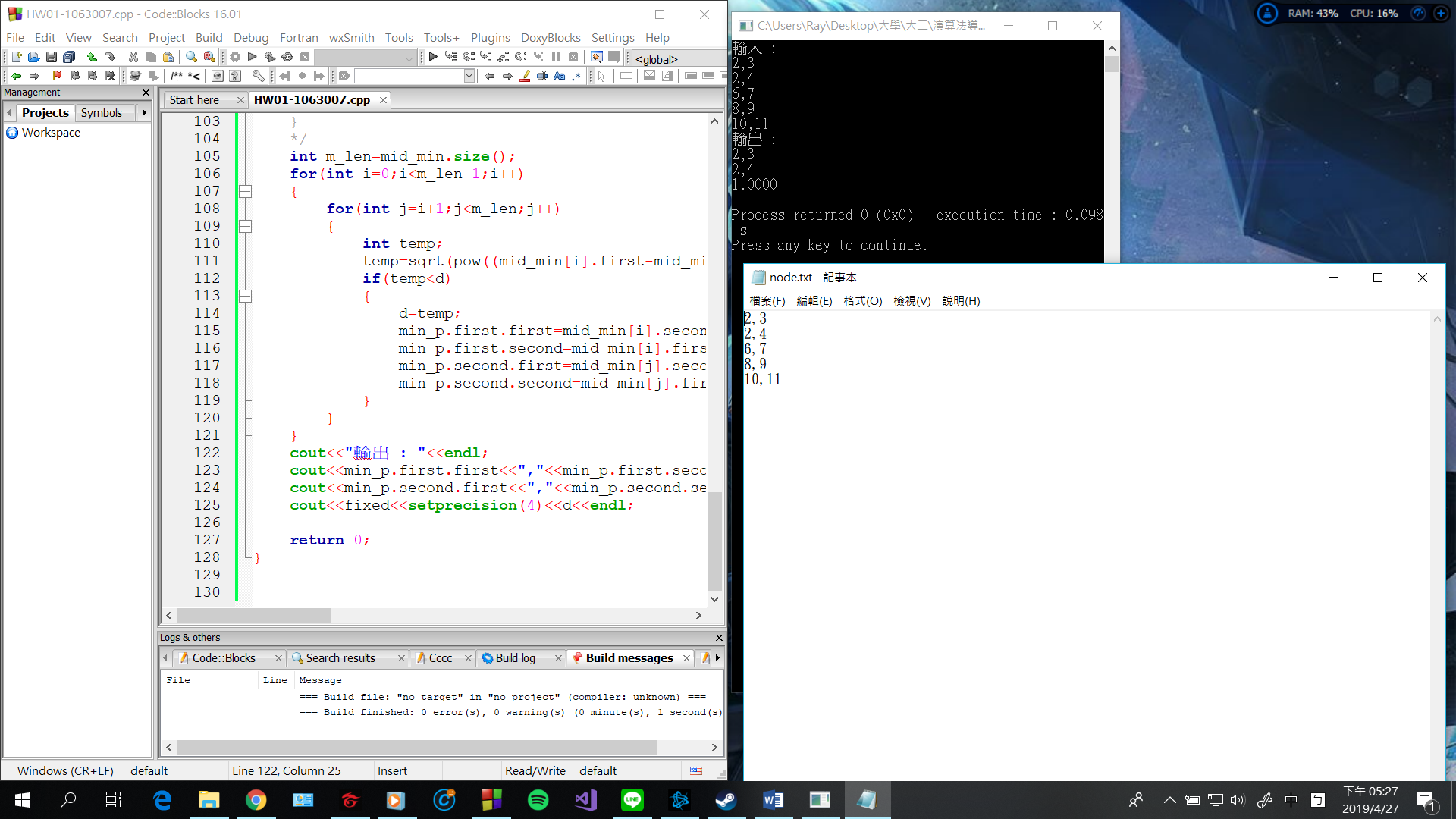
程式說明 : divide and conquer的方法，完成一個closest pair problem程式，先將點依x座標分隔成兩邊，分別找最小距離，再找分隔線左右最所找到的最小距離內之點，進行找最小距離，三次尋找中得最小距離。  
程式執行畫面 : 

原始程式碼 :

#include<bits/stdc++.h>

#include<fstream>

#include<sstream>

using namespace std;

void left()

{

};

int main()

{

fstream fin;

string a;

string aa;

vector<pair<int,int> > point;

int i=0;

fin.open("node.txt");

while(getline(fin,a))

{

aa+=a;

char b;

stringstream ss;

pair<int,int> temp;

ss<<a;

ss>>temp.first>>b>>temp.second;

point.push\_back(temp);

}

cout<<"輸入 : "<<endl;

for(int i=0;i<point.size();i++)

{

cout<<point[i].first<<","<<point[i].second<<endl;

}

///依x排序

sort(point.begin(),point.end());

/\*

for(int i=0;i<point.size();i++)

{

cout<<point[i].first<<"-"<<point[i].second<<endl;

}

\*/

///找切割線

int len=point.size();

int le=len/2;

int ri=le+1;

///處理左邊

pair<pair<int,int>,pair<int,int> > min\_p;

float d=100000000;

for(int i=0;i<le-1;i++)

{

for(int j=i+1;j<le;j++)

{

int temp;

temp=sqrt(pow((point[i].first-point[j].first),2) + pow((point[i].second-point[j].second),2));

if(temp<d)

{

d=temp;

min\_p.first.first=point[i].first;

min\_p.first.second=point[i].second;

min\_p.second.first=point[j].first;

min\_p.second.second=point[j].second;

}

}

}

//cout<<"left min : "<<min\_p.first.first<<"-"<<min\_p.first.second<<"\_"<<min\_p.second.first<<"-"<<min\_p.second.second<<fixed<<setprecision(4)<<" d : "<<d<<endl;

///處理右邊

for(int i=ri-1;i<len-1;i++)

{

for(int j=i+1;j<len;j++)

{

int temp;

temp=sqrt(pow((point[i].first-point[j].first),2) + pow((point[i].second-point[j].second),2));

if(temp<d)

{

d=temp;

min\_p.first.first=point[i].first;

min\_p.first.second=point[i].second;

min\_p.second.first=point[j].first;

min\_p.second.second=point[j].second;

}

}

}

//cout<<"right min : "<<min\_p.first.first<<"-"<<min\_p.first.second<<"\_"<<min\_p.second.first<<"-"<<min\_p.second.second<<fixed<<setprecision(4)<<" d : "<<d<<endl;

///處理中間

vector<pair<int,int> > mid\_min;///x,y相反存

int mid\_le=point[le-1].first-d,mid\_ri=point[le-1].first+d;

for(int i=0;i<len;i++)

{

if(point[i].first>=mid\_le&&point[i].first<=mid\_ri)

{

pair<int,int> temp;

temp.second=point[i].first;

temp.first=point[i].second;

mid\_min.push\_back(temp);

}

}

sort(mid\_min.begin(),mid\_min.end());

/\*

for(int i=0;i<mid\_min.size();i++)

{

cout<<mid\_min[i].first<<"-"<<mid\_min[i].second<<endl;

}

\*/

int m\_len=mid\_min.size();

for(int i=0;i<m\_len-1;i++)

{

for(int j=i+1;j<m\_len;j++)

{

int temp;

temp=sqrt(pow((mid\_min[i].first-mid\_min[j].first),2) + pow((mid\_min[i].second-mid\_min[j].second),2));

if(temp<d)

{

d=temp;

min\_p.first.first=mid\_min[i].second;

min\_p.first.second=mid\_min[i].first;

min\_p.second.first=mid\_min[j].second;

min\_p.second.second=mid\_min[j].first;

}

}

}

cout<<"輸出 : "<<endl;

cout<<min\_p.first.first<<","<<min\_p.first.second<<endl;

cout<<min\_p.second.first<<","<<min\_p.second.second<<endl;

cout<<fixed<<setprecision(4)<<d<<endl;

return 0;

}

簡要心得 : 運用課堂學到的divide and conquer來做，因為是第一次寫，所以也花了不少的時間，倒是沒遇上甚麼大問題，由於輸入資料不多，我也不知道到底有沒有比窮舉法還快。