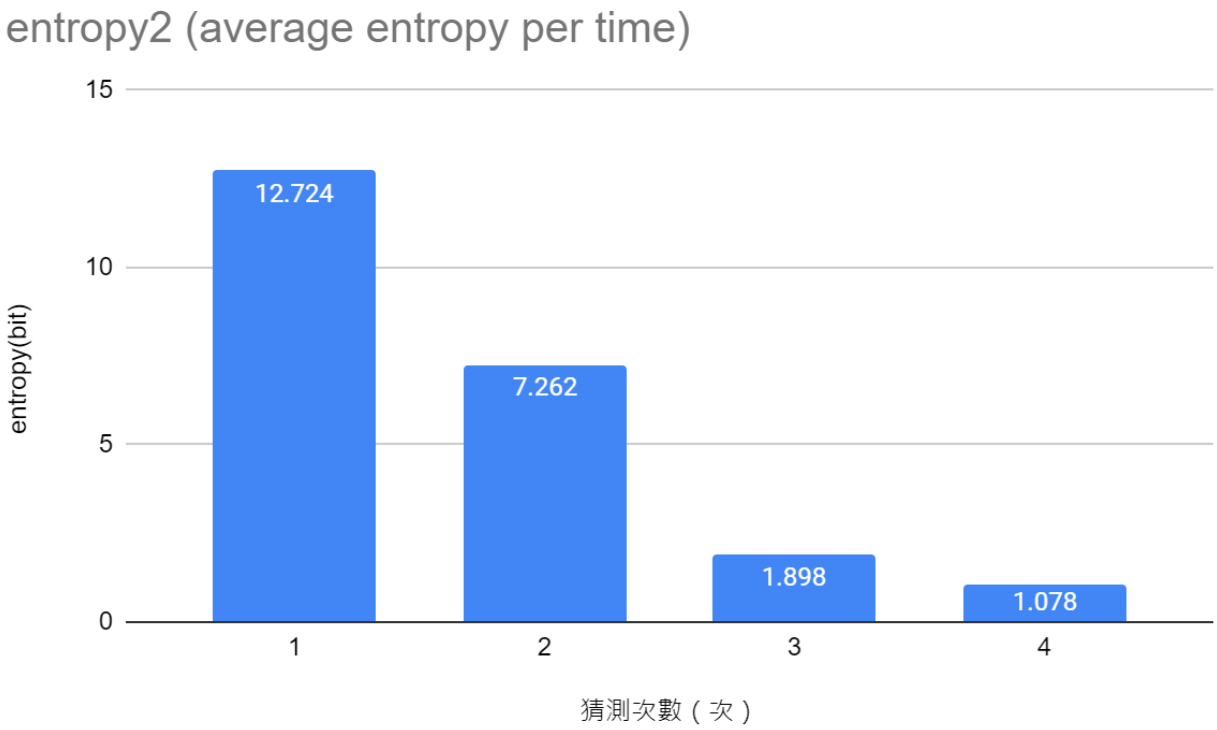


圖十五、entropy1 平均每次猜測完剩餘答案數量長條圖（第三作者製作）

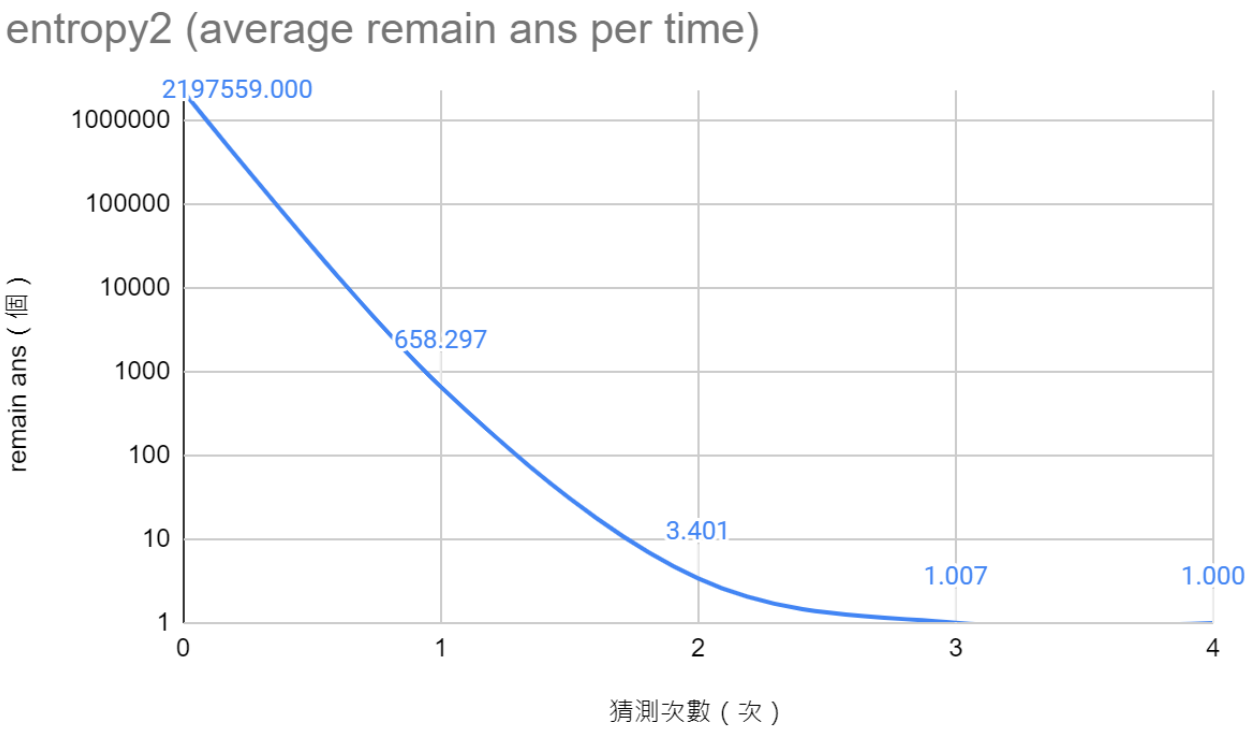
四、 entropy2 演算法



圖十六、entropy2 所有答案情況猜測所需次數長條圖（第三作者製作）

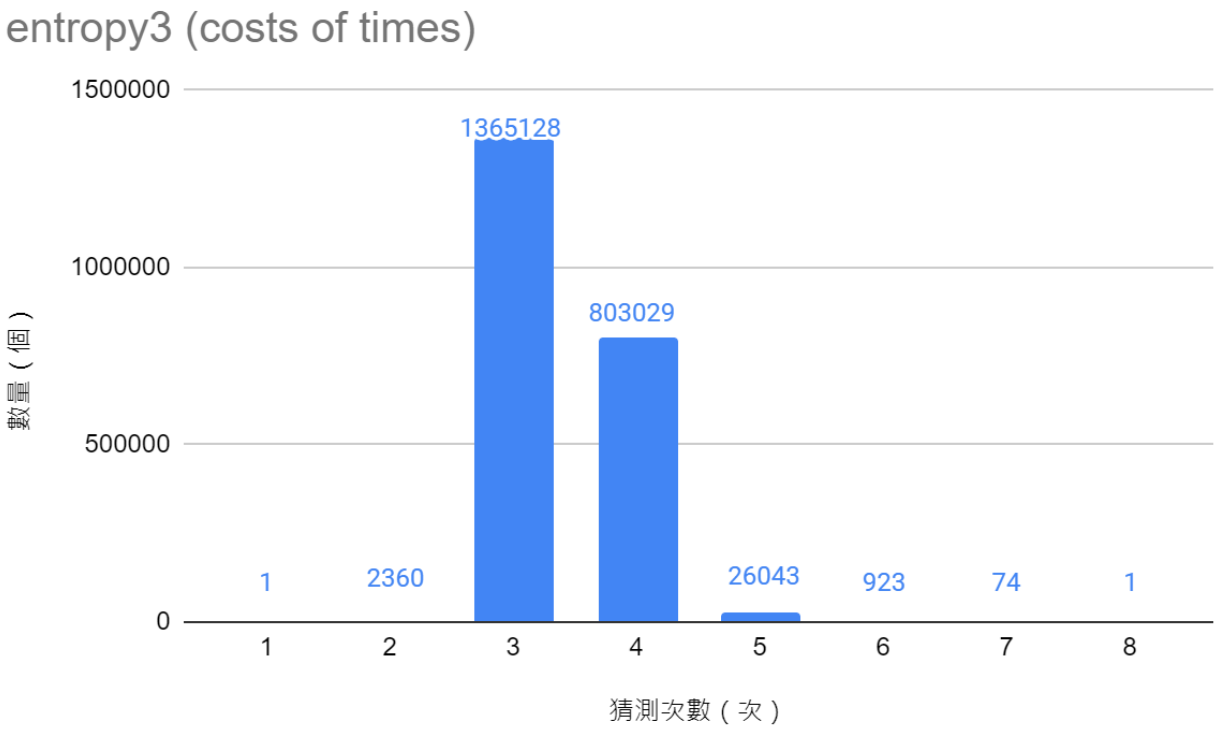


圖十七、entropy2 平均每次猜測獲得資訊熵長條圖（第三作者製作）

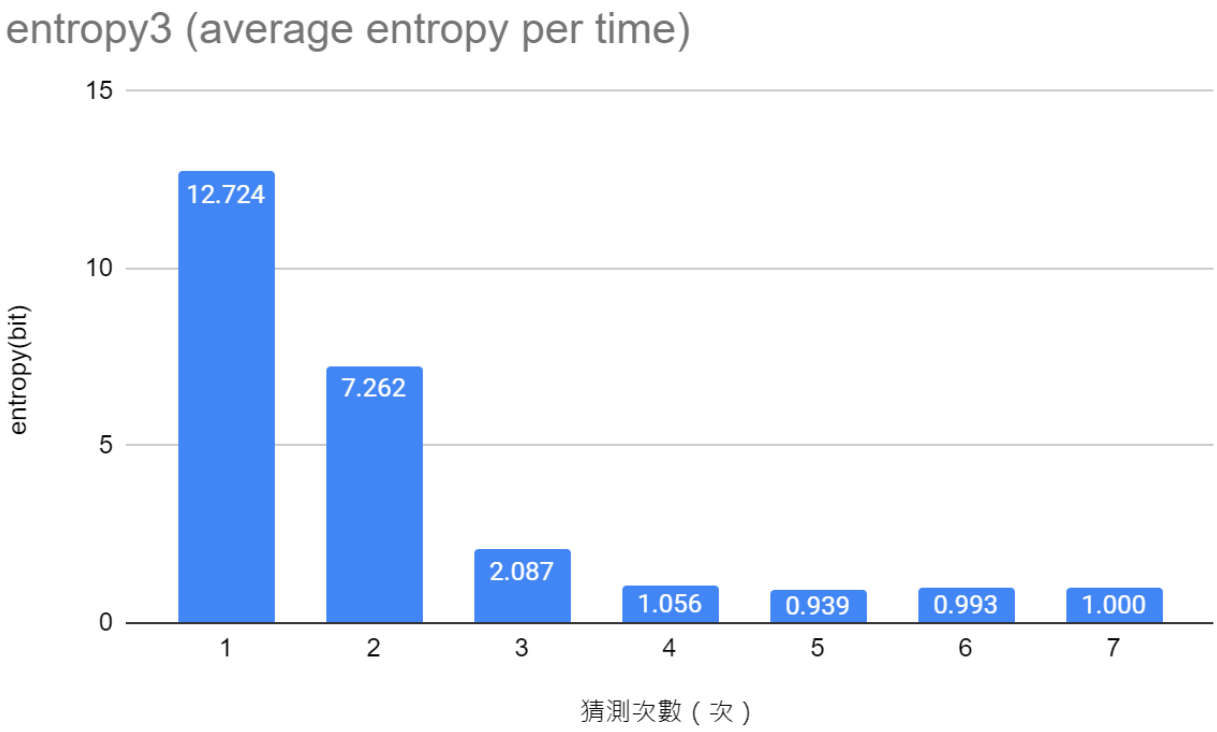


圖十八、entropy2 平均每次猜測完剩餘答案數量長條圖（第三作者製作）

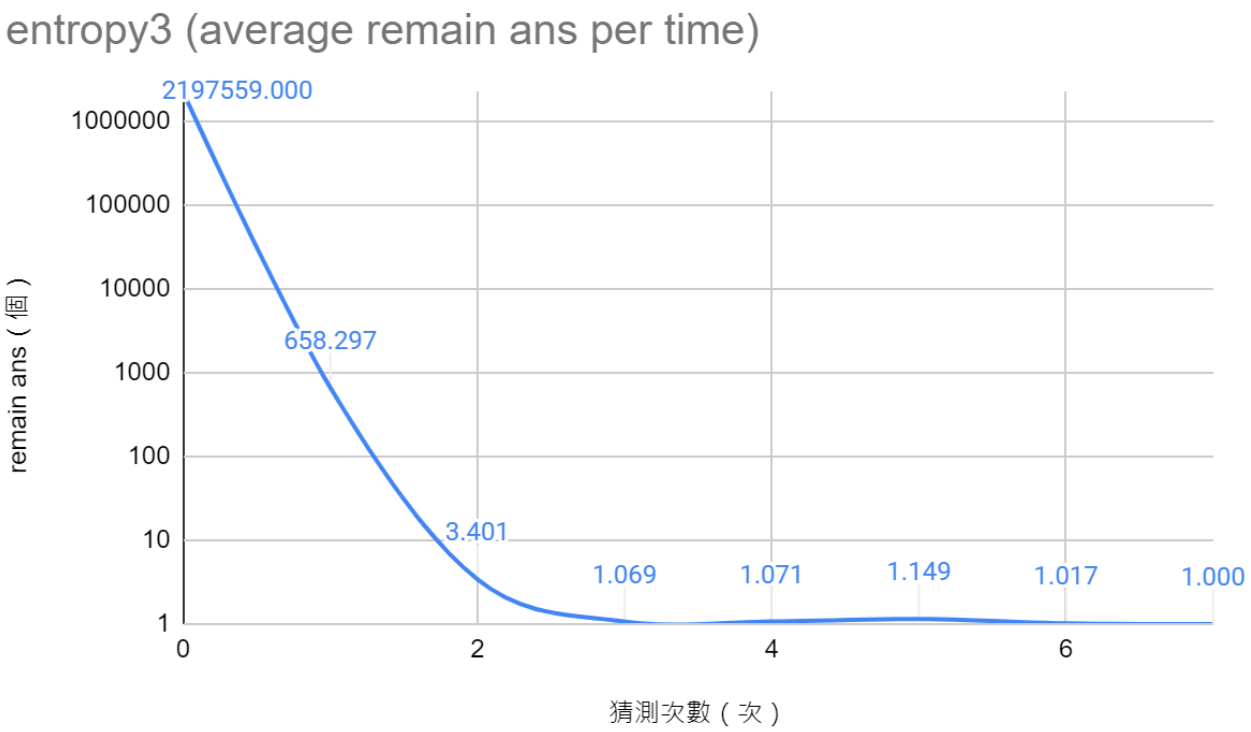
五、 entropy3 演算法



圖十九、entropy3 所有答案情況猜測所需次數長條圖（第三作者製作）



圖二十、entropy3 平均每次猜測獲得資訊熵長條圖（第三作者製作）



圖二十一、entropy3 平均每次猜測完剩餘答案數量長條圖（第三作者製作）

伍、討論

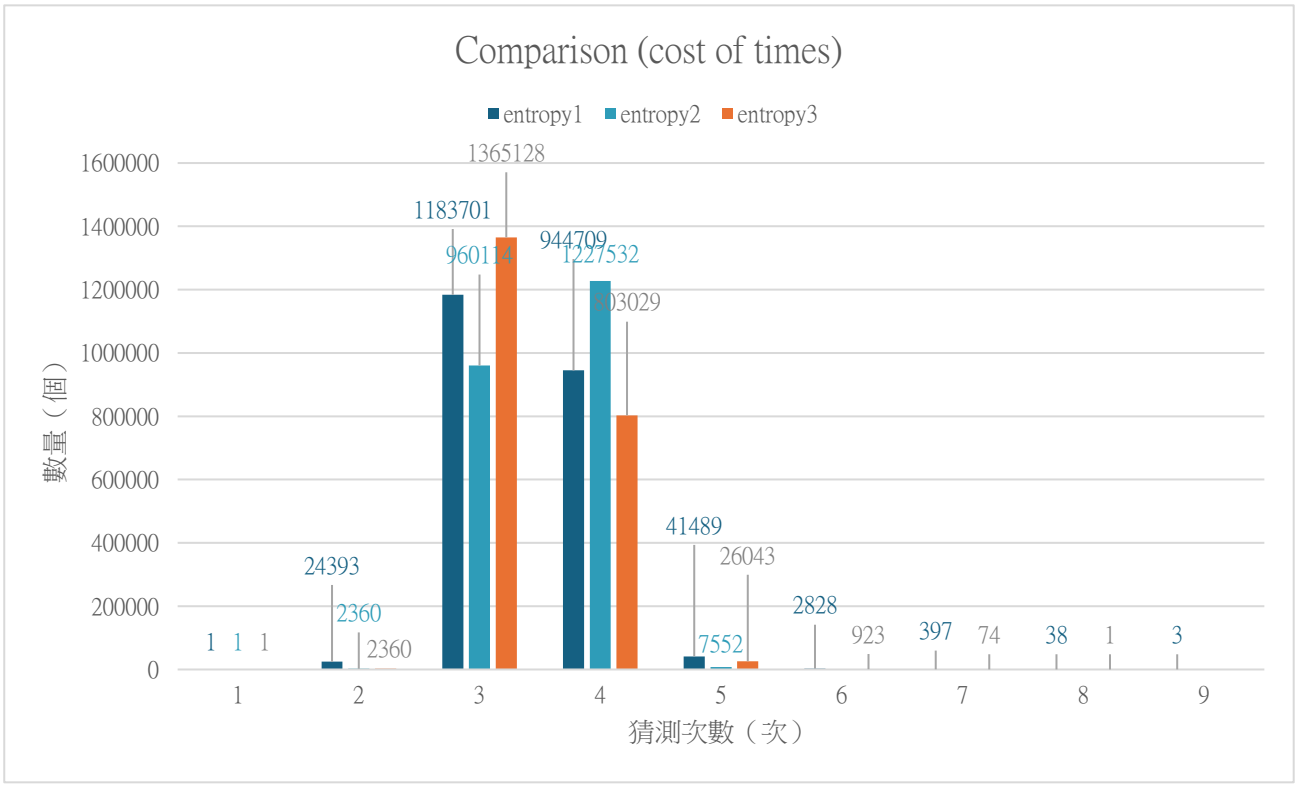
一、不同方法花費猜測次數之結果比較

表二、各演算法猜測次數綜合比較（第一創作者製作）

	greedy1	greedy2	entropy1	entropy2	entropy3
算數平均 （花費猜測次數）	4.20	3.52	3.46	3.56	3.38
標準差	0.90	0.57	0.56	0.50	0.52
眾數（花費猜測次數）	4	3	3	4	3
最差情況（花費猜測次數）	12	9	9	5	8

依據我們的實驗結果 entropy2、entropy3 各有優缺點，使用哪一種方法應該視要達成的目標而定。從圖表中可以看出，若以平均次數作為比較基準，entropy3 的表現最優，但如果以確保能在限制次數內猜中答案，則應該選擇 entropy2 較優。

二、entropy 方法之結果討論



圖二十二、entropy 系列方法猜測次數分布長條圖（第三創作者製作）

我們以資訊熵為核心的方法有 entropy1、entropy2、entropy3。從平均花費次數來看，entropy2 其實是三者中最差的，但從花費次數的分布可以看到 entropy2 最多只使用 5 次便猜中答案，而且 entropy2 縮減可能答案範圍的效果也較好， 在第二次猜測完畢後，entropy1 剩餘答案可能數量平均為 4.78，而 entropy2 是 3.40（entropy3 在第二次猜測的演算法與 entropy2 相同）。

我們認為 entrop2 的平均花費次數會比 entrop1 多是因為， entropy2 的猜測範圍是全部可能等式，所以只有在當 *legal* 集合大小縮減至 1 後，才能在下一次猜測中猜出（直接猜中答案機率非常小），對比原本的 entropy1，猜測範圍就是 *legal* 集合，因此在 *legal* 集合大小夠小時，就有較高機率猜中答案。因此 entropy2 的結果才會都是在五次內猜中，但平均花費次數最多。

entropy3 是基於 entropy1，融合 entropy2 優點後的改良版本，它符合我們預期地，平均與最差情況均使用更少次數的猜測，就猜到答案。

綜上所述，如果目標是能保證解決 Nerdle Maxi 問題，我們建議使用 entropy2 演算法，而如果目標是能在平均上用最少次數猜中答案，我們建議使用 entropy3 演算法。

三、不同種類算法的時間複雜度分析與實際運行時間

下表為令 G 為平均第一次猜測的剩餘答案數量，N 為合法等式種類數，L 為等式長度，比較不同演算法的時間複雜度、實際執行時間以及硬體。其中 entropy2、entropy3 兩種演算法執行時間由 RunPod 估算，不代表實際執行時間。

表三、 不同演算法的時間複雜度、實際執行時間、硬體比較（第一創作者製作）

	greedy1	greedy2	entropy1	entropy2	entropy3
時間複雜度	$O(NGL)$	$O(NG(GL))$	$O(NG(GL))$	$O(NN(GL))$	$O(NN(GL))$
運行硬體	30 CPU thread	30 CPU thread	30 CPU thread	RunPod GPU L40S*4	RunPod GPU L40S*4
實際運行時間 (小時)	1.06	3.58	10.83	約 55.03	約 28.96

陸、結論

一、研究成果

本研究利用 CUDA，以顯卡加速，實作出了尚未有人嘗試的猜測方式（entropy2），保證玩家一定能在遊戲限制次數（六次）內猜出正確答案。

經過不同的演算法比較後，發現 entropy2 演算法的優點與缺點，並將新的方法融合原本的演算法（entropy1），最終模擬所有可能答案後，新演算法（entropy3）平均只需花費 3.38 次猜測即可猜出答案。

二、未來展望

（一） 答案出現機率不同的 Nerdle Maxi

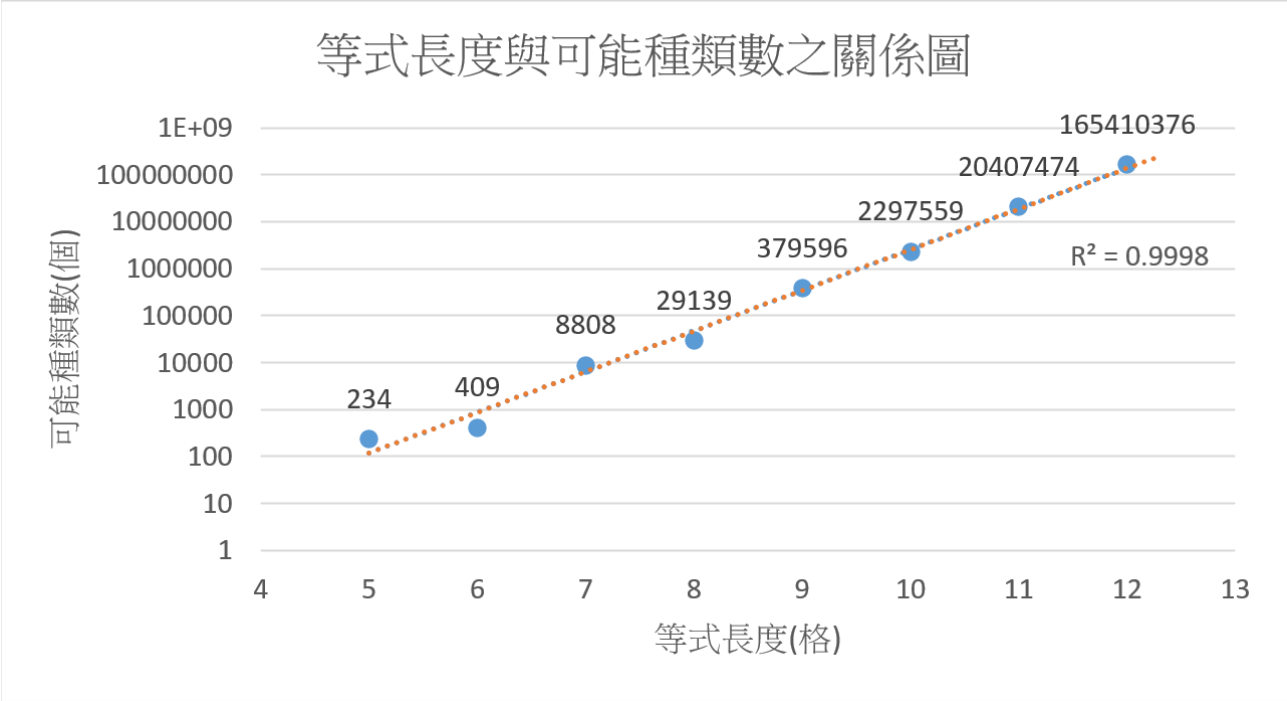
當等式的出現機率不相等時，總熵無法用我們原本的運算方法。這種情況下，需考慮不等機率下的熵分解公式。

1. 假設整體等式集合 W 每個猜測 A 為答案的機率為 $P(A)$
2. 可能為答案的集合為 $W+$ 且其出現的總率為 $w^+ = \sum p(w)$
3. 不可能為答案的集合為 $W-$ 且其出現的總率為 $w^- = \sum p(w)$
4. 可能為答案的集合的熵 = $H^+ = \sum \left(\frac{p(w)}{w^+} \log_2 \left(\frac{w^+}{p(w)} \right) \right)$
5. 不可能為答案的集合的熵 = $H^- = \sum \left(\frac{p(w)}{w^-} \log_2 \left(\frac{w^-}{p(w)} \right) \right)$ ：
6. 總熵 $H(W)$ 可表示為：

$$H(W) = w^+H^+ + w^-H^- + w^+ \log_2 \left(\frac{1}{w^+} \right) + w^- \log_2 \left(\frac{1}{w^-} \right)$$

（二）事件數量大小過大問題

討論事件數量時，我們發現當 Nerdle Maxi 格子總數過大時，可能等式的數量會上升到難以在有限時間內計算完畢，導致我們的演算法無法正常運行。下圖是以 5~12 作為可用的字元數生成等式來估算可能等式數量成長的趨勢。



圖二十三、等式長度與可能答案種類數關係圖與回歸線（第二作者製作）

漸進線方程式為 $y \approx 0.0055 \times 7.3566^{x}$ ，其中 x、y 分別為等式長度、合法等式種類數

我們查到在這種情況下，可使用另一種運用資訊熵的方法： Bayesian Optimization (Donald R. Jones Matthias, Schonlau, & William J. Welch 1998)

此方法會利用到以下三種概念：

1. Black-Box Function(Franz Breisig, 1924)：指的是一種你只能以「輸入→黑 Black-Box Function→輸出」的方式來使用、但無法直接取得其內部結構或解析式的函數。
2. Surrogate Model ：進行反算以降低運算時間。Surrogate Model 建立使用模式的方法為先訓練、再驗證、最後代理真實模型。大部份使用 Gaussian Process, GP 作為 Surrogate Model。

3. MES Acquisition Function：Acquisition Function 根據目前 Surrogate Model 對不同 的預測均值及不確定度來評分每一個候選點，決定下一輪最值得的輸

入。才能用最少的試算次數獲得最多資訊或最快提升目標值。其中，The max value entropy search (MES)是一種基於資訊理論的選擇策略，其核心目的是去選擇那些目前對結果影響最大、但還不確定的決策，而最小化我們對最優結果的選擇。也就是說，MES Acquisition Function 會挑選其計算方法下資訊熵最大的決策。

Bayesian Optimization 能在有限次的函數評估下（可能價格昂貴或實驗時間需要極長或儘可能少的查詢次數內找到最大值），藉由 Surrogate Model ＋Acquisition Function 重覆計算與更新，找到 Black-Box Function 中最能影響結果的變因：

「初始隨機探索 → Surrogate Model & MES Acquisition Function → Black-Box Function（實際實驗）→ 更新 Surrogate Model 」。

綜上所述，即便在 Nerdle Maxi 的格子數量過大的情況下，透過 Bayesian Optimization 與 MES Acquisition Function，依然能夠透過有限次猜測，每次選擇資訊熵最大的猜測，藉由回饋調整 Surrogate Model，最終猜到正確答案。

柒、參考文獻資料

[1] 3B1B. (2022, February 6). *Solving Wordle Using Information Theory.* YouTube. https://www.youtube.com/watch?v=v68zYyaEmEA

[2] Markowsky, G. (2024, April 12). *Entropy in Information Theory in Classical Information Theory*.Britannica. https://www.britannica.com/science/information-theory/Entropy

[3] Marcon, E. (2022, December 3). *Partitioning Information in 3B1B Wordle Video.* Github. https://ericmarcon.github.io/3B1B-Wordle/3B1B-Wordle.pdf

[4] Lokshtanov, D., & Subercaseaux, B. (2021, May 14). *Wordle Is NP-Hard.* ArXiv. https://arxiv.org/abs/2203.16713

[5] Rosenbaum, W. (2021, May 8). *Finding a Winning Strategy for Wordle Is NP-Complete.* ArXiv.https://arxiv.org/abs/2204.04104

[6] Botorch. (2019, November 6). *The Max-Value Entropy Search Acquisition Function.* BoTorch. https://botorch.org/docs/tutorials/max_value_entropy/

[7] Kenton, W. (2024, February 2). *What Is a Black Box Model? Definition, Uses, and Examples.* Investopedia. https://www.investopedia.com/terms/b/blackbox.asp

[8] Jones, D.R., Schonlau, M. & Welch, W.J. *Efficient Global Optimization of Expensive Black- Box Functions. Journal of Global Optimization 13, 455 – 492 (1998).* https://doi.org/10.1023/A:1008306431147

[9] Draper, N.R. (1992). *Introduction to Box and Wilson (1951) On the Experimental Attainment of Optimum Conditions. In: Kotz, S., Johnson, N.L. (eds) Breakthroughs in Statistics. Springer Series in Statistics.* Springer, New York, NY. https://doi.org/10.1007/978-1-4612-4380-9_22

[10] 3B1B. (2022, February 14). *Oh, Wait, Actually the Best Wordle Opener Is Not "Crane" ….* YouTube. https://www.youtube.com/watch?v=fRed0Xmc2Wg

[11] Benítez Gómez, Á., & Cavanillas Puga, E. (2022). *Wordle solving algorithms using Information Theory.*

[12] Crawford, T. (2023, July). *Using Information Entropy to 'Solve' Wordle.* TOM ROCKS MATHS. https://tomrocksmaths.com/wp-content/uploads/2023/07/using-information-entropy-to-e28098solve-wordle.pdf?utm_source=chatgpt.com

[13] Bischoff, M., Stix, G., & Yuhas, D. (2023, April 28). *Information Theory Finds the Best Wordle Starting Words.* Scientific American. https://www.scientificamerican.com/article/information-theory-finds-the-best-wordle-starting-words1/

[14] Unzueta, D. (2022, February 15). *Information Theory Applied to Wordle.* Medium. https://medium.com/data-science/information-theory-applied-to-wordle-b63b34a6538e

[15] Choy , S. (2023, December). *Demystifying Wordle: A Crash Course in Information Theory. Theory Hong Kong University of Science and Technology (HKUST).* https://sciencefocus.hkust.edu.hk/demystifying-wordle-a-crash-course-in-information-theory

[16] Liu, C. (2022, April 30*). Using Wordle for Learning to Design and Compare Strategies.* ArXiv. https://arxiv.org/abs/2205.11225

[17] Fodor, Z. (2011, October 24). *The Max-Value Entropy Search Acquisition Function*. Prezi. https://prezi.com/akbcahssqmwr/black-box-system/

【評語】052508

1. 本作品旨在探討如何將資訊熵（Information Entropy）應用於數學益智遊戲 Nerdle Maxi。研究聚焦於利用資訊熵衡量每次猜測所帶來的資訊量，以優化猜題策略並提升解題效率。作品嘗試使用不同 greedy 及 entropy 方式幫助降低所需猜測次數，

2. 本作品是資訊熵理論的實作延伸，展現運算效能管理、演算法比較設計與資料統計分析能力，體現學術嚴謹與創意思維並重。

作品海報

以資訊熵策略解決Nerdle Maxi 問題：尋找最少猜測次數的演算法

摘要

本研究試圖探索如何以最少猜測次數解決 Nerdle Maxi 問題。此問題涉及對資訊的處理，讓我們聯想到資訊理論中關於不確定性的研究。參考文獻後，我們發現有尚無人嘗試的作法，並實作、壓縮了猜測次數，保證我們能在六次內猜中。再將其與文獻中用在 Wordle 的做法融合後，壓縮平均猜測次數至 3.38 次。

壹、前言

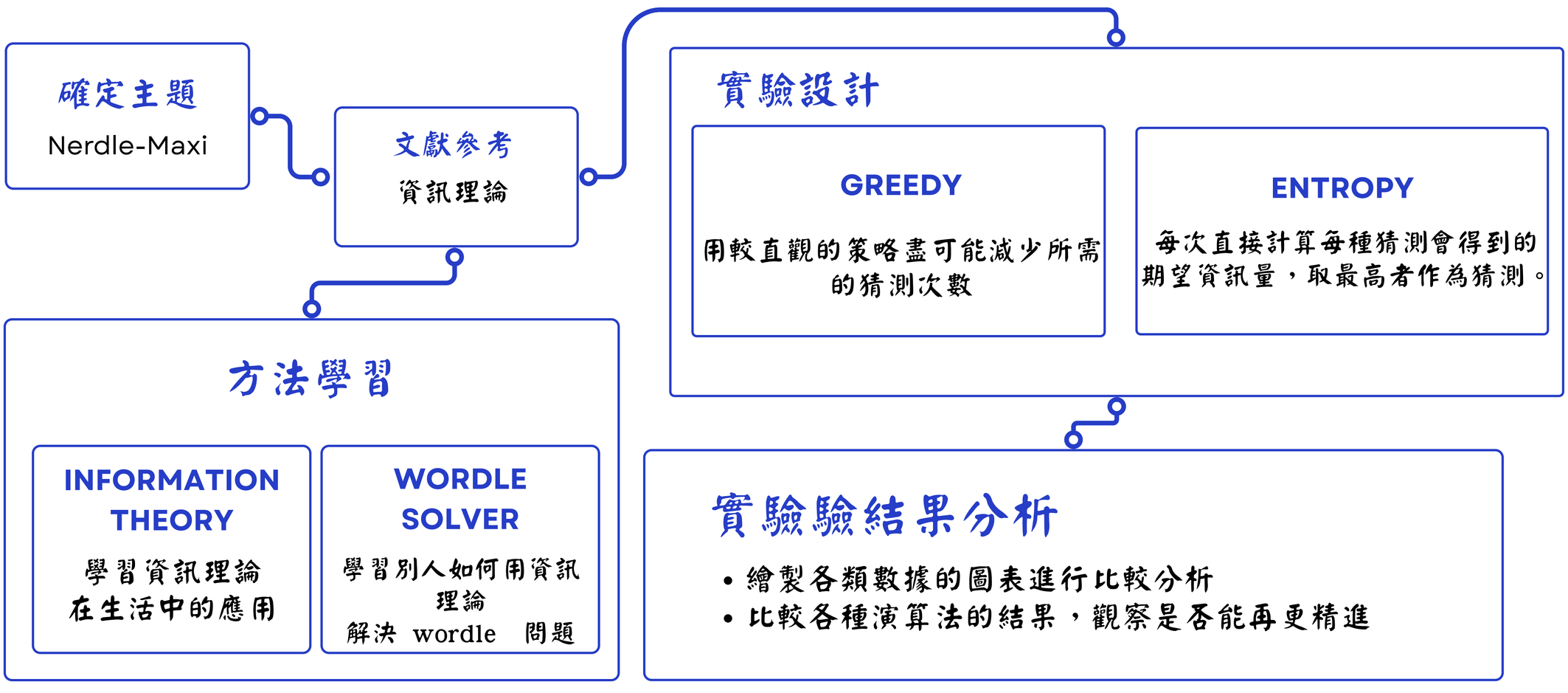
一、研究動機

對於 Nerdle Maxi，我們想說是否可以利用演算法縮減猜測次數? Nerdle Maxi 要求玩家通過不斷猜測，最終猜出正確的數學等式，為了探討如何有效減少猜測次數，我們開始研究各種方法，最終發現資訊理論中量化不確定性的工具，能幫助我們針對不同的猜測進行分析與評估，進一步探討如何應用資訊理論來設計演算法，希望能夠最小化 Nerdle Maxi 的花費猜測次數。

二、研究目的：

1. 使用多種不同的啟發式演算法，觀察其之間的花費猜測次數、資訊熵、計算時間
2. 比較不同的資訊熵方法，並比較其猜測次數、獲得的熵值
3. 藉由前面的實驗結果，找出能用最少猜測次數之演算法

貳、研究過程或方法



一、找出所有合法等式

需生成等式左側，生成後只需解析並計算其結果，檢查生成的等式是否符合 10 個字元的長度限制。左式還需小於或等於 8 個字元，因為至少需要 2 個字元用於等號和右式。左式是由左而右一格一格生成的，每添加一個字元時，下一個可添加的字元取決於上一個字元。

我們將符號分成以下幾種集合，並設定集合之間是否可以接續之關聯，即可確認下一個可添加的字元有哪些。

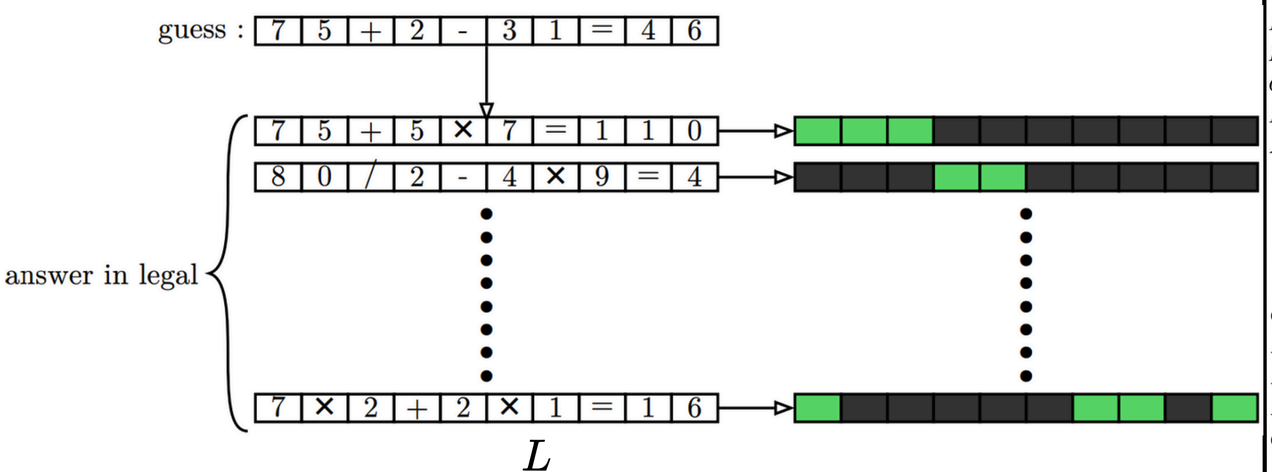
1. 數字集合 $N = \{0,1,2,3,4,5,6,7,8,9\}$
2. 運算符集合 $O = \{-,+,*,/\}$
3. 次方符號集合 $F = \{2,3\}$
4. 左括號集合 $U = \{\ (\ \}$
5. 右括號集合 $B = \{\ )\ \}$

二、Greedy1 演算法

定義 $\text{cal\_green}(w)$ 為，設等式 w 為猜測，枚舉所有可能的答案等式，並計算對應之回饋後，計算回饋中平均有多少格綠色（位置正確且符號或數字正確）。定義 g^* 為所有可能中 $\text{cal\_green}(w)$ 最大者。此演算法進行猜測時，先計算出所有可能答案等式的 $\text{cal\_green}(w)$，並使用 g^* 作為猜測。

此方法主要是要模擬人類遊玩的直覺策略，也就是在每次猜測時，盡可能讓更多格子是正確的，並與其他的方法比對。

下式 w_i、v_i 分別代表等式 w、v 的第 i 個字元，legal 代表目前可能為答案的等式的集合。



定義 $$G(w,v) = \sum_{i=1}^{L} \mathbf{1}\left[w_i = v_i\right]$$

式一 $$\text{cal\_green}(w) = \frac{1}{|\text{legal}|} \sum_{v \in \text{legal}} G(w,v)$$

式二 $$g^* = \underset{w \in \text{legal}}{\text{argmax}}\, \text{cal\_green}(w)$$

三、Greedy2 演算法

該演算法會在進行每次猜測時，目標是選擇可以平均讓答案可能數減少最多的等式，但由於使用平均可能會選中剩餘答案數標準差非常大之等式，因此最後決定選擇可以最小化剩餘可能答案的最大值（minimizing the maximum）之猜測，也就是下式中的 m^*。

首先，定義 w、v 為兩個任意符合遊戲規則的字串，r 為回饋字串（由綠、紅、黑三種顏色組成）集合 legal 為所有符合之前猜測的字串（可能為答案的字串）定義 $f(w,v)$ 為把 w 當作猜測，v 作為答案，的回饋字串。

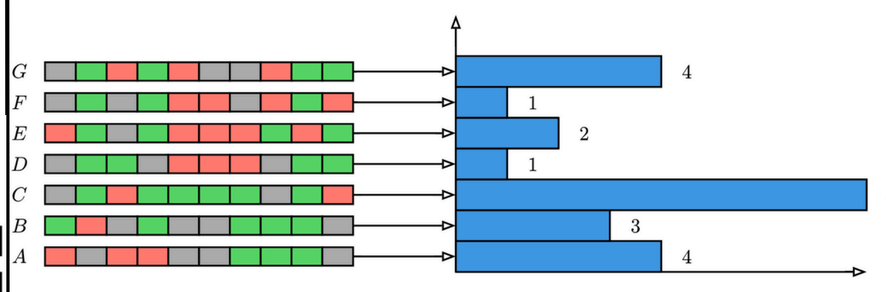
定義 $\text{count}(w,r)$ 為將 w 作為猜測，對於所有 v 屬於 legal，並且 $r = f(w,v)$，的數量。$\mathcal{F}$ 代表所有包含所有回饋的集合。

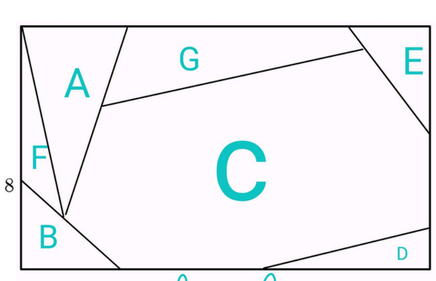
式三 $$\text{count}(w,r) = \left|\{v \in \text{legal} : f(w,v) = r\}\right|$$

定義 $$M(w) = \max_{r \in \mathcal{F}} \left|\text{count}(w,r)\right|$$

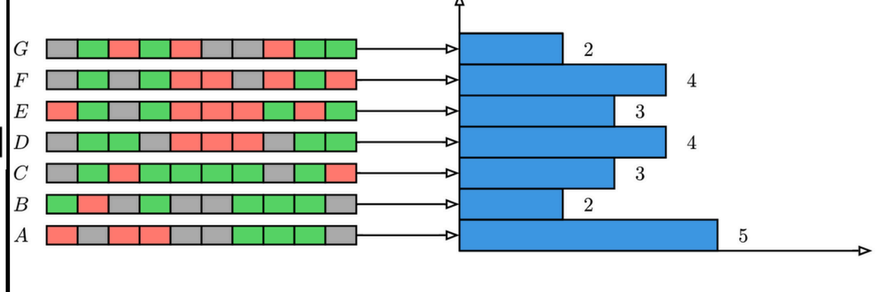
定義 $$m^* = \underset{w \in \text{legal}}{\text{argmin}}\, M(w)$$

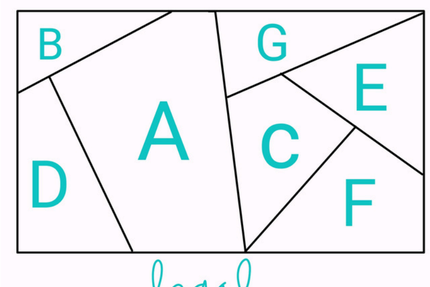
Guess：甲





Guess：乙





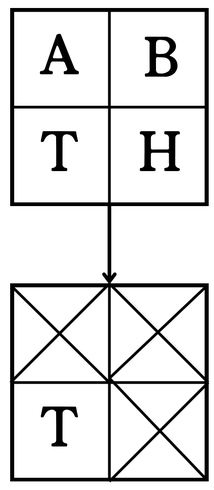
上面兩張圖分別代表某兩等式甲、乙作為猜測對應回饋與回饋出現次數示意圖，其中以甲等式做為猜測，獲得的回饋中，數量最多的種類 C 有 8 個，而以乙作為猜測，回饋中數量最多的種類 A 有 5 個，因此演算法會選擇乙等式作為猜測。

四、entropy1 演算法

(一)、將資訊量量化

資訊理論將資訊量量化，具體來說，得到 k bit 的資訊量，就是將可能結果縮小 $1/2^k$ 公式為 $log_2\left(\frac{1}{p(i)}\right)$

而資訊量的期望值就是資訊熵



獲得 2 bit 的資訊

(二)、將資訊理論應用於 Nerdle-Maxi

在每一輪的猜測時，會對於所有目前還可能是答案的等式（對於之前的猜測與回饋合法），計算每個等式的資訊熵。

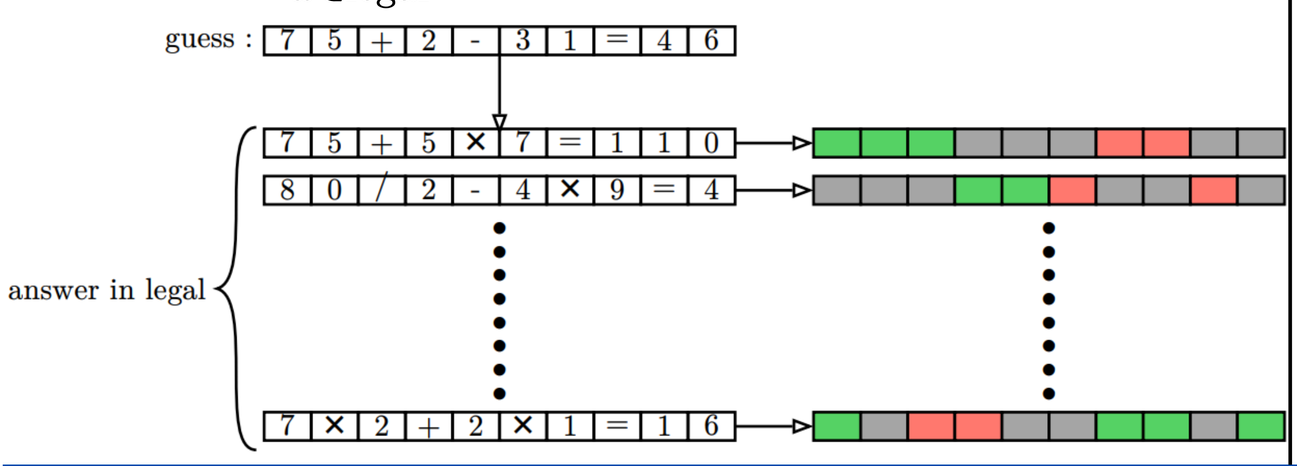
假設將 w 作為猜測問出，系統給的回饋字串為 r，可能答案的範圍，也就是 legal，就縮減成那些 $f(w,v)=r$ 的 v，可能答案種類數會從 $|\text{legal}|$ 變為 $\text{count}(w,r)$。

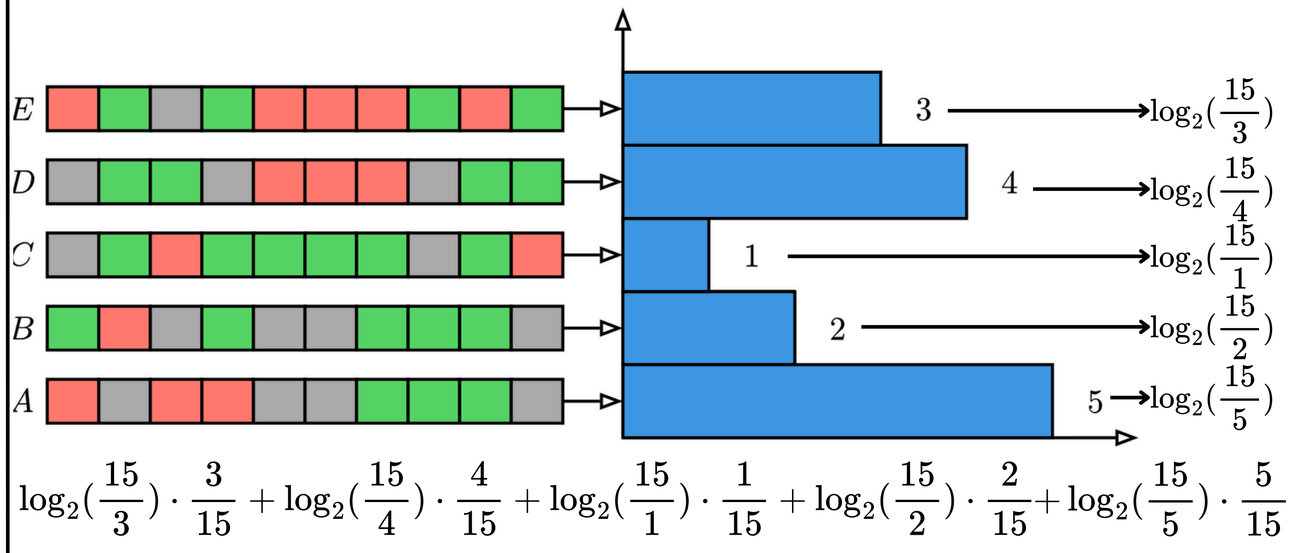
因此根據資訊量定義，獲得了式四的資訊量，又回饋字串為 r 的機率為 $\text{count}(w,r)/|\text{legal}|$，因此可用期望值公式算出式五，也就是以 w 作為猜測，加總所有情況獲得的資訊量乘以發生之機率，可以得到 w 的資訊熵（$P(w)$）。計算所有 legal 中等式的 $P(w)$ 後，取最大之等式（w^*）作為猜測。

式四 $$\log_2\left(\frac{|\text{legal}|}{\text{count}(w,r)}\right)$$

式五 $$P(w)=\sum_{r\in\mathcal{F}}\frac{\text{count}(w,r)}{|\text{legal}|}\cdot\log_2\left(\frac{|\text{legal}|}{\text{count}(w,r)}\right)$$

定義 $$w^*=\underset{w\in\text{legal}}{\text{argmax}}\,P(w)$$





將各種答案情況下獲得的資訊量乘以事件發生機率後加總得到此猜測的資訊熵為 2.15 bit。

五、entropy2 演算法

網路上大部分的研究都是以期望資訊量最大之猜測會出現在可能答案集合中為前提（或者加設差距非常小），並沒有實際將所有合法等式都列入猜測考慮，因此我們想要在實驗後了解此方法實際的表現會如何。

詳細算法與 entropy1 演算法差別在於 legal 集合換成以所有可能等式作計算，我們堆測這樣做可以在每次猜測時獲得更多資訊。因為原本如果有已經正確的格子（綠色），用 entropy1 時，下個猜測會繼承上面的綠色，無法獲得資訊。

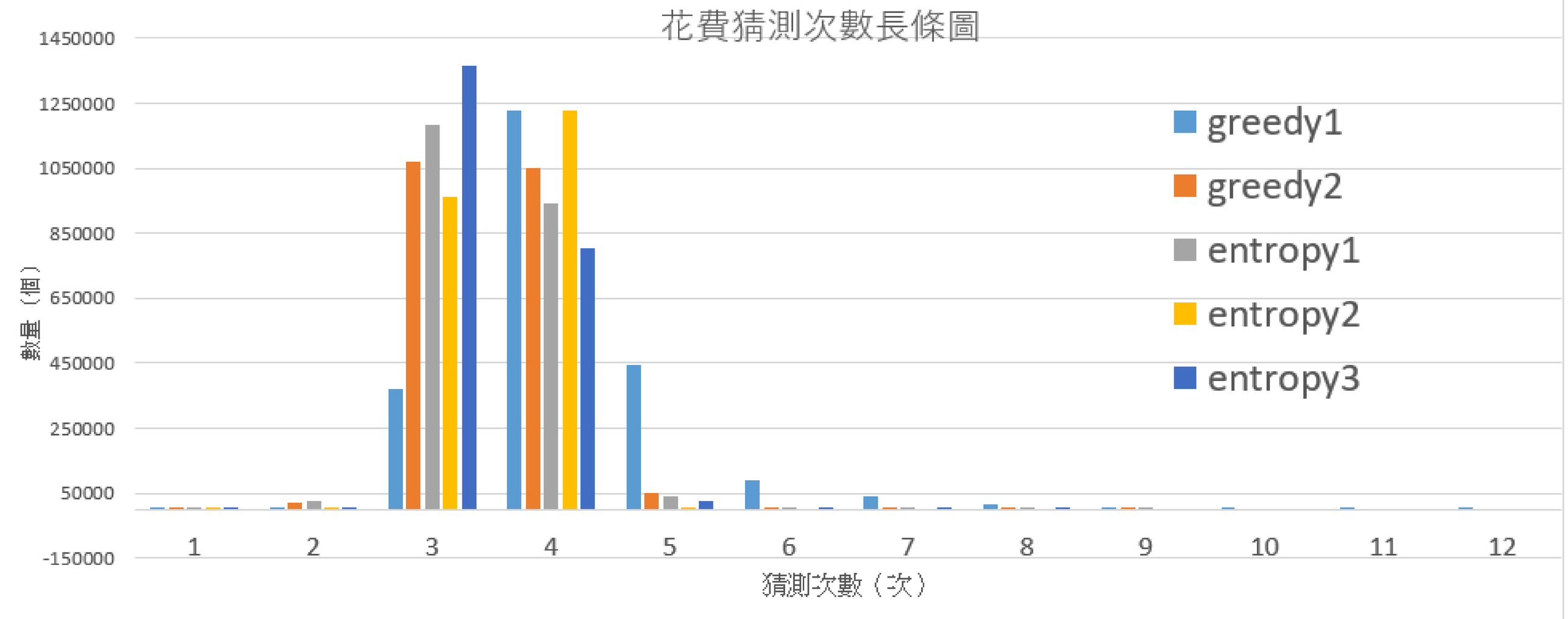
由於此方法所使用的計算量極大（是 entropy1 的千倍以上），我們將原本程式結合了 NVIDIA CUDA，藉由平行演算法與 GPU 的運算能力，將原本要跑一年以上的程式加速至1 ~ 2 日就可以算完。

六、entropy3 演算法

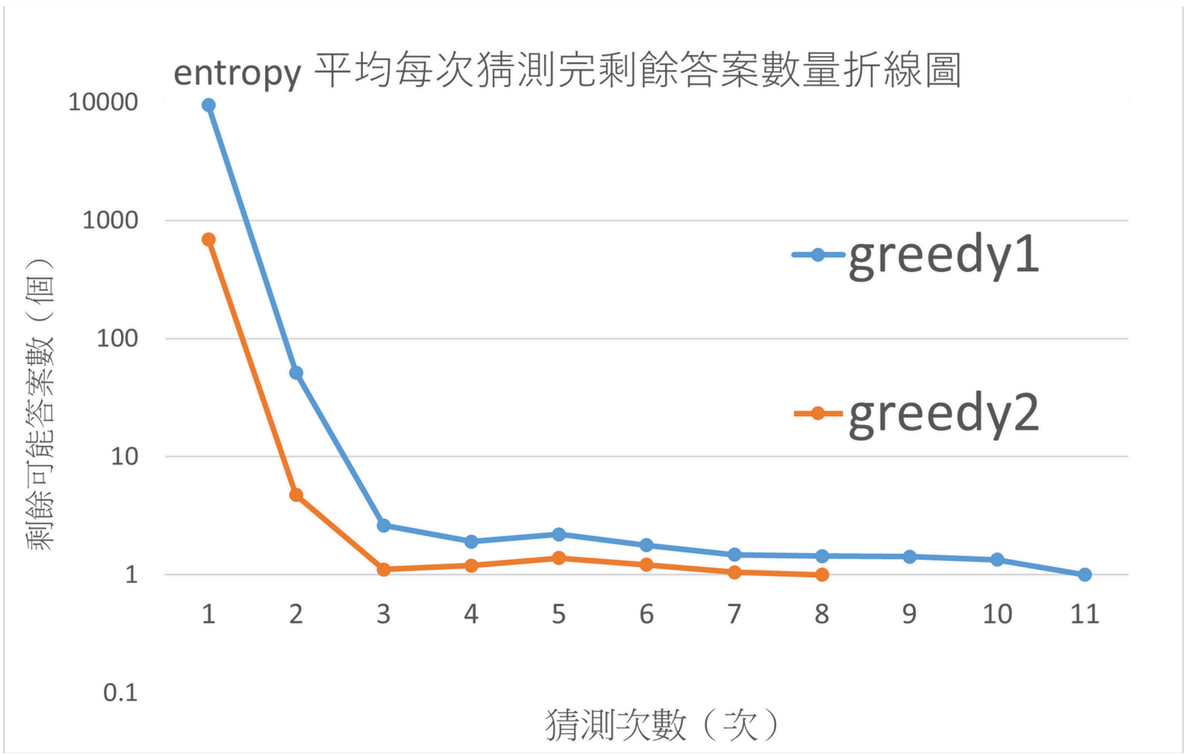
我們推測，entropy2 可以在 1、2 次猜測將可能答案範圍縮至更小，entropy1 則是在後續猜測上表現更好，因此我們想到融合 entropy1、entropy2 兩種方法，設計出了 entropy3，讓 entropy2 計算前兩次猜測，從第三次猜測開始，使用 entropy1。

參、研究結果

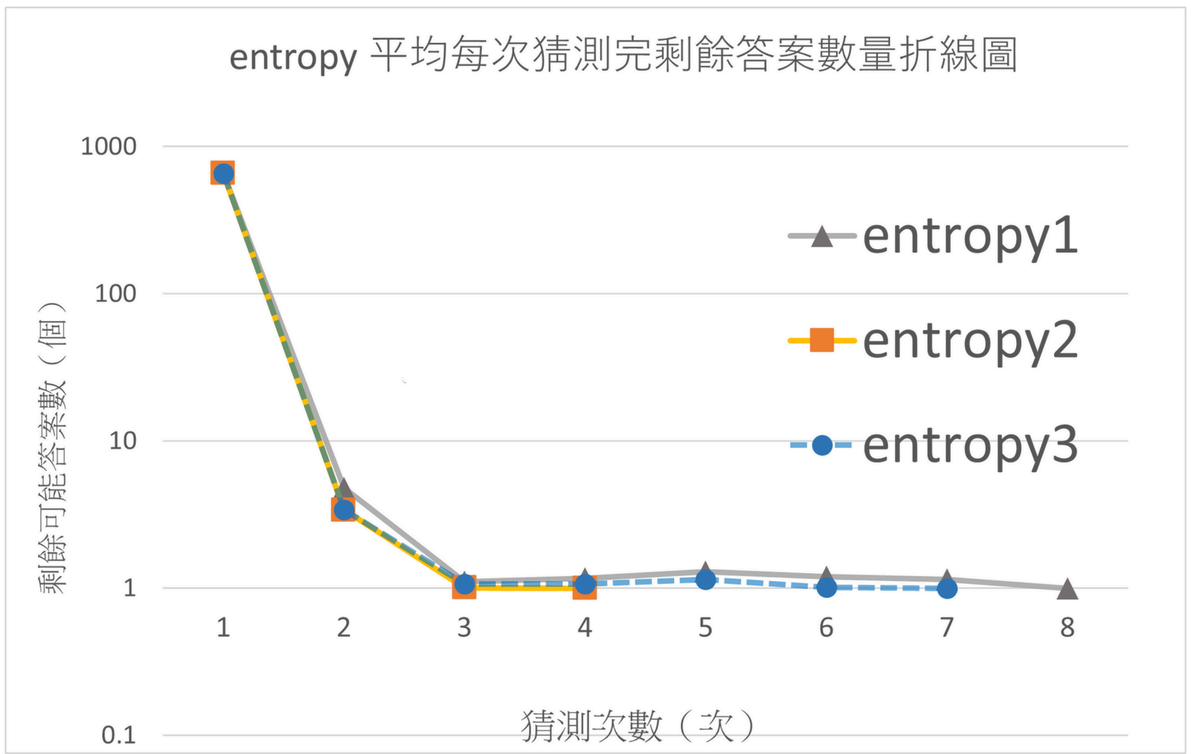
一、各演算法花費猜測次數長條圖



二、Greedy 系列剩餘答案數量折線圖



三、 entropy 系列剩餘答案數量折線圖



肆、討論

一、不同方法花費猜測次數之結果比較

	greedy1	greedy2	entropy1	entropy2	entropy3
算數平均 （花費猜測次數）	4.20	3.52	3.46	3.56	3.38
標準差	0.90	0.57	0.56	0.50	0.52
眾數（花費猜測次數）	4	3	3	4	3
最差情況（花費猜測次數）	12	9	9	5	8

二、entropy 方法之結果討論

從平均花費次數來看，entropy2 其實是三者中最差的，但從花費次數的分布可以看到 entropy2 最多只使用 5 次便猜中答案，而且 entropy2 縮減可能答案範圍的效果也較好，在第二次猜測完畢後，entropy1 剩餘答案可能數量平均為 4.78，而 entropy2 是3.40（entropy3 在第二次猜測的演算法與 entropy2 相同）。

綜上所述，如果目標是能保證解決 Nerdle Maxi 問題，我們建議使用 entropy2 演算法，而如果目標是能在平均上用最少次數猜中答案，我們建議使用 entropy3 演算法。

三、不同種類算法的時間複雜度分析與實際運行時間

	greedy1	greedy2	entropy1	entropy2	entropy3
時間複雜度	O(NGL)	O(NG(GL))	O(NG(GL))	O(NN(GL))	O(NN(GL))
運行硬體	30 CPU thread	30 CPU thread	30 CPU thread	RunPod GPU L40S*4	RunPod GPU L40S*4
實際運行時間 (小時)	1.06	3.58	10.83	約55.03	約28.96
平均單筆測資運行時間（秒）	0.0017	0.0058	0.0177	0.0901	0.0474

G 為平均第一次猜測的剩餘答案數量， N 為合法等式種類數， L 為等式長度

伍、結論

一、研究成果

本研究利用 CUDA，以顯卡加速，實作出了尚未有人嘗試的猜測方式（entropy2），保證玩家一定能在遊戲限制次數（六次）內猜出正確答案。

經過不同的演算法比較後，發現 entropy2 演算法的優點與缺點，並將新的方法融合原本的演算法（entropy1），最終模擬所有可能答案後，新演算法（entropy3）平均只需花費 3.38 次猜測即可猜出答案。

二、未來展望

(一) 答案出現機率不同的 Nerdle Maxi

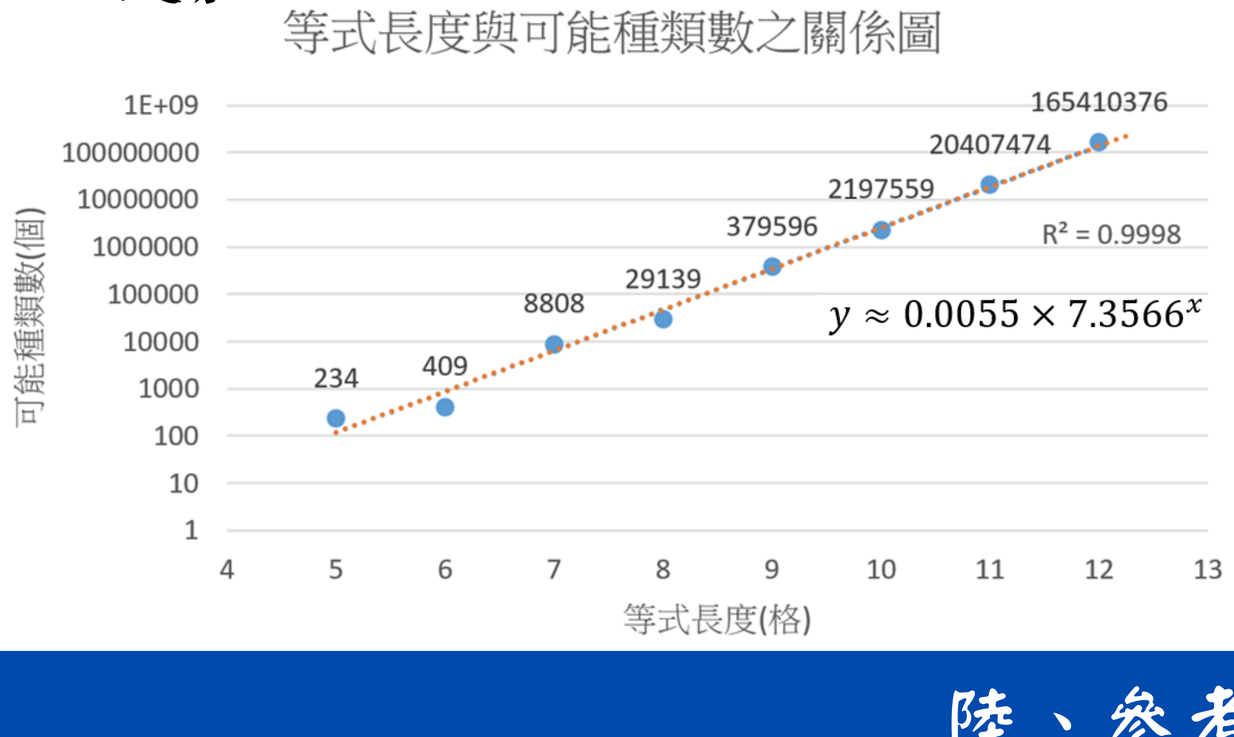
當等式的出現機率不相等時，總熵無法用我們原本的運算方法。這種情況下，需考慮不等機率下的熵分解公式。

1. 假設整體等式集合 W 每個猜測 A 為答案的機率為 P (A)
2. 可能為答案的集合為 W+ 且其出現的總率為 $w^+ = \sum p(W+)$
3. 不可能為答案的集合為 W- 且其出現的總率為 $w^- = \sum p(W-)$
4. 可能為答案的集合的熵 = $H^+ = \sum \left(\frac{p(w)}{w^+} \log_2 \left(\frac{w^+}{p(w)}\right)\right)$
5. 不可能為答案的集合的熵 = $H^- = \sum \left(\frac{p(w)}{w^-} \log_2 \left(\frac{w^-}{p(w)}\right)\right)$
6. 總熵 H (W) 可表示為：

$$H(W) = w^+H^+ + w^-H^- + w^+ \log_2(\frac{1}{w^+}) + w^- \log_2(\frac{1}{w^-})$$

（二）事件數量大小過大問題

討論事件數量時，我們發現當 Nerdle Maxi 格子總數過大時，可能等式的數量會上升到難以在有限時間內計算完畢，導致我們的演算法無法正常運行。下圖是以 5~12 作為可用的字元數生成等式長度來估算可能等式數量成長的趨勢



我們查到在這種情況下，可使用另一種運用資訊熵的方法： Bayesian Optimization（ Donald R. Jones Matthias, Schonlau, & William J. Welch 1998 ）

此方法會利用到以下三種概念：

- Black-Box Function(Franz Breisig, 1921)：指的是一種你只能以「輸入→Black-Box Function→輸出」的方式來使用、但無法直接取得其內部結構或解析式的函數。

- Surrogate Model ：進行反算以降低運算時間。Surrogate Model 建立使用模式的方法為先訓練、再驗證、最後代理 Black-Box Function。大部份使用 Gaussian Process, GP 作為 Surrogate Model。

- MES Acquisition Function：Acquisition Function 根據目前 Surrogate Model 對不同 的預測均值及不確定度來評分每一個候選點，決定下一輪最值得的輸入。才能用最少的試算次數獲得最多資訊或最快提升目標值。其中，The max value entropy search (MES)是一種基於資訊理論的選擇策略，其核心目的是去選擇那些目前對結果影響最大、但還不確定的決策，而最小化我們對最優結果的選擇。也就是說，MES Acquisition Function 選擇能最大化「資訊熵」的猜測。

- Bayesian Optimization 能在有限次的函數評估下（可能價格昂貴或實驗時間需要極長或儘可能少的查詢次數內找到最大值），藉由 Surrogate Model + Acquisition Function 重覆計算與更新，找到 Black-Box Function中最能影響結果的變因：

「初始隨機探索 → Surrogate Model & MES Acquisition Function → Black-Box Function（實際實驗）→ 更新 Surrogate Model 」。

綜上所述，即便是在 Nerdle Maxi 的格子數量過大的情況下，透過 Bayesian Optimization 與 MES Acquisition Function，依然能夠透過有限次猜測，每次選擇資訊熵最大的猜測，藉由回饋調整 Surrogate Model，最終猜到正確答案。

陸、參考文獻資料

[1] 3B1B. (2022, February 6). Solving Wordle Using Information Theory. YouTube. https://www.youtube.com/watch?v=v68zYyaEmEA
[2] Markowsky, G. (2024, April 12). Entropy in Information Theory in Classical Information Theory. Britannica. https://www.britannica.com/science/information-theory/Entropy
[3] Marcon, E. (2022, December 3). Partitioning Information in 3B1B Wordle Video. Github. https://ericmarcon.github.io/3B1B-Wordle/3B1B-Wordle.pdf
[4] Lokshtanov, D., & Subercaseaux, B. (2021, May 14). Wordle Is NP-Hard. ArXiv. https://arxiv.org/abs/2203.16713
[5] Rosenbaum, W. (2021, May 8). Finding a Winning Strategy for Wordle Is NP-Complete. ArXiv. https://arxiv.org/abs/2204.04104
[6] Botorch. (2019, November 6). The Max-Value Entropy Search Acquisition Function. BoTorch. https://botorch.org/docs/tutorials/max_value_entropy/
[7] Kenton, W. (2024, February 2). What Is a Black Box Model? Definition, Uses, and Examples. Investopedia. https://www.investopedia.com/terms/b/blackbox.asp
[8] Jones, D.R., Schonlau, M. & Welch, W.J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13, 455 - 492. https://doi.org/10.1023/A:1008306431147
[9] Draper, N.R. (1992). Introduction to Box and Wilson (1951) On the Experimental Attainment of Optimum Conditions. In: Kotz, S., Johnson, N.L. (eds) Breakthroughs in Statistics. Springer Series in Statistics. Springer, New York, NY. https://doi.org/10.1007/978-1-4612-4380-9_22
[10] 3B1B. (2022, February 14). Oh, Wait, Actually the Best Wordle Opener Is Not “Crane”…. YouTube. https://www.youtube.com/watch?v=fRed0Xmc2Wg
[11] Benítez Gómez, Á., & Cavanillas Puga, E. (2022). Wordle solving algorithms using Information Theory.
[12] Crawford, T. (2023, July). Using Information Entropy to ‘Solve’ Wordle. TOM ROCKS MATHS. https://tomrocksmaths.com/wp-content/uploads/2023/07/using-information-entropy-to-e28098solve-wordle.pdf?utm_source=chatgpt.com
[13] Bischoff, M., Stix, G., & Yuhas, D. (2023, April 28). Information Theory Finds the Best Wordle Starting Words. Scientific American. https://www.scientificamerican.com/article/information-theory-finds-the-best-wordle-starting-words1/
[14] Unzueta, D. (2022, February 15). Information Theory Applied to Wordle. Medium. https://medium.com/data-science/information-theory-applied-to-wordle-b63b34a6538e
[15] Choy, S. (2023, December). Demystifying Wordle: A Crash Course in Information Theory. Theory Hong Kong University of Science and Technology (HKUST). https://sciencefocus.hkust.edu.hk/demystifying-wordle-a-crash-course-in-information-theory
[16] Liu, C. (2022, April 30). Using Wordle for Learning to Design and Compare Strategies. ArXiv. https://arxiv.org/abs/2205.11225
[17] Breisig, F. (1921). Über das Nebensprechen in Fernsprechkreisen. Elektrotechnische Zeitschrift, 42(34), 933 - 939.