**程式設計研討專題一**

110403518 林晉宇 資工2B

1. **暴力法 O(n^2)**
2. **解釋:** 把每個點跟除了自己的每個點的座標去做比較，這樣需要一個雙層迴圈，時間複雜度為O(n^2)。
3. **Pseudo code:**

struct point { int x, y, z}

int n <- point number

point p[n] <- all point

bool check[n] <- set all to true //最後是true 代表該點為解

for i <- 0 to n do

for j <- 0 to n do

if p[i].x<=p[j].x and p[i].y<=p[j].y and p[i].z<=p[j].z and i!=j

check[i]=false

1. **線段樹 O(nlogn)**
2. **解釋:** 先將所有點按照x從大到小排序，接著將y座標離散化，然後建一個線段樹用來維護z座標的最大值，離散化後的y座標代表線段樹的位置，每次對位置的值更新為z座標的值，要判斷該點是否為maximal point，就看從線段樹y ~ n的位置最大值是否大於該點的z座標，如果大於，則當前這點就不是maximal point。我們用一個for迴圈遍歷所有點，每次對線段樹做一次查詢，故時間複雜度為O(nlogn)。
3. **Pseudo code:**

tree[MAXN] //MAXN=1000000

discrete[n] <- y座標

build\_tree(l ,r, now): //建立線段樹

if l==r: tree[now]=MIN

build\_tree(l,mid,左節點), build\_tree(mid+1,r,右節點) //遞迴建樹

tree[now]=max(左節點, 右節點)

update(pos, val, l, r, now): //更新線段樹

if l==r: return tree[now]

if pos<=mid: update(pos, val, l, mid, 左節點) //遞迴更新

else update(pos, val, mid+1, r, 右節點)

tree[now]=max(左節點, 右節點)

query(L, R, l, r, now):

if L<=l and r<=R: return tree[now]

ll ret=MIN

if L<=mid: ret=max(ret, query(L,R,l,mid,now<<1))

if R>mid: ret=max(ret, query(L,R,mid+1,r,now<<1))

return ret

Main():

sort(p[] 從大到小排序)

sort(discrete[] 從小到大)

discrete.erase() 刪除重複的y座標

build\_tree(1, n, 1) 建線段樹

for i <- 0 to n do

if query( find\_index\_y(p[i].y)) to discrete.size() ) >= p[i].z: check[i]=false

update( find\_index\_y( p[i].y to p[i].z ) )

1. **時間比較**
2. 當點數設為10000點，點隨機產生:

用暴力法: 0.688秒

用線段樹: 0.016秒

1. 當點數設為100000點，點隨機產生:

用暴力法: 0.688秒

用線段樹: 0.016秒

點數差十倍，時間也差大約十倍，而用線段樹明顯比暴力快不少，隨著點數越多，差距越明顯。

* + 產生側資的程式: