鄭然而生

高中職組數學科三等獎

國立臺灣師範大學附屬高級中學
學生姓名：伍志忠　徐滀玢　郭長霖
指導老師：洪允東　殷灝

計畫名稱:郵然而生

摘要

由 6 張郵票排成 2×3 的矩形，填上了 6 種不同的面額後可以從 1 開始直到 36 的正整數都撕得出來。此為數學界中流傳已久的郵票問題。不同的填法將導致不同的結果。在此研究中，我們將各郵票的以點來表示面額的填入位置，以連接線來表示郵票的連通，這種圖被稱作 *IC* 圖，填入面額即 *IC* 著色。本次研究著重於矩形圖。

壹、 研究動機

數學課上老師曾經提到的一個古老的問題:「現有 2×3 矩形郵票，如何設計這 6 個面額使得從 1~36 元都可以撕出。撕的軌跡必須連續不可間斷。」起初我們僅是想要將這題的答案解出，但沒想到這並非一個簡單的工作。於是，我們決定從較簡單的題目開始進行研究，歸納出相關的解。

貳、 研究目的及研究問題

一、 原問題概述

2×3 矩形排列的正方形郵票，將面額填入後，以連續不間斷的方式撕郵票，使得從 1 開始，連續 k 個正整數都能撕出。欲找出 k 的最大值，及此時的面額配置。

二、 研究目的

我們想找出的是關於 $1\times n$ 與 $2\times n$ 的矩形郵票之相關解，並且推廣至其他情況。

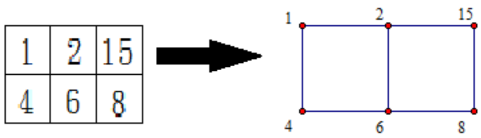
參、 研究設備及器材

紙、筆、電腦、Microsoft Office Excel、Dev-C++

肆、 研究過程與方法

一、先將郵票轉化為由點與圖構成的 *IC* 圖，則問題轉為將每個點配置數值，找出最大的 k 值，即 *IC* 著色數。定義相關名詞如下:

1. 點:可以填入面額的地方。
2. 圖:由點與連接線所構成的圖。
3. 連通圖:任二點均有連接的圖。



4. 子圖:一個圖的一部份稱為該圖的子圖。子圖依其是否為連通圖分為連通子圖與不連通子圖。

5. 連通子圖數:一個圖其所有連通子圖的數量，即原始問題中，郵票的撕法數。

6. *IC* 著色(*IC*-Coloring):對於一個給定的圖 G 及 G 的頂點之標號方式 f，所有連通子圖的頂點標號數字的和為從 1 開始，連續的 k 個正整數，(可重複)，則稱 f 為 G 的一個 *IC* 著色。

7. *IC* 著色數:所有的 *IC* 著色中，k 的最大值。通常以 $M(G)$表示圖 G 的 *IC* 著色數。

8. $1\times n$ 的矩形期圖形為一鏈狀(Path)，以 P_n 表示有 n 個點的鏈狀圖。

9. $m\times n$ 的矩形以 $P_m\times P_n$ 表示之。

二、基本規則

1. 每張郵票(即每個頂點)均只能填入一正整數的面額。
2. 撕郵票路徑必須連續，故子圖只取連通子圖。
3. *IC* 著色數不大於頂點數值的總和。

三、 $1\times n$ 矩形(P_n)的研究

(一) 、連通子圖數

P_n 中，含有 1 個點的連通子圖共有 n 個，含有 2 個點的有$(n$-1)個，依此類推，含有 k 個點的有$(n$-$k)$個。因此，P_n 的連通子圖數有 $\sum_{k=1}^{n} k=\frac{n(n-1)}{2}$ 個。

(二) *IC* 著色數

在討論 *IC* 著色數時，容易知道其 *IC* 著色數不可能超過連通子圖數，因為當 *IC* 著色數等於連通子圖數時，每個連通子圖恰對應 1 個頂點數值和。

在文章”IC-Colorings and IC-Indices of graphs”(參考資料三)中有提及關於鏈狀圖(Path)的 IC 著色數。該篇文章中提出了一個 *IC* 著色數的下界:

$\left(2+\left\lfloor\frac{n}{2}\right\rfloor\right)\left(n-\left\lfloor\frac{n}{2}\right\rfloor\right)+\left\lfloor\frac{n}{2}\right\rfloor-1$。不過此問題目前仍未被解決。我們利用了電腦程式協助找出幾組 *IC* 著色數，雖然後面有幾組比上述文章中所提及的配置方法所得的數值大，但目前還未找出這些數值與 n 的確切關係。

而目前我們找到的 P_n 的 *IC* 著色數與數值的配置列表於表 1。

n 值	連通子圖數	*IC* 著色數	滿足此 *IC* 著色數的數值配置方法數	滿足此 *IC* 著色數的數值配置列表
1	1	1	1	1. (1)
2	3	3	1	1. (1,2)
3	6	6	1	1. (1,3,2)
4	10	9	2	1. (1,1,4,3) 2. (1,3,3,2)
5	15	13	3	1. (1,1,4,4,3) 2. (1,3,1,6,2) 3. (1,5,3,2,2)
6	21	17	6	1. (1,1,1,5,5,4) 2. (1,1,4,4,4,3) 3. (1,1,6,4,2,3) 4. (1,1,6,4,3,2) 5. (1,3,6,2,3,2) 6. (1,7,3,2,2,2)
7	28	23	2	1. (1,1,9,4,3,3,2) 2. (1,3,6,6,2,3,2)
8	36	29	3	1. (1,1,12,4,3,3,3,2) 2. (1,2,3,7,7,4,4,1) 3. (1,3,6,6,6,2,3,2)
9	45	36	1	1. (1,2,3,7,7,7,4,4,1)
10	55	43	1	1. (1,2,3,7,7,7,7,4,4,1)
11	66	50	2	1. (1,2,3,7,7,7,7,7,4,4,1) 2. (1,1,1,20,5,4,4,4,4,3,3)

▲表 1:P_n 的相關數值

(三) 已知的 *IC* 著色數間的關係

1. 上表中的數據是已經經由電腦程式運算，確定的數值。我們將這 11 項 *IC* 著色數利用 OEIS 網站(參考資料一)查詢。

2. 查詢的結果恰有 1 組數列與這 11 項相符，而該數列的說明為”有 *n* 個點(nodes)的 Graceful Graph 的邊數(edge)最大值”。但目前我們對 Graceful Graph 並無太深刻的了解，因此我們暫時不將研究重點放於此。希望以後能夠一探其究竟。

(四) 下界的推測

1. 我們觀察到，自 n=8 起，都有一組形如(1,2,3,7,7,...,7,4,4,1)的解，因此我們於此提出猜測:對於 n≧8，(1,2,3,7,7…7,4,4,1)為一組 *IC* 著色，其總和為 $7n-27$。事實上，只要中間的 7 超過 1 個時，它一定是一種 *IC* 著色。說明如下:

   欲證:(1,2,3,7,7,…7,4,4,1)，其中 7 有 x 個($x\geq 2$ ， $x\in\mathbb{N}$)，為 *IC* 著色
   即必須要從連通子圖中找出從 1 到 k 的連續正整數都能夠被表示，那麼 k 就是一個 *IC* 著色數的下界，即 $7x+15$。
   顯然，從 1,2,3,…9 都能輕易的表示出來，而:
   10=7+3，11=7+4，12=7+3+2，13=7+3+2+1，14=7+7，15=7+4+4，16=7+4+4+1
   從 17 開始，只要將上列數都再加上一個 7，即可再產生 7 個值，因此，我們可以類推到將中間的 7 取到 $x-1$ 個，此時從 1 到 $9+7(x-1)=7x+2$ 都已經表示。接著要從 $7x+3$ 表示到 $7x+15$，顯然只要中間的 x 個 7 全取，再搭配兩側的數字，就能夠輕易地表達完畢。

   故(1,2,3,7,7,…7,4,4,1)是一個 P_n 的 *IC* 著色，所以我們可以推斷:

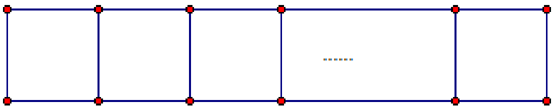
   $\boxed{M(P_n)\geq 7(n-6)+15=7n-27, n\geq 8}$，而目前可以確定的是，$n=8$、$9$、$10$、$11$ 時，此式取等號。

2. 我們再觀察到，解列表中也出現過數次形如(1,3,6,…,6,2,3,2)的解，仿照上述驗證出其為 *IC* 著色。因此又得到: $\boxed{M(P_n)\geq 6(n-5)+11=6n-19, n\geq 6}$。

   且 $n=6$、7、8 時，此式取等號。

3. 綜合上述，目前我們推斷 n 有不同的取值範圍時，有不同的數字配置。從前兩個例子來看，中間應有一連串相等的數字，而該數字為何可能和 n 的取值有關，另外兩側數字的配置，也將影響其結果。目前仍在盡力研究此部分。目標為將其通式導出。

四、$P_2 \times P_n$ 的研究



(一) 連通子圖數

$P_2 \times P_n$ 的連通子圖數相當的複雜，我們無法求出其公式。我們於 OEIS 網站中，找到了以下遞迴式:

以 a_n 代表 $P_2 \times P_n$ 的連通子圖數

1. $a_1 = 3$ ； $a_2 = 13$ ； $a_3 = 40$ ； $a_4 = 108$

2. $a_n = 2a_{n-1} + a_{n-2} + 4n - 1$ ， $n > 2$

3. $a_n = 3a_{n-1} - a_{n-2} - a_{n-3} + 4$ ， $n > 3$

4. $a_n = 4a_{n-1} - 4a_{n-2} + a_{n-4}$ ， $n > 4$

5. $a_n = \dfrac{7+5\sqrt{2}}{4}(1+\sqrt{2})^n + \dfrac{7-5\sqrt{2}}{4}(1-\sqrt{2})^n - 2n - \dfrac{7}{2}$

這些關係式，期望在未來能做出完整的證明。

(二) *IC* 著色數

第 50 屆科展作品「圈圈相連到天邊」(參考資料二)文末提出了一組 $P_2 \times P_n$ 的數值配置法如下:

圈圈相連到天邊的遞迴																	
	1	4	10	30	46	76	218	670	1030	1700	4866	14958	22990	37948	108618	333886	513174
	2	6	10	16	46	142	218	360	1030	3166	4866	8032	22990	70670	108618	179288	513174
IC著色數	3	13	33	79	171	389	825	1855	3915	8781	18513	41503	87483	196101	413337	926511	1952859

其在 n=2 時，得到了 *IC* 著色數最大值 13。

另外，在 99 年師大附中校內科展作品「有趣的撕郵票問題」(參考資料四)中，提出了一組較佳的遞迴數值配置:

有趣的撕郵票問題的遞迴																	
	1	2	15	23	38	107	329	505	834	2387	7337	11277	18614	53279	163777	251721	415498
	4	6	8	23	69	107	176	505	1553	2387	3940	11277	34665	53279	87944	251721	773777
IC著色數	5	13	36	82	189	403	908	1918	4305	9079	20356	42910	96189	202747	454468	957910	2147185

其在 n=3 時，得到了最大值 36。

而我們便思考，若在 n=4、5 等數字時得到最大值，是否能夠找出更佳的遞迴配置方法?於是，我們再次利用電腦程式，先找出 n=4 時的最大值，之後便固定前 2×4 的數值找出 n=5 的數值，依此類推，得到以下幾組數值配置。

1. 在 n=4 時，得到最大值 86 的數值配置

我們找到的遞迴(一)																	
	1	1	3	33	52	85	241	742	1139	1881	5383	16546	25431	41977	120151	369338	567663
	7	6	16	19	52	156	241	397	1139	3502	5383	8885	25431	78174	120151	198325	567663
IC著色數	1	2	34	86	190	431	913	2052	4330	9713	20479	45910	96772	216923	457225	1024888	2160214

此配置方法，雖然在 n=3 以前，*IC* 著色數不如上述 2 組解，但自 n=4 後，*IC* 著色數皆為現有資料中最佳數值。

n=5 以後，我們尚未確定其最大值，在此使用目前最大值進行運算。

2. 在 n=5 時，得到目前最大值 193 的數值配置

我們找到的遞迴(二)																	
	1	1	3	33	58	107	165	489	1516	2329	3845	10997	33804	51953	85757	245461	754532
	7	6	16	19	49	107	324	489	813	2329	7152	10997	18149	51953	159704	245461	405165
IC著色數	1	2	34	86	193	407	896	1874	4203	8861	19858	41852	93805	197711	443172	934094	2093791

3. 在 n=6 時，得到目前最大值 419 的數值配置

我們找到的遞迴(三)																	
	1	1	3	33	58	87	226	698	1063	1761	5037	15476	23789	39265	112393	345492	531013
	7	6	16	19	49	139	226	365	1063	3276	5037	8313	23789	73128	112393	185521	531013
IC著色數	8	15	34	86	193	419	871	1934	4060	9097	19171	42960	90538	202931	427717	958730	2020756

在這數組數值中，我們所找到的第一筆數據為目前最佳的解。

令第 n 行上下二數分別為 a_{n1}、a_{n2} ，則該配置方法的規律為:

前 4 行如上圖所示，而在第 5 行之後，對於第 n 行至 n+3 行，其中 $n \equiv 1 \pmod 4$ ，用以下方法填入

第n行	第$n+1$行	第$n+2$行	第$n+3$行
$\boxed{a_{(n-1)1}+a_{(n-1)2}}$	$\boxed{a_{(n-1)1}+a_{n1}}$	$\boxed{a_{(n+1)1}+a_{(n+1)2}}$	$\boxed{a_{n1}+a_{n2}+a_{(n+2)1}+a_{(n+3)2}}$
$\boxed{a_{(n-1)1}+a_{(n-1)2}}$	$\boxed{a_{(n-2)1}+a_{(n-2)2}+a_{n2}+a_{(n+1)1}}$	$\boxed{a_{(n+1)1}+a_{(n+1)2}}$	$\boxed{a_{(n+1)2}+a_{(n+2)2}}$

以上述這 5 組配置來看，我們找到的第一組遞迴數值配置為目前現有資料中最佳的一組配置。而我們目前嘗試證明這組數值配置方法為 *IC* 著色，但目前困難在於其 *IC* 著色數的通式尚未找出，因此證明時無從下手。希望未來能夠解決。

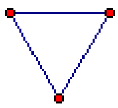
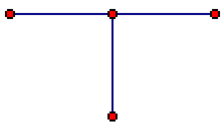
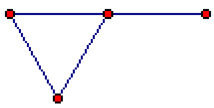
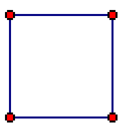
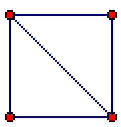
五、利用點與連接線的數量來探討

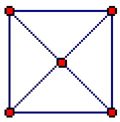
由於以上圖形均沒有得到明顯的結論，我們再朝另一個方向開始研究。我們固定節點數和連接線數，找出其所有可能圖形的連通子圖數與 *IC* 著色數，利用回歸直線的方式，嘗試找出關係式

以下以(x,y)表示「有 x 個點，y 條連接線的圖形」
例如:

 為(3,2)

整理之後，我們期望朝著 *IC* 著色數和連通子圖數的比值 R 的關係進行研究。希望透過統計來預測更多相關值

點與連接線數	可能的圖形	所有可能圖形中 IC 著色數的最大值	IC 著色數與連通子圖數的比值 R
(1,0)		1	1
(2,1)		3	1
(3,2)		6	1
(3,3)		7	1
(4,3)		10	0.909
(4,4)	 	13	1
(4,5)		13	0.929

(4,6)		15	1
(5,4)	略	18	0.883
(5,5)	略	21	0.95
(5,6)	略	23	0.945
(5,7)	略	27	0.953
(5,8)	略	27	0.965
(5,9)	略	29	0.967
(5,10)	略	31	1

但目前我們所求得的回歸直線，其相關係數均不高，因此現在仍無定論。

六、電腦程式演算法簡介

(一) 連通子圖數(舉 P_4 為例)

程式跟我們一樣用窮舉的方式找解,它並沒有比較聰明,但他算得比我們快上好幾倍，也比我們徹底，不會遺漏，程式的演算法是這樣的:

我們先假設我們輸入

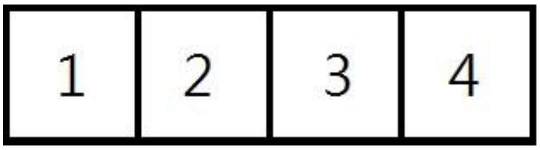
4

那麼電腦它會知道我們輸入九個連接點，再輸入

1 2

2 3

3 4



代表每個點的連接關係，第 1 個點與第 2 個連結，第 2 個點與第 3 個連結，第 3 個點與第 4 個連結。

接著電腦它要先算所有的連通子圖，他會把所有子圖列出並檢查: (在電腦內的表示方式，1 是有，0 是沒有)

1000 0100 1100 0010 1010 0110 1110 0001

1001 0101 1101 0011 1011 0111 1111

然後我們要想像把它變成一個 P_4 的郵票，但我們如何讓他知道這種子圖合不合題意?我們的方法是用 DFS(深度優先) 確定是否為連通子圖。

當他檢查完而且那確定是連通子圖後，它會利用矩陣把它存起來。

(二) *IC* 著色

接著就會進入窮舉數值配置的步驟，我們先要求輸入最大值不可能的數，預設是連接子圖數+1，接著它會先舉出可能的盤面(放到完全圖內必定能連續 1~所有盤面和)。接著他會打亂一個個放入連通子圖確定是否有連續 1~盤面和。

七、未來展望

(一) 將 P_n 與 $P_2 \times P_n$ 的問題徹底解決。

(二) 推廣至其他的圖形，如三角形、六邊形的郵票，或星狀圖、樹狀圖等 *IC* 圖。

(三) 推廣到立體的空間中。

(四) 以數學論證方式證明目前用程式找出的數值。

(五) 程式演算法優化

伍、研究結果

一、 P_n 連通子圖數= $\dfrac{n(n+1)}{2}$

二、 $M(P_n) \geq 6n-19, n \geq 6$ ， $M(P_n) \geq 7n-27, n \geq 8$

三、一組 $P_2 \times P_n$ 的數值填入法，目前現有資料中最佳。

我們找到的遞迴(一)																			
	1	1	3	33	52	85	241	742	1139	1881	5383	16546	25431	41977	120151	369338	567663
	7	6	16	19	52	156	241	397	1139	3502	5383	8885	25431	78174	120151	198325	567663
IC著色數	1	2	34	86	190	431	913	2052	4330	9713	20479	45910	96772	216923	457225	1024888	2160214

陸、討論

一、目前正在討論 $M(P_n)$ 的下界，是否能夠表為 n 的二次式(即中間填入的連續數字能否以 n 表示)

二、正在研究 $P_2 \times P_n$ 的連通子圖遞迴式與填入數值遞迴為 *IC* 著色的證明。

三、目前得到的點與連接線數之間似乎沒有太大關係。

柒、結論

一、找到兩組 $M(P_n)$ 的下界: $7n$-27、$6n$-19

二、發現單格最大值不超過連通子圖數的一半

捌、參考資料及其他

一、 OEIS 整數數列線上大全

(一) $P_2 \times P_n$ 連通子圖數:OEIS:A059020

(二) P_n 的 *IC* 著色數(與 Graceful Graph 有關):OEIS:A004137

二、 鄭晏奇、楊翔雲、黃紹宸、李育霖:民國 99 年:第 50 屆中小學科學展覽會:高中 數學組作品「圈圈相連到天邊」，頁 23

三、 Ebrahim Salehi 、 Sin-Min Lee 、 Mahdad Khatirinejad ,2005/8/28, Discrete Mathematics, IC-Colorings and IC-Indices of graphs, Pages297~310

四、 呂映霆: 99 學年度第 37 屆國立臺灣師範大學附屬高級中學科學展覽會:數學組作品「有趣的撕郵票問題」，頁 3~4

五、 電腦程式碼

```
#include<stdio.h>
#include<stdlib.h>

int node_number=0;
long subgraph_count=0;
int subgraph_connect[20971520];//node_number 最大*subgraph_count 最大
(20*2^20=20971520)

void Node_connect_input(int node_connection[400]){
    for(int point1=1,point2=1;point1 && point2;scanf("%d %d",&point1,&point2)){
        if(point1==point2||point1>node_number||point2>node_number) continue;

        node_connection[(point1-1)*node_number+point2-1]=1;//建立連接表格
        node_connection[(point2-1)*node_number+point1-1]=1;

    }
}

void DFS(int node_connection[400],int possible_subgraph[20],int here){
    possible_subgraph[here]=-1;
    for(int next_node=0;next_node<node_number;next_node++){//找路(能走的路
&&沒走過的路)

if(node_connection[here*node_number+next_node]&&possible_subgraph[next_node]>0){
            DFS(node_connection,possible_subgraph,next_node);
        }
    }
}

int Subgraph_test(int node_connection[400],int possible_subgraph[20]){
    int yes=1;
```

```
		for(int here=0;here<node_number;here++){//看到有點就開始走
			if(possible_subgraph[here]){
				DFS(node_connection,possible_subgraph,here);
				break;
			}
		}

		for(int i=0;i<node_number;i++){//檢查有沒有走完
			if(possible_subgraph[i]>0){
				yes=0;
				break;
			}
		}

		if(yes){//是的話要存起來
			for(int i=0;i<node_number;i++){
				if(possible_subgraph[i])
subgraph_connect[subgraph_count*node_number+i]=1;
			}
		}

		for(int i=0;i<node_number;i++){//矩陣換回
			if(possible_subgraph[i]<0) possible_subgraph[i]=1;
		}

		return yes;
	}

	void Subgraph_search(int node_connection[400]){
		int possible_subgraph[20];
		for(int i=0;i<node_number;i++) possible_subgraph[i]=0;
possible_subgraph[0]=1;

		while(possible_subgraph[node_number-1]!=2){

			if(Subgraph_test(node_connection,possible_subgraph)) subgraph_count++;
```

```
				possible_subgraph[0]++;
				for(int i=0;i<node_number-1;i++){//矩陣進位
					if(possible_subgraph[i]==2){
						possible_subgraph[i]=0;
						possible_subgraph[i+1]++;
					}
					else break;
				}
			}
		}

		/*-----------------------以上為尋找連通子圖數之所需附函式-----------------------*/

		long ic_color_order[20];
		long ic_color_disorder[20];
		long half_subgraph_count=0;
		long ic_color_max=0;
		int answer_count=0;//解的數量

		void Result(){
			int arry[subgraph_count]; for(long i=0;i<subgraph_count;i++) arry[i]=0;
			long one_total=0;

			for(long i=0;i<subgraph_count;i++){	//加上
				one_total=0;
				for(int j=0;j<node_number;j++){
					if(subgraph_connect[i*node_number+j])
one_total+=ic_color_disorder[j];
				}
				arry[one_total-1]++;
			}
			long real_max=0;//看是否有連續 1~整個圖的合
			for(int i=0;arry[i]&&i<ic_color_max;i++) real_max++;

			if(real_max==ic_color_max){//有:輸出盤面
				printf("\n");
				for(int i=0;i<node_number;i++) printf("%d ",ic_color_disorder[i]);
				printf("最大:%d",ic_color_max);
```

```
            answer_count++;
        }
    }


    void Make_disorder_ic_color(int there,long used[20]){//重新排列
        int past_one=0;

        for(int i=0;i<node_number;i++){
            if(used[i]||(past_one==ic_color_order[i])) continue;//這個數用過

            ic_color_disorder[there]=ic_color_order[i];//填數字
            if(there==node_number-1){//最後 1 格
                Result();
                return;
            }

            past_one=ic_color_order[i];

            used[i]++;
            Make_disorder_ic_color(there+1,used);//填下一個
            used[i]--;
        }
    }

    long little_count_1(int n){//只是要計算 2^(n+1)
        long answer=2;
        for(int i=0;i<n;i++) answer*=2;
        return answer;
    }
    //system("PAUSE");

    int Make_order_ic_color(int there,long past_one,long past_all){//做出有順序的可能
的數列
        if(ic_color_max-past_all<past_one*(there+1)) return 1;//看數字有沒有太大

        if(there==0){
            long used[20]; for(int i=0;i<node_number;i++) used[i]=0;//還沒被抓走的先
```

```
做出來並歸零
            ic_color_order[0]=ic_color_max-past_all;
            Make_disorder_ic_color(0,used);
            return 0;
        }

        int i=0;
        if(past_one>(ic_color_max+1)/little_count_1(there-1)-past_all-1) i=past_one;
        else i=(ic_color_max+1)/little_count_1(there-1)-past_all-1;

        for(;i<=past_all+1;i++){
            ic_color_order[there]=i;
            if(Make_order_ic_color(there-1,i,past_all+i)) return 0;//1 表示前面數字舉
的已經太大了，該收手了
        }
        return 0;
    }

    void Try_IC_index(){                    //嘗試 IC-iindex 是否為此數
        for(ic_color_max--;ic_color_max>0;ic_color_max--){
            ic_color_order[node_number-1]=1;
            Make_order_ic_color(node_number-2,1,1);//找盤面(先找排序好的)
            if(answer_count) break;//有了!!
            else printf("\nmax 不可能為  %d",ic_color_max);
        }
    }

    int main(){
        int node_connection[400];
        for(int i=0;i<400;i++) node_connection[i]=0;//點與點的連通方式
        for(int i=0;i<node_number*subgraph_count;i++) subgraph_connect[i]=0;

        scanf("%d",&node_number);//此圖節點數輸入
        if(node_number>20){
            if(node_number==509) printf("hellow are you call me?\n");
            printf("太大瞜，請小於等於 20");//以免溢值
            return 0;
```

```
    }
    Node_connect_input(node_connection);//連接方式輸入

    Subgraph_search(node_connection);//找連通子圖

    ic_color_max=subgraph_count;//稍稍處理一下需要的數
    half_subgraph_count=subgraph_count/2+1;

    printf("\n 撕法:%ld",ic_color_max);
    printf("\n 請輸入已知 max 最大不可能為(預設為  %ld ): ",++ic_color_max);

    long scanf_max;
    scanf("%ld",&scanf_max);
    if(scanf_max<=ic_color_max) ic_color_max=scanf_max;
    else printf("找碴阿～不理你ㄌ!");

    Try_IC_index();

    printf("\n 共  %d  組解\n",answer_count);
    system("PAUSE");
    return 0;
}
```