臺北市 108 年度中等學校學生科學研究獎助
研究計畫說明書封面

組 別： 高中職組

科 別： 數學科

計畫名稱：力爭上郵

摘　要

本研究主要探討 1×n 或 2×n 排列的矩形郵票中，各種面值分配下，連續地撕下郵票，使的撕下的郵票面值總和為 1 到 k 的連續正整數。求 k 的最大值，及其面值分配。

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。像是可以將各個郵票想成是一個觀光勝地中的各種景點，上面的面值代表其可能的花費，那麼景點經過排列過後就可以使的各種預算的觀光客都可以在連通的路線中盡情的消費。小小的一頁郵票上竟然有著這麼多耐人尋味的細節，此點令我興奮不已。

貳、研究目的及研究問題

一、研究目的

探討 1×n 或 2×n 排列的矩形郵票的各種解。

二、研究問題

1×n 或 2×n 排列的矩形郵票在各種面值分配下，使其連通的郵票子集合之郵票面值總和為 1 到 k 的連續正整數。求 k 的最大值，及其面值分配。

參、研究設計

一、郵票問題

定義相關名詞：

(一)、郵票：可以填入面值的矩形。每張大小與尺寸皆相同。

(二)、連通：即指有共用邊。

(三)、連通子集合：在本問題中指一郵票集合，其所有郵票都至少與一張郵票連通。

二、1×n 郵票的研究

(一)、連通子集合數

我們可以很容易地得到 1×n 的郵票的子集合數為$\frac{1}{2}n(n+1)$，因為，如果只撕 1 個，則有 n 種撕法，如果撕 2 個，則有 n-1 種撕法…如果撕 n 個，則有 1 種撕法。

所以 1×n 的郵票的子集合數為$\frac{1}{2}n(n+1)$。

(二)、k 的最大值的研究

這裡指的 k 的最大值為 1×n 排列的矩形郵票在各種面值分配下，使其連通的郵票子集合之郵票面值總和為 1 到 k 的連續正整數。其中的 k 的最大值，根據程式的演算，可以做出下表。程式請見目錄。

n	k	對應的排列方法(只列一種)
1	1	(1)
2	3	(1, 2)
3	6	(1, 3, 2)
4	9	(4, 1, 2, 6)
5	13	(6, 1, 2, 2, 8)
6	18	(5, 2, 6, 3, 1, 14)

(三)、不同演算法對計算時間的影響

目前僅有一種演算法的優化，在基礎的暴力破解上，進一步改良。

1. 基礎的暴力破解(枚舉法)

每一種可能跑遍，最後排序，找出最大的 k。

可以將時間花費做成下表 n = 1~5

n	計算所花費時間(秒)
1	0.0*
2	0.0*
3	0.0156
4	0.1670
5	17.2693

*註：時間過小不易測量

2. 剪枝

在得知當前組合不可能實現時，跳過此組合進到下一可能。

可以將時間花費做成下表 n = 1~5

n	計算所花費時間(秒，取小數點下四位)
1	0.0*
2	0.0*
3	0.0156

4	0.0631
5	5.6868

*註：時間過小不易測量

3.

(四)

三、2×n 郵票的研究

(一)、連通子集合數

(二)、k 的最大值的研究

四、m×n 郵票的研究

(一)、連通子集合數

因為一頁郵票的 k 值必小於等於其連通子集合數，且此數在基礎的暴力破解中(枚舉法)做為每一面值的可能數。故此值十分重要。

求值的程式請參考肆、參考資料及其他。

肆、研究結果

伍、討論

陸、結論

柒、參考資料及其他

一、程式碼

(一)、1×n 郵票最大 k 值求解程式：(最佳版本)

```
import math
import time

record_file_name = f'record_{str(time.time())}.txt'
output_file_name = f'output_{str(time.time())}.txt'

# record = {}

n = 6
```

```
print('1 1')

while True:

    start_time = time.time()
    # n = int(input('n?'))
    n += 1
    # n = 6

    # p = int(input('stamp max?'))
    p = int(n * (n + 1) / 2)

    stamps = [0] * n

    record = []
    k_list = []
    k_dict = {}

    for i in range(int(math.pow(p, n))):

        num_pos = 0
        for j in stamps:    #  下一種面值分配
            stamps[num_pos] += 1
            if stamps[num_pos] == p + 1:
                stamps[num_pos] = 1
                num_pos += 1
            else:
                break

        for j in stamps:    # must be one 1 stamp
            if j == 1:
```

```
                break
        else:
                continue    # no 1 in stamps

        record[i] = []

        for j in range(n):
                for r in range(j, n):
                        summary = 0
                        for q in range(j, r + 1):
                                summary += stamps[q]
                        record[i].append(summary)

        record[i].append(p + 1)
        record[i].sort()

        for j in range(len(record[i])):
                x = record[i][j]
                # print(record[n][i]['summary'])
                if x + 1 < record[i][j + 1]:
                        # record[i] = x
                        k_list.append(x)
                        k_dict[x] = i
                        break
        else:
                print('error occur')
                print(record)
                input()

k_list.sort()
# print(record[n]['k_list'])
```

```
        k = k_list[-1]

        f = open(output_file_name, 'a')
        f.write(f'n: {n}, k: {k}, method_num: {k_dict[k]}\n')
        print(k, k_dict[k])
        f.close()

        f = open(record_file_name, 'a')
        f.write(str(record) + '\n')
        f.close()
        print(-start_time + time.time())
        input()
        # print(record)
```

(二)、由面值排序序號回推出面值排序之程式

```
import math

while True:
    n = int(input('n?'))
    pos_num = int(input('pos_num?'))
    p = int(n * (n + 1) / 2)

    stamps = [0] * n

    for i in range(pos_num+1):

        num_pos = 0
        for j in stamps:    #  下一種面值分配
            stamps[num_pos] += 1
            if stamps[num_pos] == p + 1:
```

```
                    stamps[num_pos] = 1
                    num_pos += 1
                else:
                    break

    print(stamps)
```

(三)、求連通子集合程式

```
import math

n = int(input('input n'))
m = int(input('input m'))

counter = [0] * (n * m)


def plus1(big_base_2_num_list, place):
    if big_base_2_num_list[place] == 0:
        big_base_2_num_list[place] = 1
    else:
        big_base_2_num_list[place] = 0
        plus1(big_base_2_num_list, place - 1)


p = 0

for i in range(int(math.pow(2, n * m))):
    plus1(counter, len(counter) - 1)
    print(counter)

    old_counter = counter
```

```
        # checker = [0] * (n * m)

        for j in range(m):
            counter.insert((j + 1) * n, 0)

        counter = counter + [0]*(n+1)

        print(counter)

        for j in range((n + 1) * (m + 1)):
            if j == 1:
                if counter[j + 1] + counter[j + m + 1] == 0:
                    break
        else:
            p += 1

        counter = old_counter

print(p)
```