**演算法程式作業二**

**110502529陳柏燊**

**Q1 演算法實作:**

利用組合學，當前位置組合數 = 來自左邊的組合數加上來自下面的組合數(遇到障礙則不加)

Time complexity

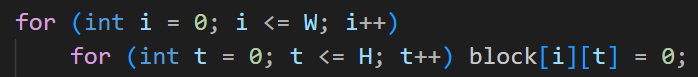
**程式解釋:**



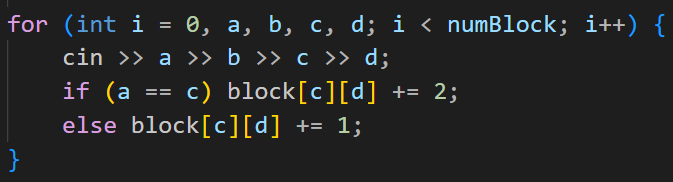
ans[i][j] 代表從起點走到(i,j)的組合數

block[i][j] 代表(i,j)的左或下是否被阻擋

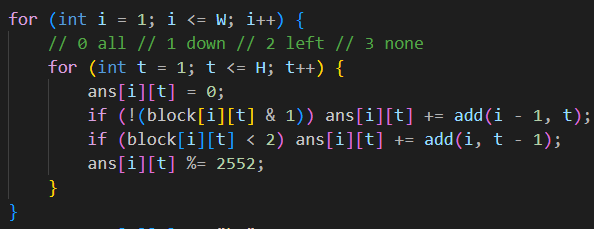
0:都沒被阻擋, 1:左被阻擋, 2:下被阻擋, 3:左和下皆被阻擋



block 初始化 (ans由於會被覆蓋因此不需要初始化)



Block 輸入(0:都沒被阻擋, 1:左被阻擋, 2:下被阻擋, 3:左和下皆被阻擋)



主計算程式，從(1,1)開始計算，透過讀取block確認是否被阻擋，來計算是否加上來自左或下的組合數

**DP 演算法實作:**

dp[i] 表示不經過其他障礙物，直接從起點到第i個障礙物起點的組合數

遞迴式: DP[i] = table[i.fx][i.fy] –{dp[n]\*table[i.fx – n.ex][i.fy – n.ey]} (n 滿足 n.ex <= i.fx and n.ey <= i.fy)

//fx :表示障礙物起點的x ex : 表示障礙物終點的x

//fy :表示障礙物起點的y ey : 表示障礙物終點的y

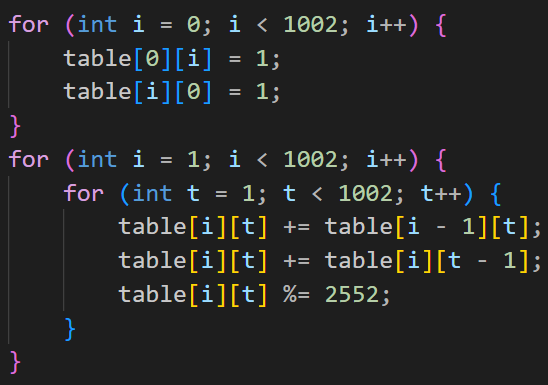
//遞迴式中的大括號內的n是代表所有終點的x和y皆小於等於i起點的x和y的障礙

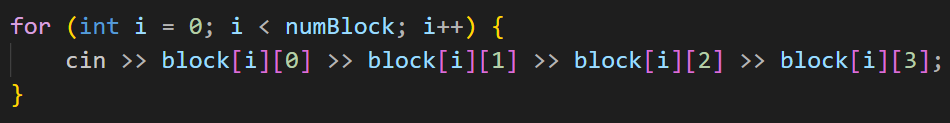
//table[i][j] 代表從起點走到(i,j)且中間不存在任何障礙的組合數(可事先建表)

將一個dummy的障礙放在終點，那麼dp[dummy]就會是答案

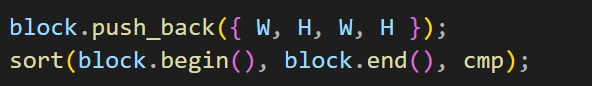
Time complexity

**程式解釋:**

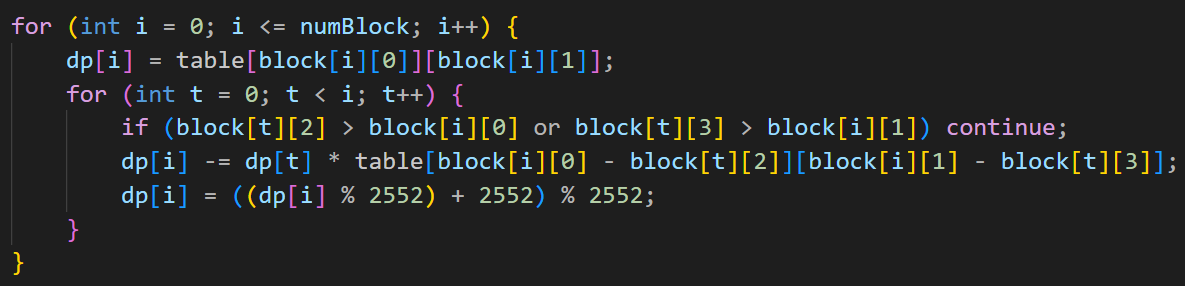
初始化table，並確保mod 2552



輸入障礙物到一個一維vector，element 為array<int,4>



加入dummy 並sort vector 以確保buttom up 的執行順序

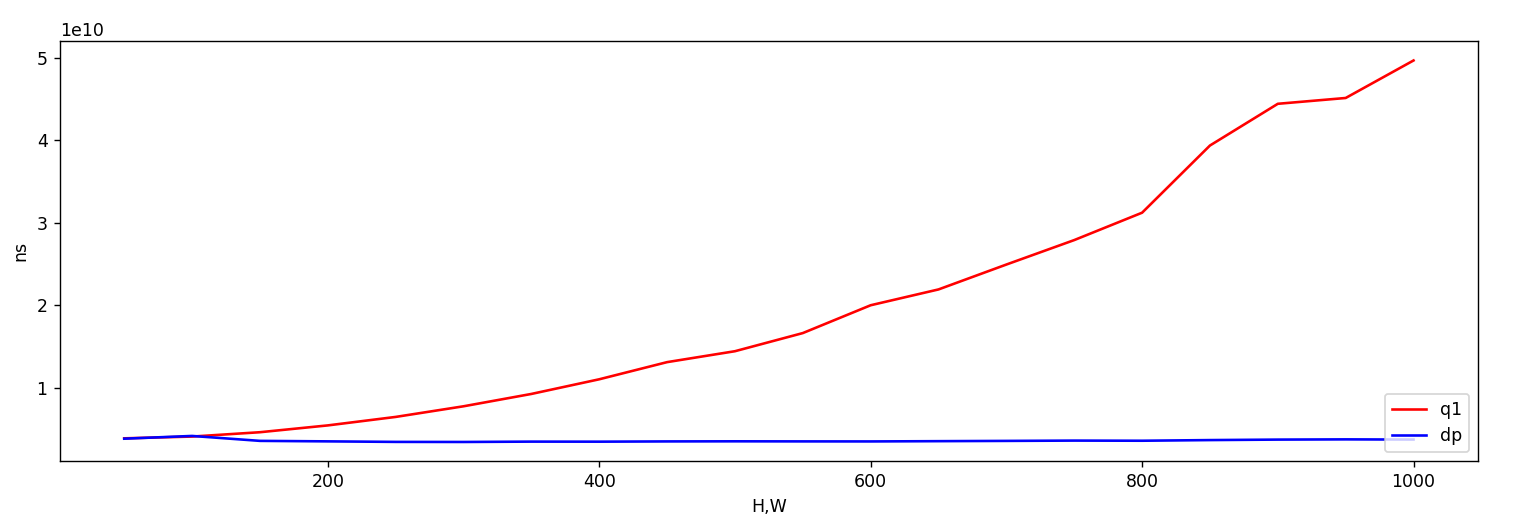


主要計算程式

先把dp[i] 設為table[i.x][i.y]，再扣掉合法的障礙物(遞迴式中定義的)，最後一行是確保dp[i] 不會變成負數

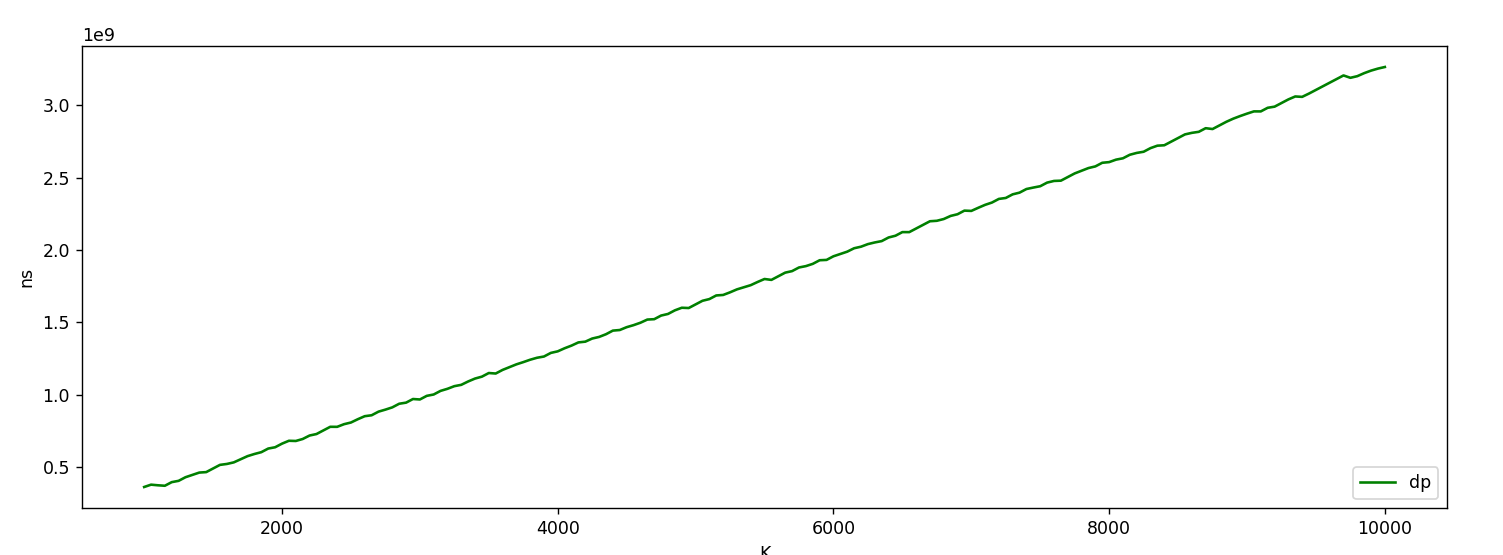
**時間測試和測資設計:**

我一開始是測試K為10000, Q為100, H,W為50 ~ 1000的時間比較



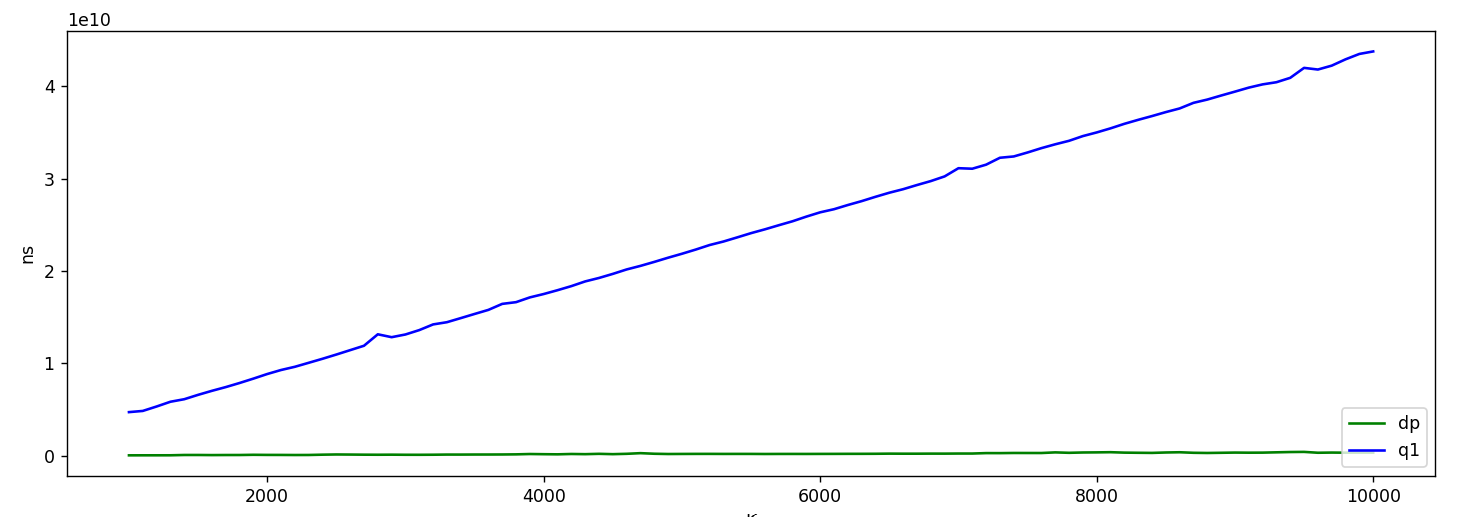
我們可以發現H,W對dp算法基本上沒什麼影響，且H,W = 1000 時q1 作法甚至需要快50秒。但，這時我發現，dp作法在k=10000, Q=100時，耗時會超過3秒(有可能是我電腦比較廢)所以我接下來打算測怎樣得k可以確保dp不會超過3秒

下圖為H,W = 1000, Q = 100, k = 1000 ~ 10000的時間比較



我們可以發現他是一個非常漂亮的斜線(為了確保穩定性我開了冷氣加超強的電風扇吹他…)大概是k = 9000時超過3秒時，我想k = 8000內時該都可以保證3秒內

最後是如何設計出q1一定超時，dp一定過的測資，由時間複雜度我們可以發現H,W影響q1，q影響dp，於是我設計H,W皆為1000，q為10(為了確保題目算法的正確性)，k從1000 ~ 10000的測資



我們可以發現dp從k = 1000 時大約是4秒， k = 10000時大約是40秒，然後dp是大約0.2秒到0.3秒(對，他有增加，只是非常不明顯)，完全不會超過3秒，所以最後的測資就決定是H,W = 1000, K=10000, Q = 10 來卡掉非dp 的算法。

**一些小故事(沒啥重點的東西):**

第二章圖的測資高達快1.5G哇，生成都花了快10分鐘



其實大部分的數據都跑了好幾次，之前沒開冷氣加電風扇的時候，那個圖都給我上下亂跳，超級醜，於是要做報告畫圖的時候，我又全部重跑一次，才有現在這樣漂亮的圖