演算法Final Report(第八組)

**組員**：b97502022張浩軒 b00901024張瑞宇 b00901079陳威宇 b00901086林宏驊

**題目**：國際積體電路電腦輔助設計軟體製作競賽題目E

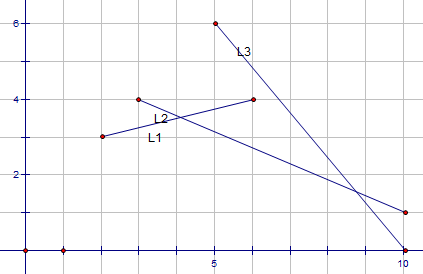
Searching for objects that intersect a specified area in two-dimensional space is commonly referred to as region query. For this problem, a set of simple polygons are given. A simple polygon is the region enclosed by a single closed polygonal chain that does not intersect itself. Thus we do not allow polygons with holes. After reading in the polygons, the program expects five types of query commands and gives the correct answers. Those commands are:

1. Find the number of polygons contained by a given (rectangular) window area.
2. Find the number of polygons interacting with a given window area. Interaction means overlap or touching among polygons.
3. Find the number of polygons interacting with given window that do not overlap with others interacting with the window.
4. Find the number of polygons whose areas are less than a given number.
5. Find the percentage of the given window area that is covered by polygons.

**核心演算法**

掃描線演篹法

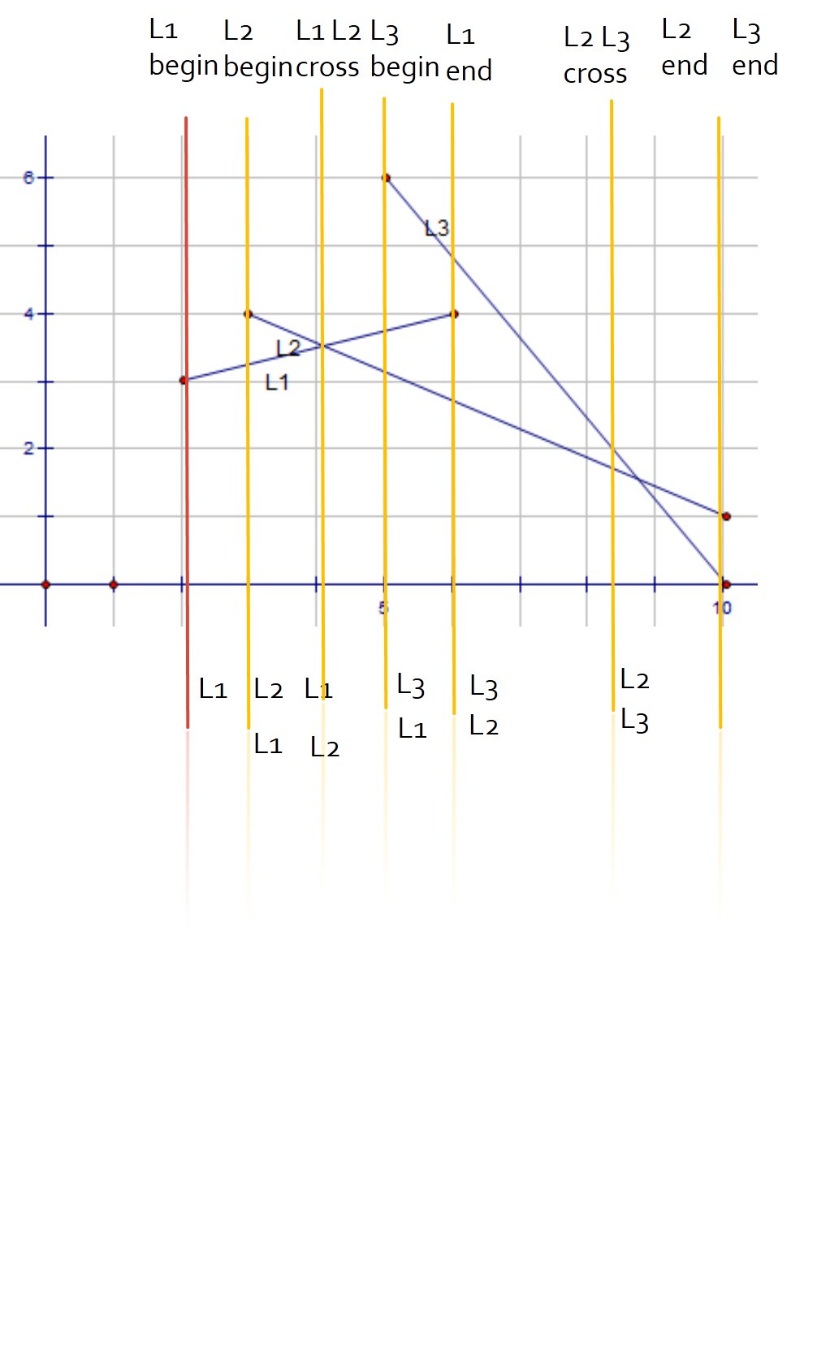
現在平面上有許多線段



現在給定義垂直x軸線，從左一直移到右，稱此線為掃描線，這個掃描線會和線段隨當下的x位置而產生不同的交點，記錄這些交點順序的演算法，通稱掃描線演算法，為計算幾何一個重要而廣泛的演算法。

Bentley–Ottmann algorithm

為掃描線演算法下的一個分支



如上圖，這個演算法會使用兩個資料結構

一為紀錄某x位置下和掃描線上相交的線段，依照其交點高低順序排出一個的線段列表L，結構為tree

一為紀錄會使得線段順序發生變化的”事件”的x位置的事件列表P，結構為priority queue

而會使交點發生變化的事件有三種情況

1. x位置對應到某線段的起點，要將線段插入線段列表中
2. x位置對應到某線段的終點，要將線段移出線段列表中
3. x位置對應到兩線段的相交處，要交換線段在列表中的位置

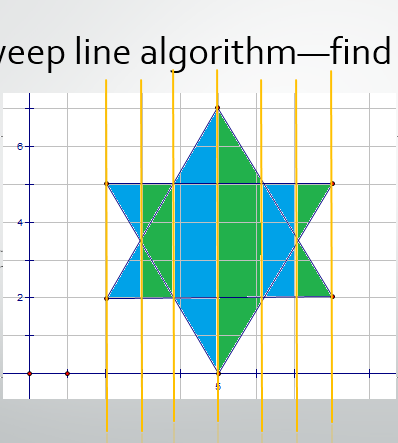
a b情況都是已知，一開始就會在事件列表中

而c情況發生的前提是這兩線段在列表上必然曾經連在一起過，也就是兩線段要相交之前，中間必然不會存在其他的線段(多線共點視為相交於多點)

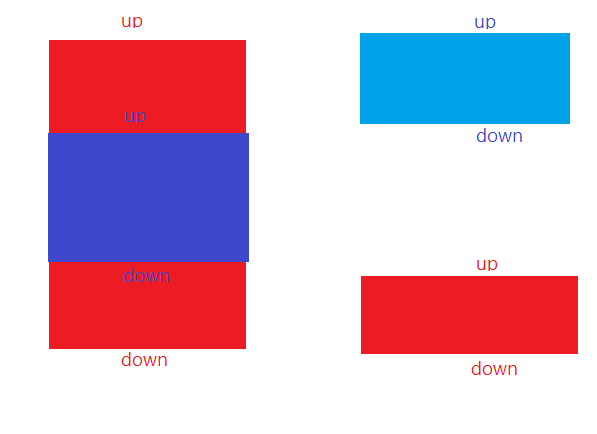
所以我們能透過線段列表建構事件列表，又能透過事件列表更改線段列表，最後能找出所有線段的交點

計算面積的應用

透過找出各個交點，我們能將任何重點情況的多邊形拆成多個梯形的組合，計算面積只要算出掃描線所找到的，點所連成的邊長，用梯形公式即可算出面積。



特別要注意的事，這個演算法無法處理垂直邊的情形，但垂直邊不影響面積大小，故可忽略。然後使用時需要界定出各個邊是多邊形的up(有面積的部分在其下面)或down(有面積的部分在其上面)，才能計算出正確的面積。



**前置作業：**

將輸入檔的各多邊形頂點座標讀入後，建立多邊形(polygons class)的資料結構：

包含面積、所有頂點座標、外接矩形、Edge list(掃描線演算法所需要)

(一)所有頂點座標 vertex list

依據給定的input，順/逆時針去儲存頂點座標，主要是用來解決第一和第二個問題，且當題目要求回傳座標(-verbose)，順序規定要和原本一樣，所以一定要存。

(二)多邊形面積

從(一)得出的座標可以接續使用行列式算出面積。如右圖，取任意頂點作為固定點，依序對其他頂點射出向量，將這些向量依序兩兩外積、相加、除以二，並取絕對值，即可得面積。



(三)外接矩形

利用<algorithm>已經寫好的sort function將頂點座標中找出極右、極左、極上、極下，就是此多邊形的邊界，也就找出外接矩形的上、下、左、右邊的高度或左右的位置。

(四)Edge List

輸入檔給的頂點順序可能是順時針或逆時針，而本方法需統一方向。因此統一為順時針。

1. 判斷順時針或逆時針：

找出最左邊的點，此點會伸出兩邊，指向vertex list中的前一個和後一個。

比較此兩邊的斜率，往順時針方向的邊，其斜率應大於往逆時針方向的邊。如此可確定要以順時針方向走完每個邊，在vertex list中，應該往前走，還是往後走。若往前走所得的邊，斜率較大，往前走是順時針，反之為逆時針。

決定方向後，就依照此方向造訪每個邊。

1. 開始走訪各邊
2. up down邊的判斷：

先走到的點在比較左邊，稱之up邊。反之為down邊。如果是垂直邊，會把他丟棄，後面另外討論。

1. 邊的相連關係：

多邊形的各個邊會有一個指標，和其他邊相連（指向某一個邊）。

依照順時針的順序，up邊連到下一個邊，down邊連到上一個邊。如圖中的up和down邊。



1. 起點終點的判斷：

掃描的起點和終點，必定發生在up邊和down邊交替的地方。以順時針來看，如果是先up在down，則是個終點。反之則是起點。

1. 紀錄起點：

紀錄起點：把所有起點所伸出的up邊和down邊，存成up list和down list於polygon中，並依照起點的x座標，由左至右排序。

1. 終點指到NIL

接到終點的邊，均指向NIL，不再連到其他的邊。

1. 垂直邊的狀況：

對於垂直邊，將其直接跳過。在處理上，等同於把所有的垂直邊丟棄(但頂點仍然保留)，再從最左邊的點開始，順時針造訪每個邊，並處理以上的狀況。

(7)紀錄每條邊的出發點與結束點，還有這條邊指向哪條邊，以及隸屬於哪個多邊形。

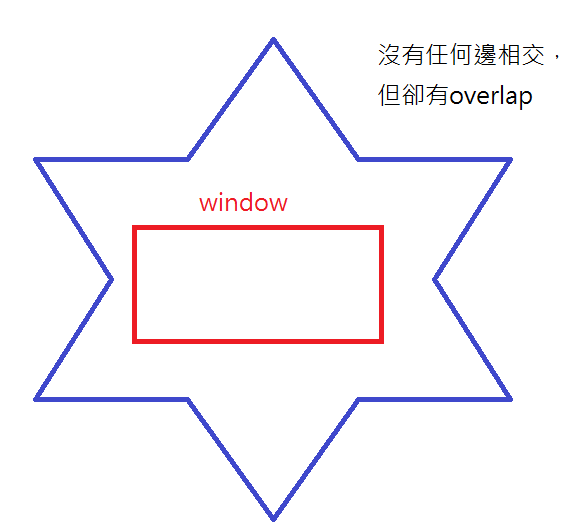
**解決問題一：**

因為我們已經有存每個多邊形的外接矩形，所以只要判斷外接矩形的上下界、左右界是否被給定的視窗(必為長方形)包住即可，故判斷每個多邊形只需要constant time，時間複雜度是(多邊形數量)。



**解決問題二：**

先利用問題一判斷是否多邊形在視窗內，若否，則檢查外接矩形是否與視窗相交，但相交不一定代表多邊形就和視窗相交，接著利用vertex list，從特定的邊開始順/逆時針檢查，檢查是否有和視窗的四邊相交，若又全部都沒相交，最後還要小心是否視窗被包在多邊形內部即可，時間複雜度是O(多邊形總邊數)。



**解決問題三：**

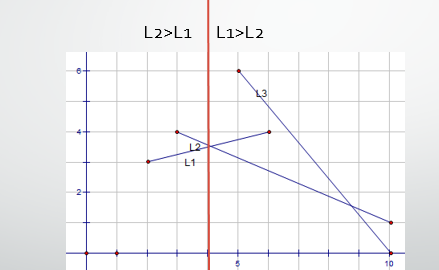
利用掃描線演算來解決問題三和問題五，參考Bentley–Ottmann algorithm的時間複雜度是O((n+k)logn)。(k為交點數,n為邊數)

多創一個Point Class，每個point要存取它的x,y座標(不一定是整數)，還有它的type(有分begin,mid,end,cross)，以及兩條edge。

Begin point就是up edge list每邊的起始點，它指出去兩條edge，分別up edge list起始邊跟down edge list起始邊；mid point就是就是從edge list 起始邊一路指過來的點，兩條edge分別是指向它和指出它；end point則是edge指向null，用來erase edge set；cross point，則是判斷edge set上下兩條是否相交，並且轉換edge set的順序。(其中edge set代表目前掃到的edge，從高排到低)

edge set記錄當下掃到的edge，由高排到低

首先要有priority\_queue<Point\*>，裡面要存begin point、mid point、end point，還有相交產生的cross point，這些point代表接下來要檢查的事件，從x座標小的一路檢查到x座標大的，除此之外還要有set<Edge\*>代表目前在y=k的情況下掃到邊的順序，這些順序在第五題計算面積時十分重要，所以要小心處理，例如遇到cross point時，上下兩邊的edge要交換，然後每插入新的線段，都要檢查是否有上下邊產生新的交點，要存入priority\_queue，一直到所有在queue裡面的point都被pop出來才停止。

遇到cross point所屬兩條egde交換順序

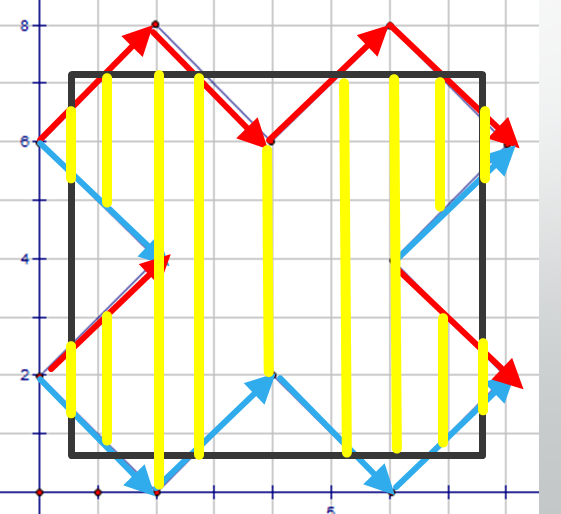
最後每次檢查set<Edge\*>，看相鄰的邊(up就往下看；down就往上看)是否都屬於一個同一個多邊形，若不同，則代表此二多邊形必定overlap，時間複雜度= O(總點數\*每次掃到的邊數)。

**解決問題四：**

利用行列式，已於前置作業時計算完面積，只需要再判斷面積是否小於給定的值即可，時間複雜度是(多邊形數量)。

**解決問題五：**

結構上跟問題三解法很像，一樣從左邊掃到右邊，並利用up、down的關係，去計算出每一個小梯形的面積，最後再加總起來。



大致上都和問題三一樣，但是因為每次碰到交點時，都要去看對應的所有邊，故複雜度為(n+k)n。

**結果：**

官方testcase：

時間（單位：毫秒）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 問題： | 1 | 2 | 3 | 4 | 5 |
| 多邊形數量 |  |  |  |  |  |
| 126430 | 3.4 | 4.4 | 338.5 | 1.7 | 463.4 |
| 159909 | 4.2 | 5.6 | 646.5 | 2 | 831.7 |
| 180378 | 4.2 | 5.4 | 460.9 | 2.2 | 1000.7 |
| 237960 | 5.4 | 7.1 | 664.6 | 3 | 613.56 |
| 423850 | 9.72 | 13.3 | 1264.4 | 4.7 | 2628.4 |
| 582110 | 14.8 | 19.7 | 3483.4 | 7.8 | 3339.4 |

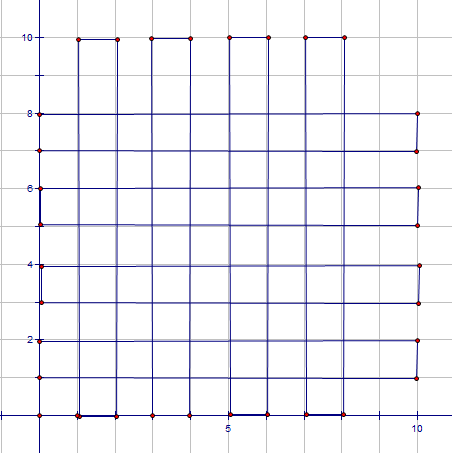
記憶體使用（單位：MB）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 問題： | 1 | 2 | 3 | 4 | 5 |
| 多邊形數量 |  |  |  |  |  |
| 126430 | 103.78 | 105.51 | 274.88 | 447.76 | 633.67 |
| 159909 | 135.56 | 137.29 | 435.05 | 646.48 | 831.73 |
| 180378 | 126.45 | 128.51 | 352.74 | 538.84 | 773.04 |
| 237960 | 146.52 | 148.48 | 404.51 | 699.38 | 908.37 |
| 423850 | 260.24 | 264.66 | 757.12 | 1113.55 | 1538.64 |
| 582110 | 417.38 | 423.81 | 1327.56 | 2322.38 | 3064.51 |

官方給解答的testcase：

前四題可和解答相同。

自行產生的簡單testcase：

1. 柵欄：

poly 4

0 0 10 0 10 1 0 1

poly 4

0 3 10 3 10 4 0 4

poly 4

0 5 10 5 10 6 0 6

poly 4

0 7 10 7 10 8 0 8

poly 4

1 0 2 0 2 10 1 10

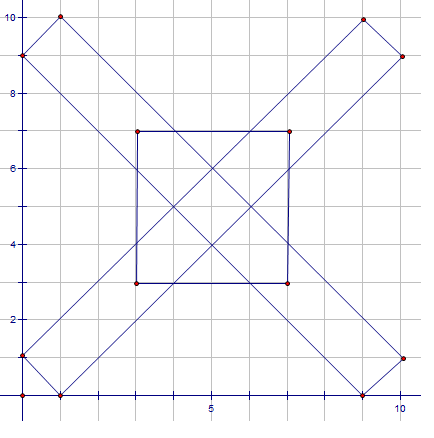
poly 4

3 0 4 0 4 10 3 10

poly 4

5 0 6 0 6 10 5 10

poly 4

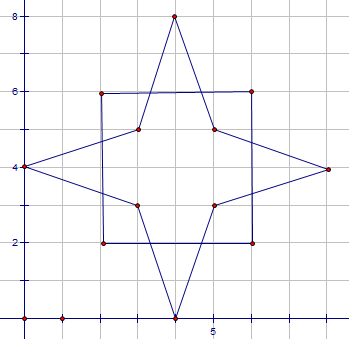
7 0 8 0 8 10 7 10

FIND\_DENSITY\_INSIDE 0 0 10 10

答：64%。可以處理。

2. 交叉

可以處理。

3. 飛鏢與正方形

poly 4

2 2 6 2 6 6 2 6

poly 8

0 4 3 3 4 0 5 3 8 4 5 5 4 8 3 5

FIND\_DENSITY\_INSIDE 0 0 10 10

答：21.3333...，正確。

**待解決的問題**

我們的線段列表使用的資料結構是紅黑樹，這代表其本身的結構是依照各線段的大小關係在維持的。隨著x值的不同，各線段的大小會發生改變(因x對應的y值隨x變化。在”事件”發生時，我們需要更動線段的順序。而實作層面上，是以之前的值找出要更動的線段，拔掉之後以新的值差回去，因此會衍生許多問題:

1. 多線共點

多線共點的時候，需要將共點視為很多點的疊合，其結構可能會發生混亂

1. 重合

我們無法定義重合線之間的交點，當重合的多條線和一條線相交時，交點前後順序會一次變動很多，會造成結構上的混亂

1. 精確度問題

我們求線段對應的y值時是使用該線段的方程式去計算，如果斜率是小數，就可能發生0.33333\*3和1不同這樣的情形，即使同位置可能也會被判斷成不同的大小。

參考資料；

演算法筆記 <http://www.csie.ntnu.edu.tw/~u91029/>

wikipedia <http://en.wikipedia.org/>

分工：

b97502022張浩軒:problem1,problem2

b00901024張瑞宇:problem3

b00901079陳威宇:problem5

b00901086林宏驊:problem4,read inputs