# **郵票問題**

壹、研究動機

雖然郵票面值分配的問題不是什麼了不起的學問，卻是一種可以在生活中幫助人們的數學。(未完待續)

貳、研究目的

給定一頁m\*n的長方形郵票求可連續私下的面值總額達到的連續最大值及其面值分配。

參、研究設備及器材

* A4筆記本一個
* A4內頁一疊
* 筆，文具
* 可計算之電腦

肆、研究過程或方法

一、 可能可行的解法

\* 由最大值往下尋找可能的分配

\* 由一個區域最佳解逐步調整各面值往上逼近可達到的最大的連續最大值

二、 已知資訊

總數為M\*N的郵票可以達到的連續最大值為其連續子集合個數。公式正在導出。

三、 解法

由一個區域最佳解逐步調整各面值往上逼近可達到的最大的連續最大值

四、辭典

用途：整理各種名詞及縮寫

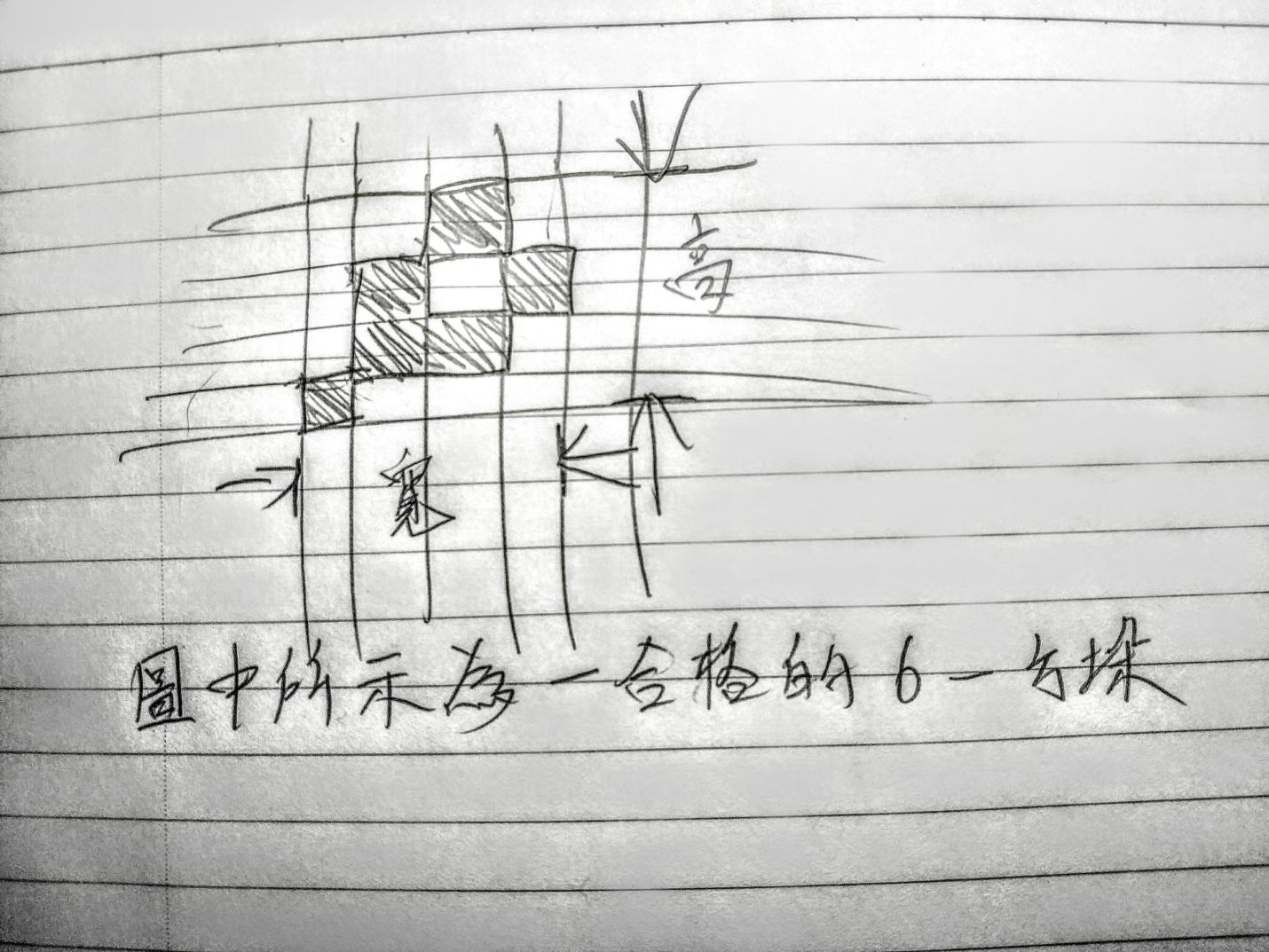
b、Block、方塊，指的是在一頁郵票可以撕下的郵票

一頁郵票，指的是一頁長m張郵票寬m張郵票的長方形郵票集合，而且每一個郵票至少有兩個共用邊

郵票，指的是正方形的紙，具有正整數面額

寬和高，如下圖所示。

注意。就算方塊中間有空洞，依然是合格的郵票組合。



(本圖經過適當修圖使得細節更加清楚)

五、 日誌(沒有整理)

2020/1/4 17:26

也許可以在給MxN的方格後，再慢慢調整其中的數值讓他可以達到的連續最大值慢慢變大…

這麼的話，初始的排法就很重要了。

11111 12121

11111 or 21212

11111 12121

暫先不考慮郵票上最多可貼幾張。

'''

總數為M\*N的郵票可以達到的連續最大值為其連續子集合個數。

'''

#以下為導出公式的嘗試

M\*N +

（1\*1）

同一形狀可擺的位置數量取決於可容納他的最小格子點長方形長寬

1 m\*n

2 (3\*3) 6+6+8

sigma g1 = 1 ~ MN

G2 1 ~ G2max

G2()

1 2 3 4 5 6 7 8 9 10

1 3 2+4+4+2+2+4

1/26

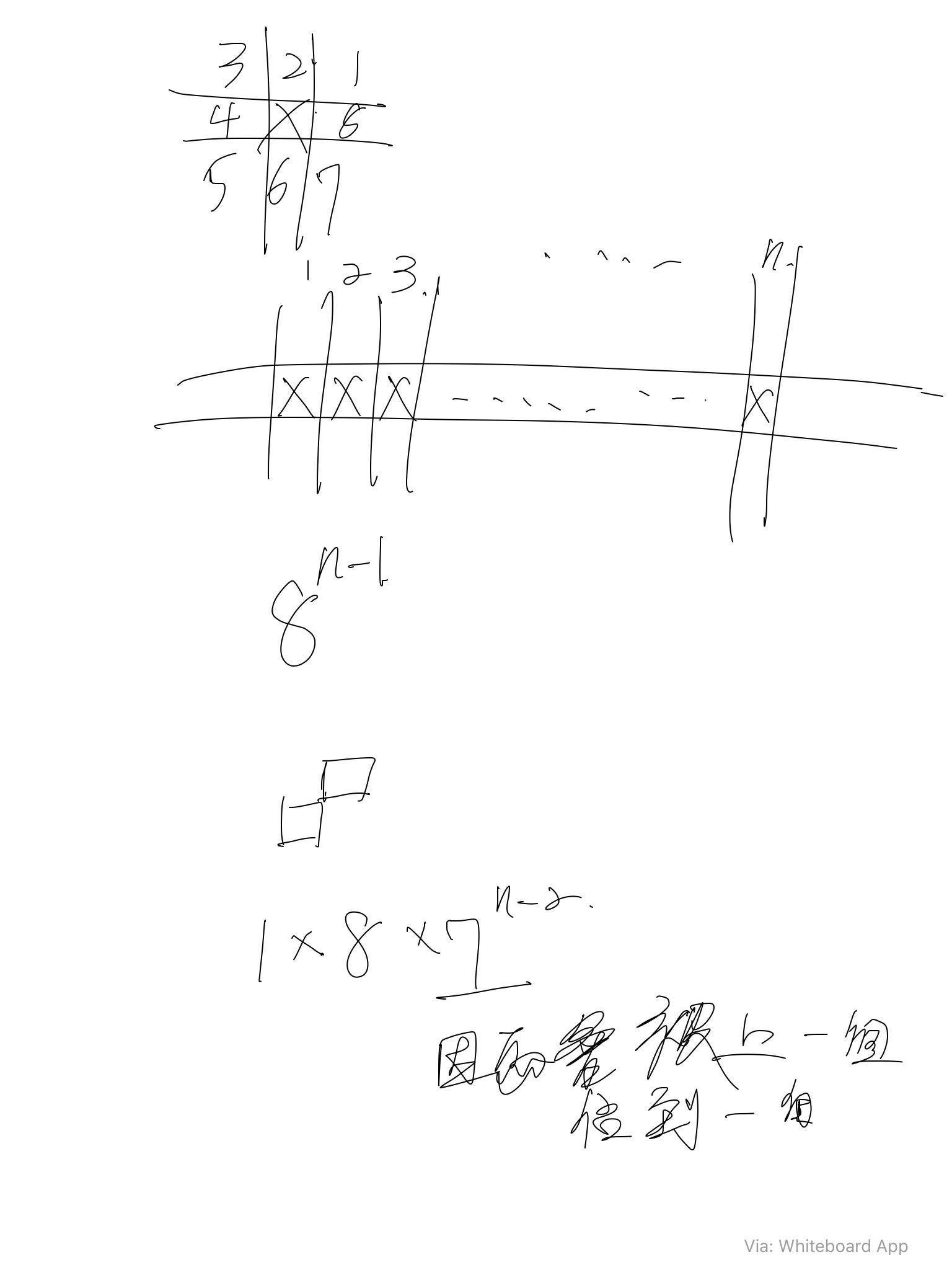
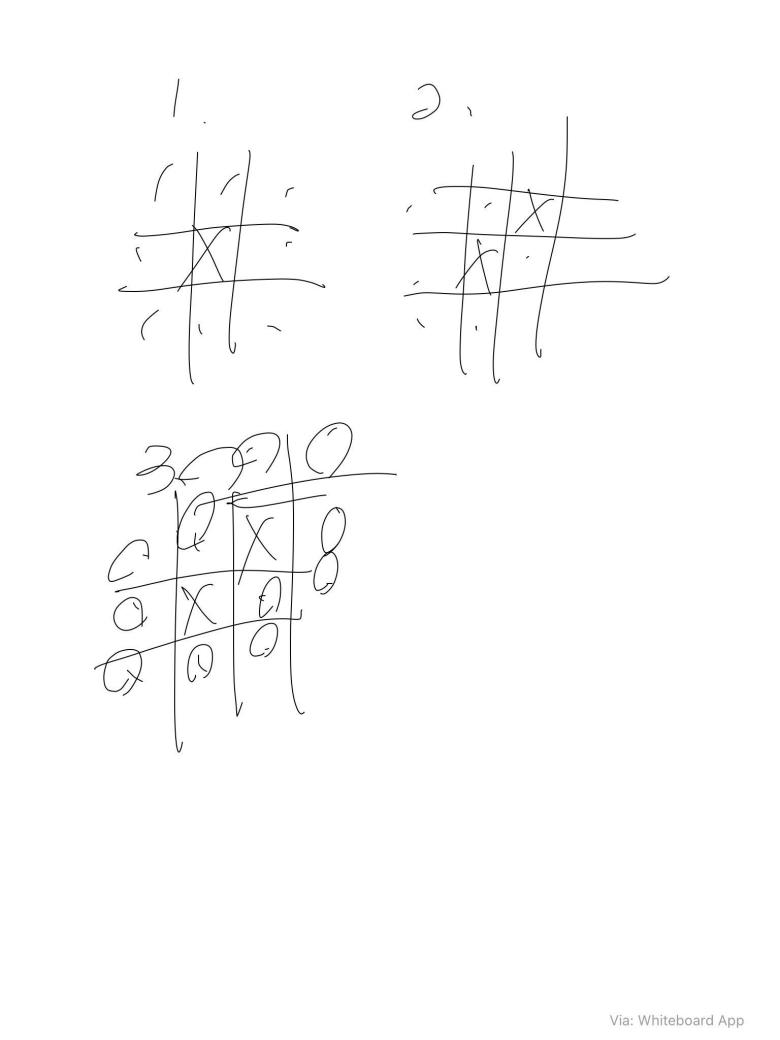
「數學上把n個方塊構成的類似俄羅斯方塊的圖形，就稱為是一個「n –方垛」或「n –方塊」（n-polyomino）。如果翻面或旋轉都視為相同一種，則4 –方塊只有5 片， 5 –方塊有12 片， 6 –方塊有35 片， 7 –方塊有108片， 8 –方塊有369 片。圖二就是6 –方塊的35 種圖形組合。」

這是我在維基百科上的Polyomino頁面上看到的。

然而我們討論的「總數為M\*N的郵票的連續子集合個數」是指sigma a=1~m\*n a-方垛的可能數

而且就算旋轉後一樣，也會視為相異的形狀。

所以我目前的目標會放在以n表示我們口中的n-方垛的可能數



根據仔細的觀察後發現n+1-方垛的可能數就是所有可能的n-方垛的外圍在接上一個方塊的方法數剪掉其中重複的圖案再減掉一樣的。

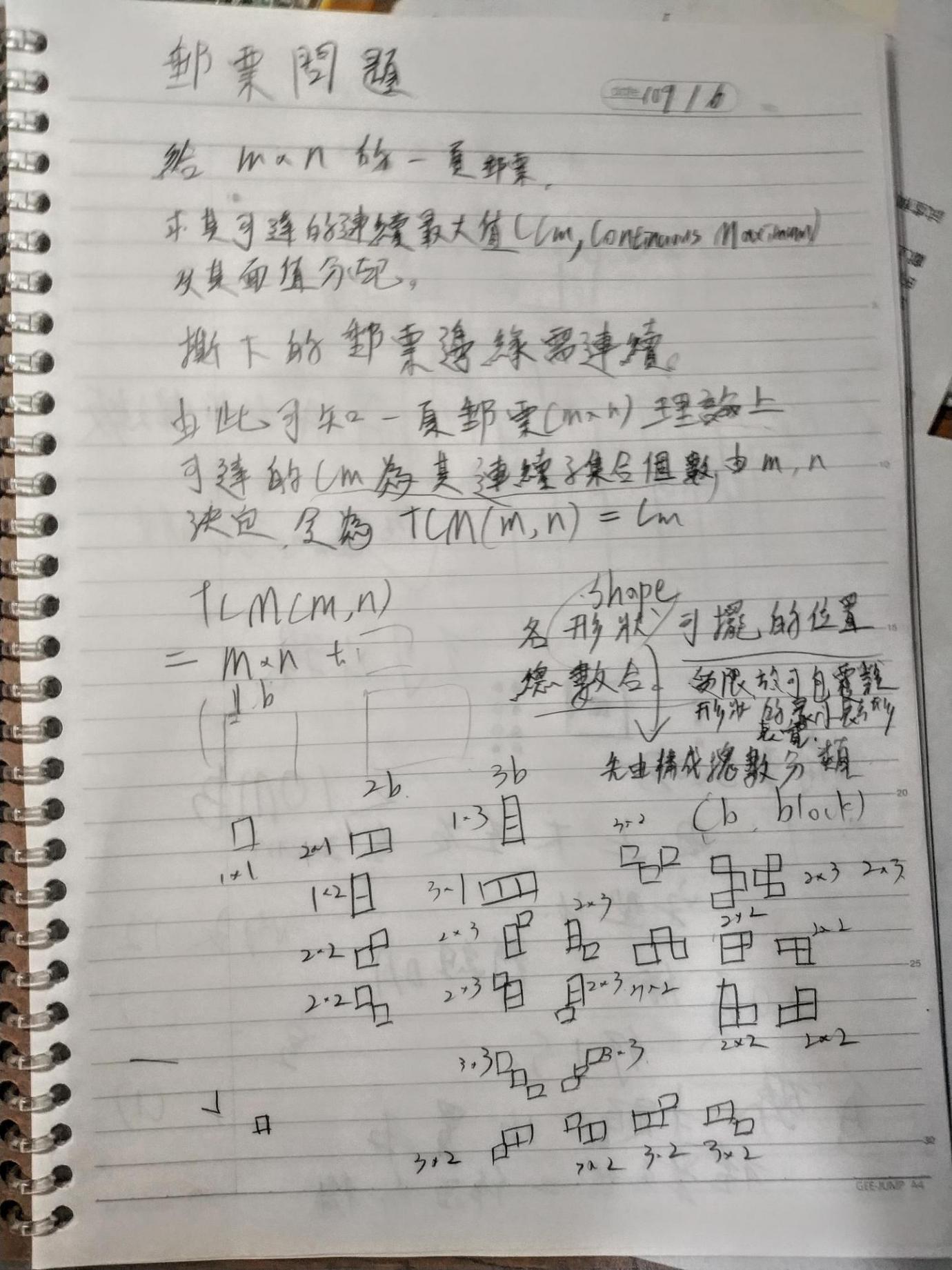
這句話經過斷句後，大致上呈現了上圖中右邊那張的意思

不用減掉旋轉後一樣的，因為在我們的這個狀況裡，不同的方向的同一圖案會有可能會有

不同的寬和高。

所以，目前先寫出可以算出n-方垛有幾種可能的程式吧。

由n=1時，只有一種，而n+1-方垛的可能數就是所有可能的n-方垛的外圍在接上一個方塊的方法數剪掉其中重複的圖案再減掉一樣的。所以可以用遞迴的方式來寫，為了減低重複讀取相同結果會造成資源浪費，將會使用動態規劃的技巧。

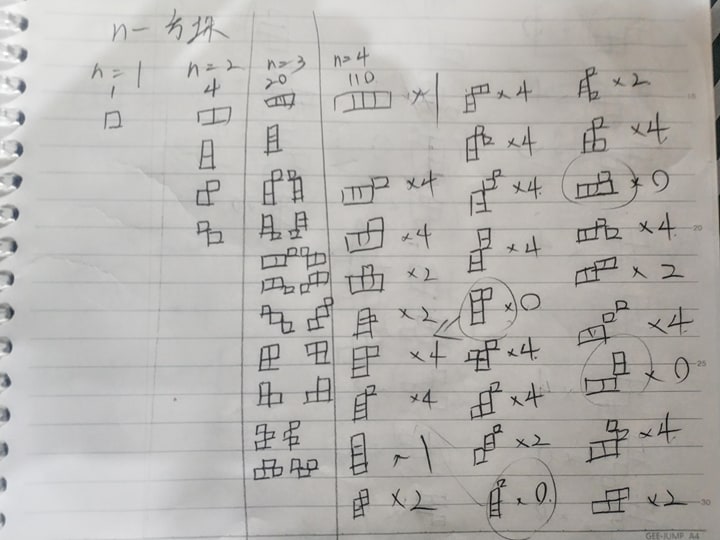


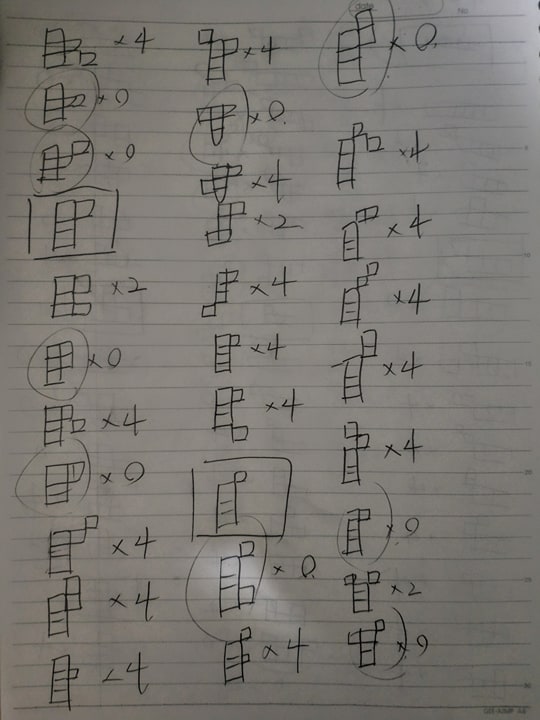
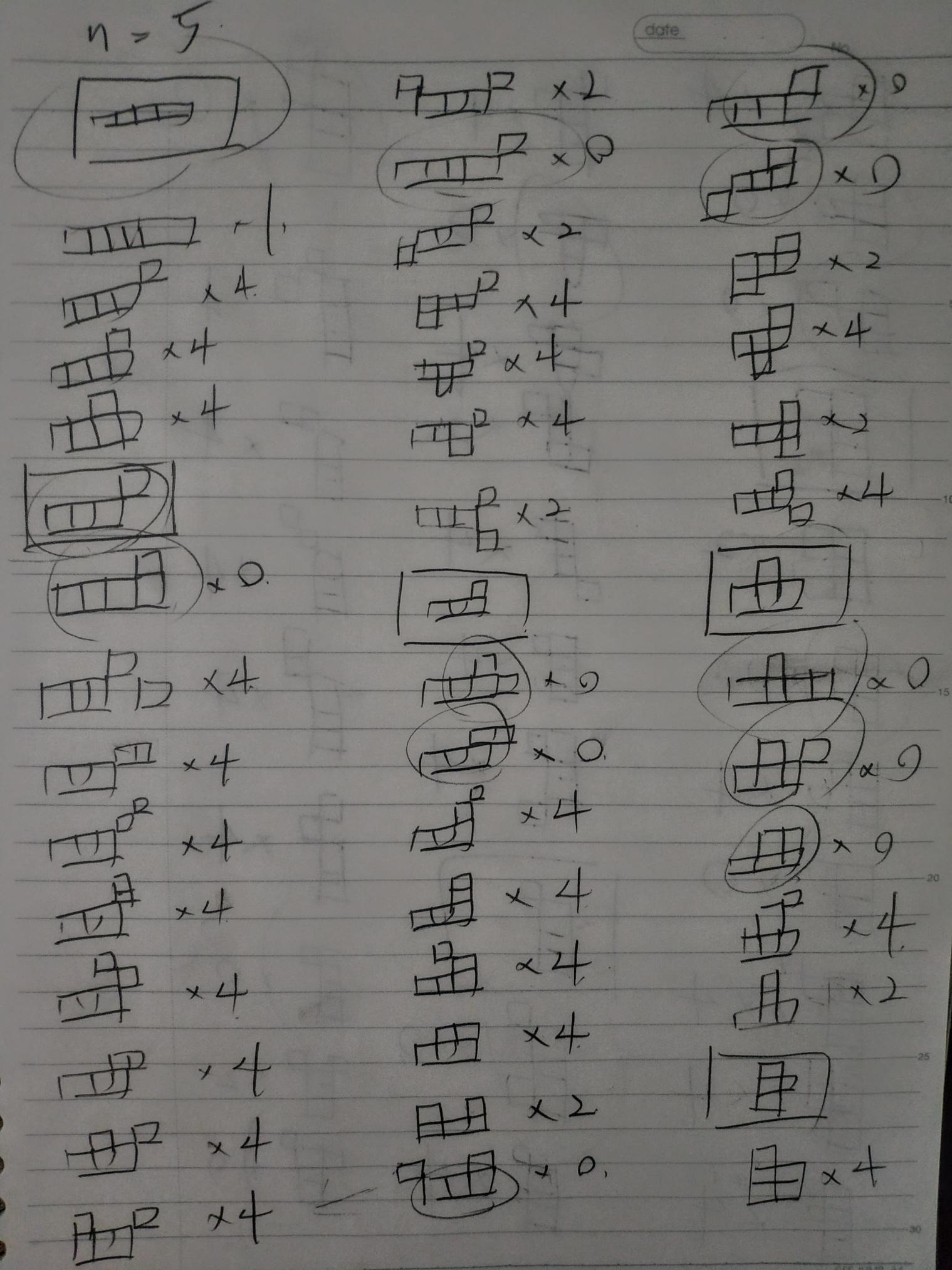
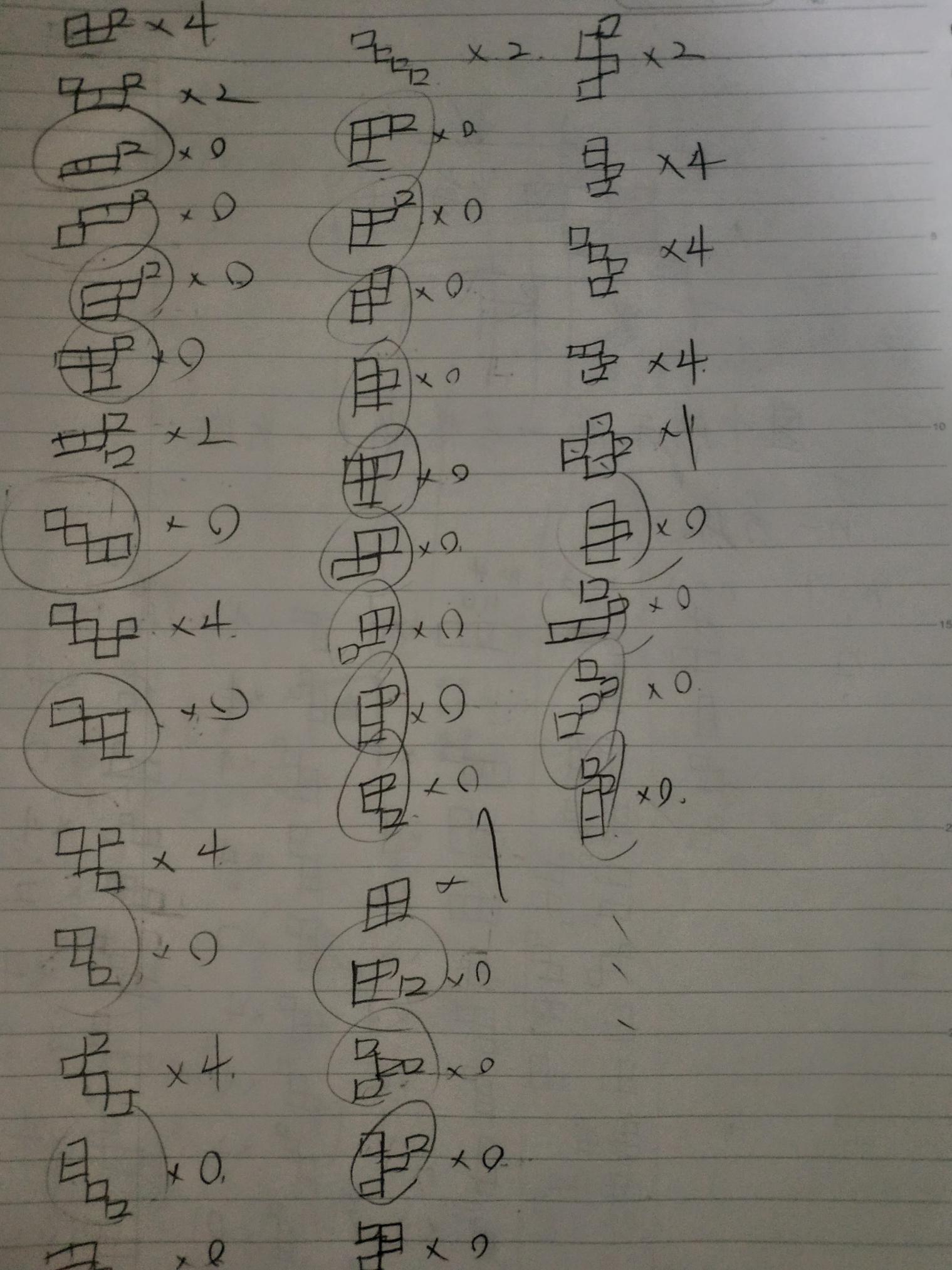
(此圖也經過適當修圖使得細節清楚)

這是筆記內頁，記載了一些想法

import numpy as np  
  
shape\_num = [1]  
shape = np.zeros((3, 5, 6, 7), dtype=np.int)  
shape[1, 2, 2, 0] = 1  
  
print(shape)  
  
  
def calculate\_shape\_num(n):  
 if len(shape\_num) > n:  
 return shape\_num[n-1]  
 else:  
 pass  
  
  
while True:  
 try:  
 n = int(input('n?'))  
 except:  
 print('err, pls input positive int')  
 continue  
 if n == 1:  
 print(1)  
 continue  
 elif n < 1:  
 print('err, negative or zero get')  
 continue  
 else:  
 x = 0  
 while x < n:  
 x += 1  
 print(x, end=' \r')  
  
  
print('Program finish')

目前的程式碼，還沒有完全完成，目前可以執行但無實際功能，其中因為Python的基礎函式庫(library)好像不支援多維陣列，所以我使用numpy套件來實現多維陣列，目的是為了要儲存過程中計算出的每一個n-方垛的所有形狀，所以有四維[n, n-方垛的數量, 圖形的x, 圖形的y]

另一方面，我也試著從數字中找出規律。並試著把n=1~5 方垛列出來，結果5還沒列完，我就有了想法。



在經過一系列的機械性動作後，我發現我重複著以下動作。

1. 在n-1方垛中找還沒加工過的形狀
2. 從最右下角開始加上新的b，然後依逆時針方向將n-1方垛的原圖都各加一個b
3. 遇到翻轉過後一樣的形狀就圈起來，標記\*0
4. 如果沒有畫過就看上下左右翻轉，不是旋轉，共有幾種不同的形狀。然後標記
5. 最後再把所有數字加起來標記在n的旁邊

所以我目前會想辦法找出一個方法讓電腦比較好判斷哪些形狀有畫過了，如果做到了，那麼我們就完成了一個小目標。

伍、參考資料

Python中三維陣列位置詳解：<https://www.itread01.com/content/1547394667.html>

多方塊的謎題：[http://scimonth.blogspot.com/2010/12/blog-post\_1360.html](http://scimonth.blogspot.com/2010/12/blog-post_1360.html" \t "_blank)

Polyomino：[https://en.wikipedia.org/wiki/Polyomino](https://en.wikipedia.org/wiki/Polyomino" \t "_blank)