臺北市 108 年度中等學校學生科學研究獎助
研究計畫申請書封面

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
1	1	(1)
2	3	(1, 2)
3	6	(1, 3, 2)
4	9	(1, 3, 3, 2), (1, 1, 4, 3), (4, 1, 2, 6)
5	13	(1, 3, 1, 6, 2), (1, 1, 4, 4, 3), (6, 1, 2, 2, 8)
6	18	(2, 5, 7, 1, 3, 6), (2, 5, 6, 3, 1, 8), (6, 4, 5, 2, 1, 13), (5, 2, 6, 3, 1, 14)
7	24	(8, 10, 1, 3, 2, 7, 8)

(三)、不同演算法對計算時間的影響

目前演算法的優化的方向是，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，取小數點下四位)
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，減去沒有 1 的面值組合

在得知當前組合不可能實現時，跳過此組合進到下一可能。透過檢查該組合內有無 1，因為若該組合內不包含 1，則必湊不出 1 這個面值。故可以透過先剪去這類組合達到加快程式速度的效果。

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. 剪枝，減去反轉後曾經算過的組合

由於郵票沒有方向，故同一個面值的組合可能被重複計算過 1 到 2 次。減去這類組合的面值可以省下大筆時間。且計算是否算過的時間遠小於後續的計算時間。

n	計算所花費時間(秒，取小數點下四位)
1	0.0*
2	0.0010
3	0.0010
4	0.0450
5	3.7230

*註：時間過小不易測量

可以從整體時間變少發現算法的確有效，而當 n 較小時，時間稍微增加應該是因為多了一個檢查的時間，且因為 n 夠小，故此計算時間大於不做剪枝的時間。但是可以發現當 n 較大時，時間就有明顯的下降了。

令目前單一面值的最大值為 p，則檢測使用的方法是將整列的郵票轉為一個 n 位 p 進位的數字，若此數字倒轉過後比原本小，則代表此組合已經算過了。

詳細的流程為：

(1) ．判斷第 1 位是否比最後一位大，若是的話，代表倒轉過後會比較大。

(2) ．若(1)判斷為是，則跳過此組合，若非，就不跳過，若兩者一樣，就再比下一個 。

以下為此判斷的程式。

```
for j in range(check_times):
    if stamps[j] > stamps[n - j - 1]:
        break   # 跳過這一次
    elif stamps[j] == stamps[n - j - 1]:
```

```
                Continue    # 再比下一個
            else:
                # 這裡要放後續判斷的函式，也就是不跳過的組合會執行的函式
```

4. 不同演算法之間的執行時間比較

n	枚舉法	剪枝，減去沒有 1 的面值組合	剪枝 ,減去反轉後曾經算過的組合
1	0.0*	0.0*	0.0*
2	0.0*	0.0*	0.0010
3	0.0156	0.0156	0.0010
4	0.1670	0.0631	0.0450
5	17.2693	5.6868	3.7230

三、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、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，包含時間測量)

```
import math
import time

n = 2

while True:
    start_time = time.time()
    p = int(n * (n + 2) / 2)
```

```
stamps = [0] * n

k = n
k_list = []
k_dict = {}

check_times = int((len(stamps) - (len(stamps) % 2)) / 2)

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

    for j in range(check_times):
        if stamps[j] > stamps[n - j - 1]:
            break    # Skip this time
        elif stamps[j] == stamps[n - j - 1]:
            continue
```

```
else:
        record = []
        # print(stamps, i)
        for m in range(n):    # Methods of coloring.
                for r in range(m, n):
                        summary = 0
                        for q in range(m, r + 1):
                                summary += stamps[q]
                        record.append(summary)

        record.append(p + 1)    # Make a BIG gap.
        record.sort()

        for m in range(len(record)):
                x = record[m]
                if x + 1 < record[m + 1]:
                        if x > k:
                                # print(k_list)
                                k_list = [str(i) + ':' + str(stamps)]
                                # k_dict = {i: str(stamps)}
                                k = x
                        elif x == k:

                                k_list.append(str(i) + ':' + str(stamps))
                                # k_dict[i] = str(stamps)
                                # print(stamps)
                                # print(k_list)
                        break
        else:
                print('error occur')
                print(record)
```

```
                    input()
                break


        print(n)
        print(k, k_list)
        # print(k_dict)
        f = open('result.md', 'a')
        f.write(f'n: {n}\nk: {k} method: {k_list}\n')
        f.close()
        n += 1
        print(time.time() - start_time)
```

(二)、找面值編號對應面值

```
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)、尋找連通子集數

```
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 counter[j] == 1:
                if counter[j + 1] + counter[j + m + 1] == 0:
                    break
        else:
            p += 1

        counter = old_counter

print(p)
```

(四)、多核心加快版

```
import math
import time
import multiprocessing

n = 5


def find_k(first_stamp):
    global n, p, k_dict, k, k_list, group_stamps, check_times
    stamps = [0] * (n - 1) + [first_stamp]
```

```
for i in range(group_stamps):
    # print(i, stamps)
    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

    for j in range(check_times):
        if stamps[j] > stamps[n - j - 1]:
            break    # Skip this time
        elif stamps[j] == stamps[n - j - 1]:
            continue
        else:
            record = []
            # print(stamps, i)
            for m in range(n):    # Methods of coloring.
                for r in range(m, n):
                    summary = 0
                    for q in range(m, r + 1):
                        summary += stamps[q]
                    record.append(summary)
```

```
                record.append(p + 1)    # Make a BIG gap.
                record.sort()

                for m in range(len(record)):
                    x = record[m]
                    if x + 1 < record[m + 1]:
                        if x > k:

                            k_list = [str(stamps)]
                            k = x
                        elif x == k:
                            k_list.append(str(stamps))
                        break
                else:
                    print('error occur')
                    print(record)
                    input()
                break


if __name__ == '__main__':
    start_time = time.time()
    multiprocessing.freeze_support()

    p = int(n * (n + 1) / 2)
    group_stamps = int(math.pow(p, n - 1))
    k = n
    k_list = []
    k_dict = {}
    check_times = int((n - (n % 2)) / 2)
```

```
pools = multiprocessing.Pool(4)

first_stamp_queue = multiprocessing.Queue()

for i in range(p):
        first_stamp_queue.put(i)

ps_list = []

for i in range(p):
        ps_list.append(multiprocessing.Process(find_k(i)))
        ps_list[i].start()

for i in range(p):
        ps_list[i].join()

print(n)
print(k, k_list)
f = open('result.md', 'a')
f.write(f'n: {n}\nk: {k} method: {k_list}\n')
f.close()
print(time.time() - start_time)
```

伍、研究計畫進度表

工作項目	日期	中華民國109年									
		1月	2月	3月	4月	5月	6月	7月	8月	9月	10月
1	題目解析										
2	1×n 程式主體										
3	發展1×n 更快的算法										
4	研究新的雲端計算載具										
5	降低1×n 的程式硬體使用概率										
6	將1×n 程式時間壓制至可接受範圍										
7	研究2×n 的問題										
8	構造2×n 程式主體										
9	加快2×n 計算速度										
10	研究 k 值與 n 值的關係										
11	研究 n 值與其最佳的配置方式之間的關係										
12	校對，檢查說明書										
13	報名國際科展										

註：請以粗黑筆劃出或以色塊填滿每一工作項目之起迄月份，列數得依實際需求自行增刪。

陸、研究經費申請明細表

編號	項目	規格	單位	數量	單價	總價	說明
1	科展看板海報		組	1	1500	1500	展示作品用
2	行動硬碟	1TB	個	1	1999	1999	備份及存放科展資料
3	碳墨匣		個	2	990	1980	印作品說明書和相關文件
4	彩色墨水匣		個	2	1982	3964	印作品說明書和相關文件
5	影印紙	A4 80 磅	包	1	150	150	印作品說明書和相關文件
6	筆記本	A4 70 頁	本	1	150	150	紀錄有關科展的資料
7	文具(原子筆、立可帶……)		組	1	300	300	紀錄有關科展的資料
8	科展展版	A0 展版	組	1	6000	6000	展示作品用
9	演算法的樂趣(參考書)		本	1	480	480	研究用途

總計：16523