

演算法程式作業二

110403518 林晉宇

一、UVA 1257 Your ways

本次作業題目為 uva 1257 your ways，並且障礙物為獨立，使用兩種方法實作：組合學及動態規劃，透過比較兩者的時間複雜度，產生測資並以圖表呈現兩者的時間差距。題目輸入包含 h, w (地圖大小 ≤ 1000)， k (天數 ≤ 10000)， q (障礙物數量 ≤ 100)。

二、組合學

透過教授上課簡報第六頁的方法去實作， $path[i][j]$ 代表從起點到座標 (i, j) 的方法數，遞迴式如下：

$$path[i, j] = \begin{cases} path[i-1][j] + path[i][j-1], & \text{if no obstacle} \\ path[i-1][j], & \text{if obstacle below} \\ path[i][j-1], & \text{if obstacle on the left} \\ 0, & \text{else if obstacle on both side} \end{cases}$$

這樣的時間複雜度為 $O(KHW)$ ，總共 K 天，每一天都要花 HW 的時間去重新建表。

三、動態規劃

我參考李品雋同學簡報的方法來實作：

1. $block[]$ 儲存障礙物座標 (依照離終點由遠到近排序)。
2. $sub[i]$ 代表障礙物 i 左下角涵蓋的所有障礙物。
3. $dp[i]$ 代表從起點到 $block[i]$ 起始位置並且沒有經過 $sub[i]$ 任一障礙物的路徑數量。
4. $p[(x, y)]$ 代表在無障礙物下，起點到該座標的路徑數量。

遞迴式：

$$dp[i] = \begin{cases} p[block[0].s], & \text{if } i = 0 \\ p[block[0].s] - \sum(dp[j] * p[block[i].s - block[i].t \ (j \text{ in } sub[i])]), & \text{else} \end{cases}$$

Pseudo code:

```
//填 sub

for(i = 0~q)
{
    tmp = [];

    for(j = 0~i-1)
        if(block[i] dominate block[j])
            tmp.push_back(j)

    sub[i] = tmp;
}

//算 dp

int path(i)
{
    q_path=0;

    for(j:sub[i])
        q_path+=dp[j]*p[block[i].s-block[j].t];

    return p[block[i].s] - q_path
}

dp[0] = p[block[0]]; //初始化

for(i = 0~q)
    dp[i]=path(i); //推算 dp
```

時間複雜度： $O(HW+KQ^2)$ ，一開始先用加法原理建出 $p[]$ ，總共 k 天，每天需要：

1. 排序障礙物， $O(Q\log Q)$
2. 填 $sub[]$ ， $O(Q^2)$
3. 推算 $dp[]$ ， $O(Q)$

則每天要花 $O(Q^2)$ 的時間。

四、設計測資

經過前面的實作及時間的推算，方法一為 $O(KHW)$ ，方法二為 $O(HW+KQ^2)$ 而 H, W (≤ 1000)， K (≤ 10000)， Q (≤ 100)，我認為最能看出兩者差異的為變數 K ，所以在設計測資方面， H, W 皆設為最大值， Q 為 1-100 之間的 random 值， K 則從 10 慢慢遞增到 10000。我認為 K 值大到一個範圍時，方法一很快就會超過三秒的限制了。

```
20 H = 1000
21 W = 1000
22 k = 1000
23 for K in range(k, k+1):
24     print(K)
25     f = open("testcase.in", "w")
26     f.write(f"{H} {W} {K}\n")
27     for i in range(K):
28         num = random.randint(1, 100)
29         f.write(f"{num} ")
30         for j in range(num):
31             VorH = random.randint(0, 1) # 0: vertical, 1: horizontal
32             if VorH == 0:
33                 x = random.randint(0, H)
34                 y = random.randint(0, W-1)
35                 f.write(f"{y} {x} {y+1} {x} ")
36             else:
37                 x = random.randint(0, H-1)
38                 y = random.randint(0, W)
39                 f.write(f"{y} {x} {y} {x+1} ")
40         f.write("\n")
41     os.popen("./khw &") # run https://blog.csdn.net/Mr_Li1/article/de
42     os.popen("./dp2 &") # run
```

▲產生測資的 python 程式碼

五、時間比較

我的筆電以 C++ 來實作，每秒大約能跑 $5e8$ 次運算。

K	$O(KHW)$ 秒數	$O(HW + KQ^2)$ 秒數
10	0.157	0.018
50	0.52	0.02
100	1.021	0.056
500	2.57812	0.0625
1000	4.78125	0.09375
2000	10.7188	0.109375
3000	15.4219	0.125
5000	18.8906	0.1875
10000	37.0625	0.25



output.txt	output1.txt
9985 956	9985 956
9986 2340	9986 2340
9987 660	9987 660
9988 1741	9988 1741
9989 1983	9989 1983
9990 2376	9990 2376
9991 440	9991 440
9992 1508	9992 1508
9993 370	9993 370
9994 1028	9994 1028
9995 589	9995 589
9996 1803	9996 1803
9997 561	9997 561
9998 913	9998 913
9999 2252	9999 2252
10000 2150	10000 2150
10001 37.0625	10001 0.25

▲執行 $k=10000$ 測資的結果截圖

六、 結論

根據上述的實驗結果，可以發現當 K 值到 500-1000 之間時，組合學的方法就會開始超時，而動態規劃的方法則一直維持在一秒以內，在 K 值越來越大的情況下，兩者的差距顯著的增加。