**演算法程式作業二**

110403518 林晉宇

1. **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)。

1. **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:**

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。

1. **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:**

**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[t].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(QlogQ)**
2. 填**sub[]**，**O(Q^2)**
3. 推算**dp[]**，**O(Q)**

則每天要花**O(Q^2)**的時間。

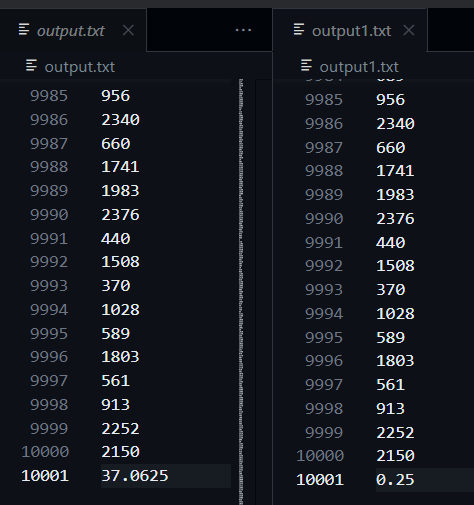
1. **設計測資**

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。

▲產生測資的python程式碼

1. **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 |

▲執行k=10000測資的結果截圖

1. **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。