threshold的設置：

threshold 是指決定 divide 到什麼程度的閥值

利用程式碼來設計，當divide 到n=threshold時，<=n的點會用暴力法（全部的點兩兩相比較大小），時間複雜度為O(n^2)，以下是我的實驗過程：

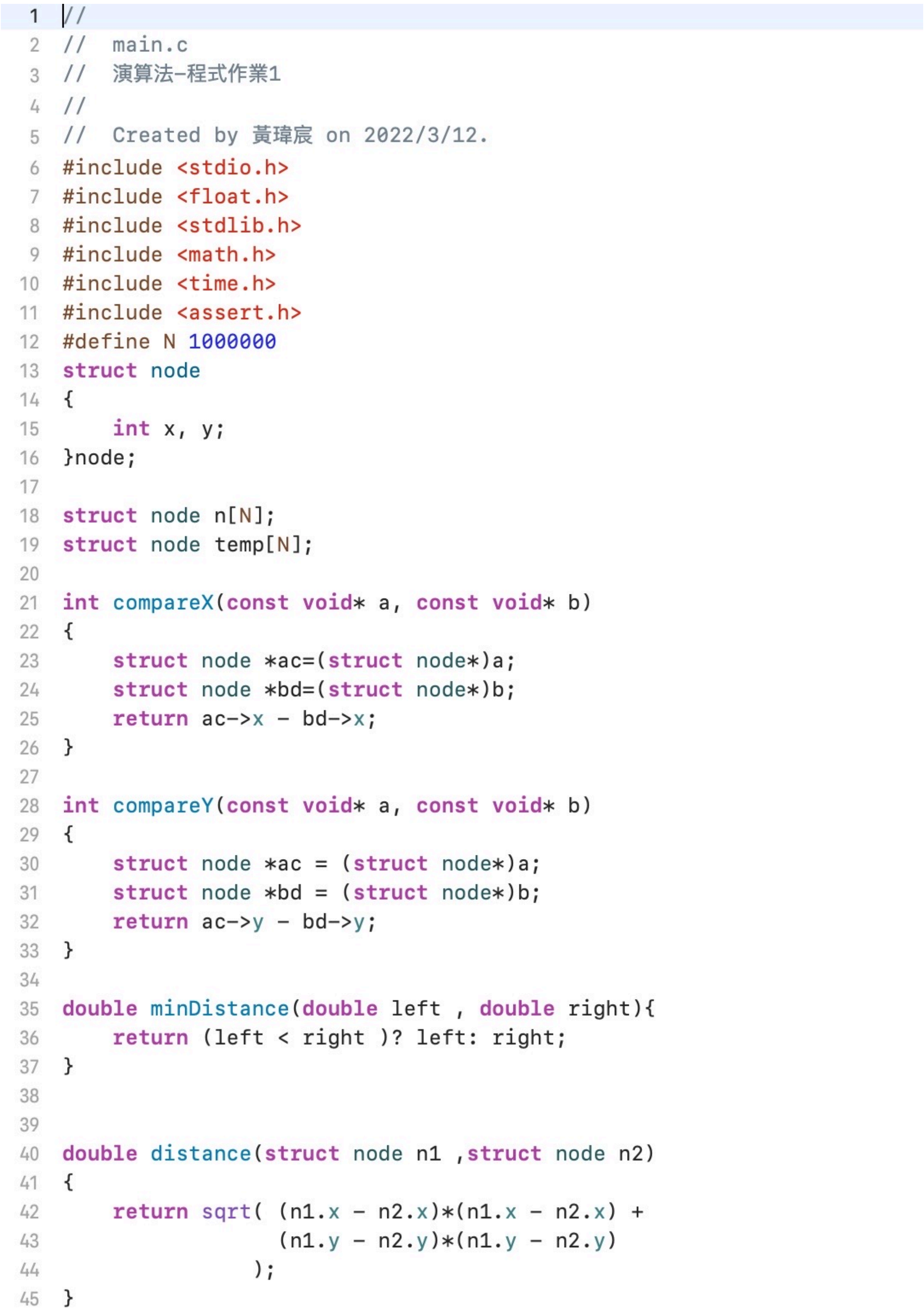
我在原本的程式上做改良，能夠產生一個隨機亂數作為n(範圍是2≦n≦1000000)，再產生兩組隨機亂數當作座標(-10000≦Xi≦10000)(-10000≦Yi≦10000)，然後利用迴圈，讓threshold帶不同的數字進去，並測試執行時間，最後得出執行時間最少時所設置的threshold。

因為在OlineJudge上有測試過，得到以下表格

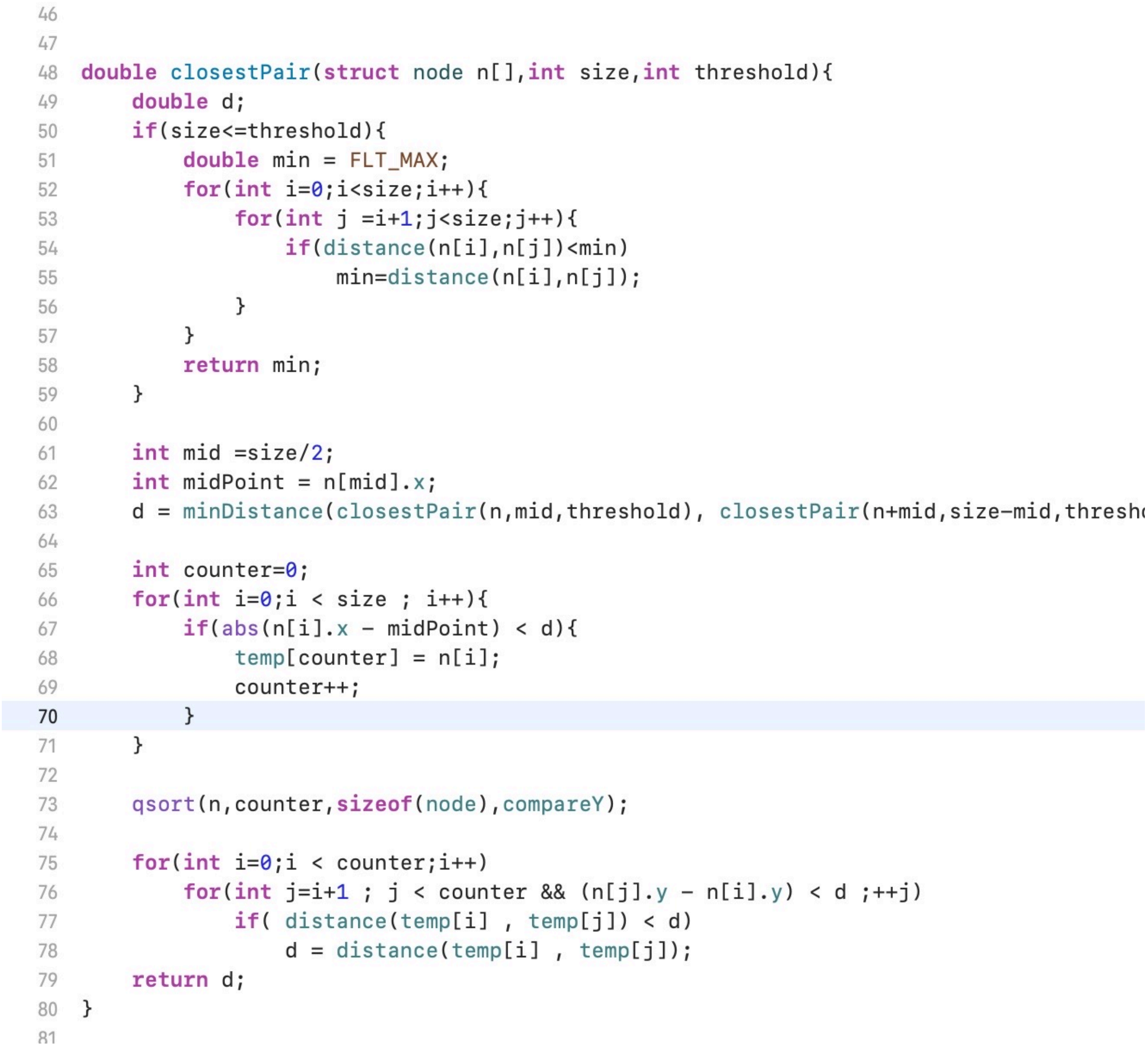
Threshold	Time(ms)
2	738
3	724
4	734
5	725
6	711
10	702
50	696
60	692
70	752
75	728
100	748
150	848
200	856

所以我的迴圈是把threshold從2到150跑一次，最終的結果為**threshold**要設置成**79**

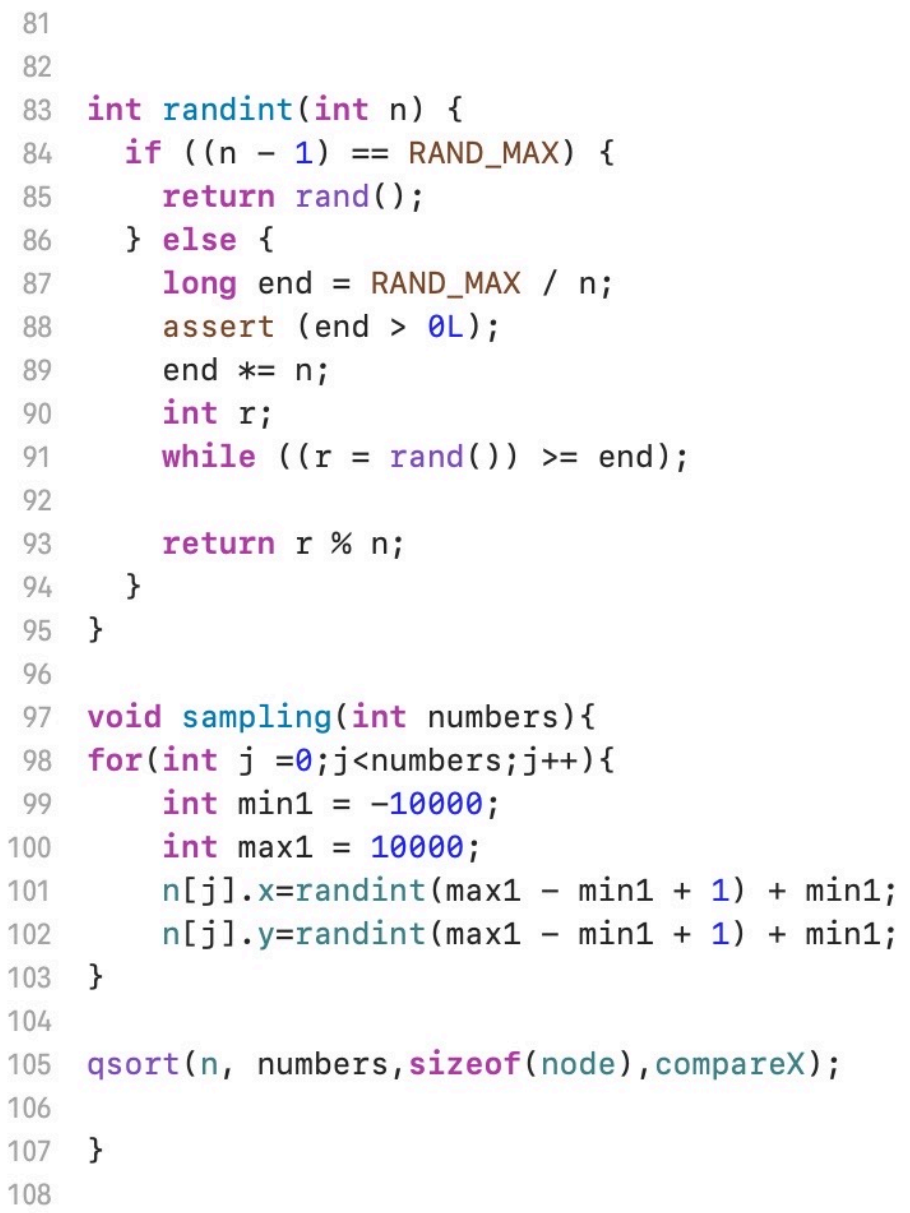
下面是我的程式碼和執行結果:



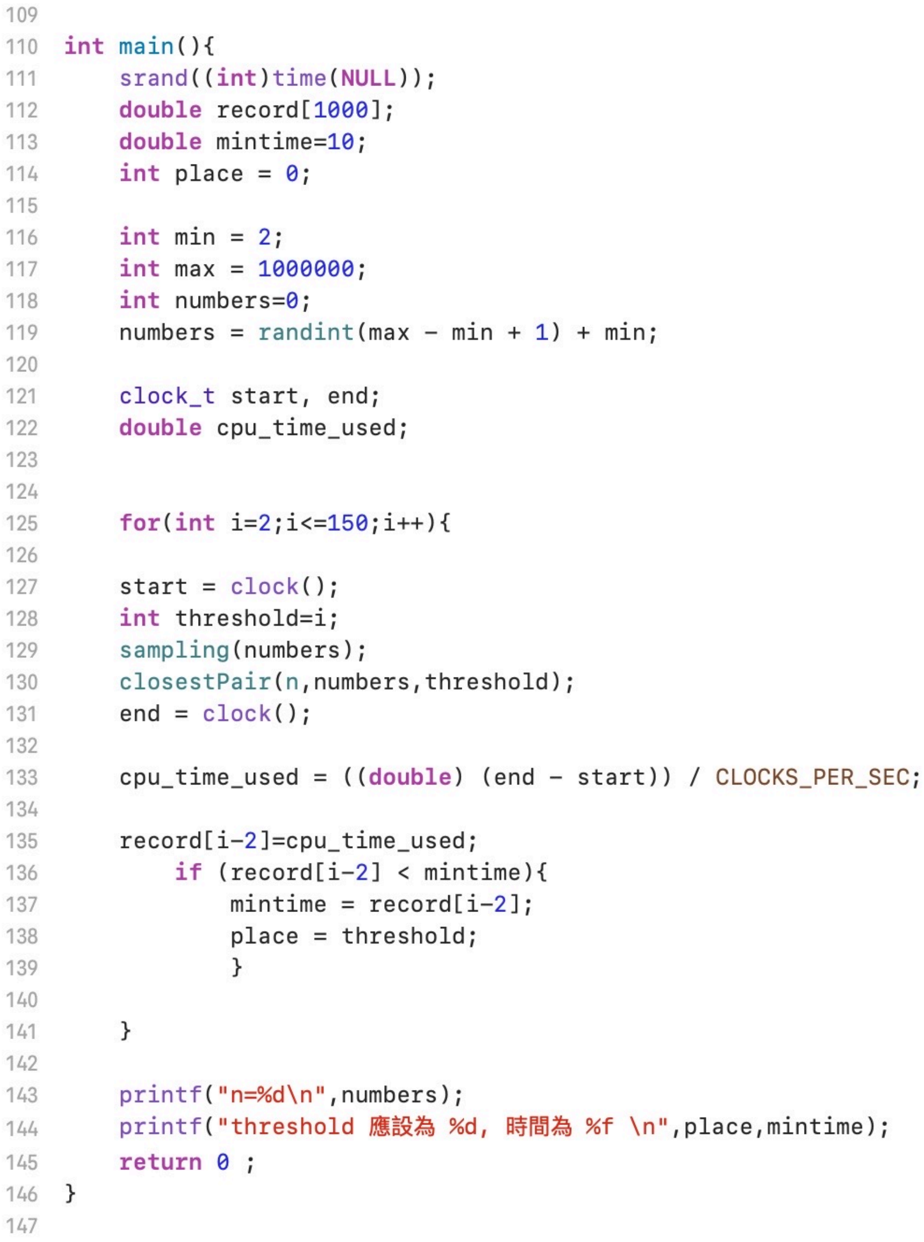
```c
//
//  main.c
//  演算法-程式作業1
//
//  Created by 黃瑋宸 on 2022/3/12.
#include <stdio.h>
#include <float.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <assert.h>
#define N 1000000
struct node
{
    int x, y;
}node;

struct node n[N];
struct node temp[N];

int compareX(const void* a, const void* b)
{
    struct node *ac=(struct node*)a;
    struct node *bd=(struct node*)b;
    return ac->x - bd->x;
}

int compareY(const void* a, const void* b)
{
    struct node *ac = (struct node*)a;
    struct node *bd = (struct node*)b;
    return ac->y - bd->y;
}

double minDistance(double left , double right){
    return (left < right )? left: right;
}


double distance(struct node n1 ,struct node n2)
{
    return sqrt( (n1.x - n2.x)*(n1.x - n2.x) +
                 (n1.y - n2.y)*(n1.y - n2.y)
               );
}
```



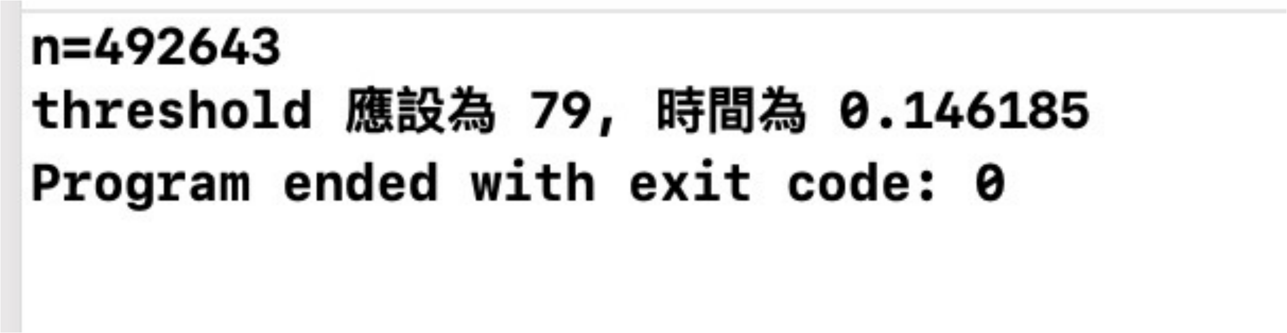
```
double closestPair(struct node n[],int size,int threshold){
    double d;
    if(size<=threshold){
        double min = FLT_MAX;
        for(int i=0;i<size;i++){
            for(int j =i+1;j<size;j++){
                if(distance(n[i],n[j])<min)
                    min=distance(n[i],n[j]);
            }
        }
        return min;
    }

    int mid =size/2;
    int midPoint = n[mid].x;
    d = minDistance(closestPair(n,mid,threshold), closestPair(n+mid,size-mid,thresh

    int counter=0;
    for(int i=0;i < size ; i++){
        if(abs(n[i].x - midPoint) < d){
            temp[counter] = n[i];
            counter++;
        }
    }

    qsort(n,counter,sizeof(node),compareY);

    for(int i=0;i < counter;i++)
        for(int j=i+1 ; j < counter && (n[j].y - n[i].y) < d ;++j)
            if( distance(temp[i] , temp[j]) < d)
                d = distance(temp[i] , temp[j]);
    return d;
}
```



```c


int randint(int n) {
  if ((n - 1) == RAND_MAX) {
    return rand();
  } else {
    long end = RAND_MAX / n;
    assert (end > 0L);
    end *= n;
    int r;
    while ((r = rand()) >= end);

    return r % n;
  }
}

void sampling(int numbers){
for(int j =0;j<numbers;j++){
    int min1 = -10000;
    int max1 = 10000;
    n[j].x=randint(max1 - min1 + 1) + min1;
    n[j].y=randint(max1 - min1 + 1) + min1;
}

qsort(n, numbers,sizeof(node),compareX);

}

```



```
int main(){
    srand((int)time(NULL));
    double record[1000];
    double mintime=10;
    int place = 0;

    int min = 2;
    int max = 1000000;
    int numbers=0;
    numbers = randint(max - min + 1) + min;

    clock_t start, end;
    double cpu_time_used;


    for(int i=2;i<=150;i++){

    start = clock();
    int threshold=i;
    sampling(numbers);
    closestPair(n,numbers,threshold);
    end = clock();

    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;

    record[i-2]=cpu_time_used;
        if (record[i-2] < mintime){
            mintime = record[i-2];
            place = threshold;
            }

    }

    printf("n=%d\n",numbers);
    printf("threshold 應設為 %d, 時間為 %f \n",place,mintime);
    return 0 ;
}
```



```
n=492643
threshold 應設為 79, 時間為 0.146185
Program ended with exit code: 0
```

以上就是我設置threshold時的考量。