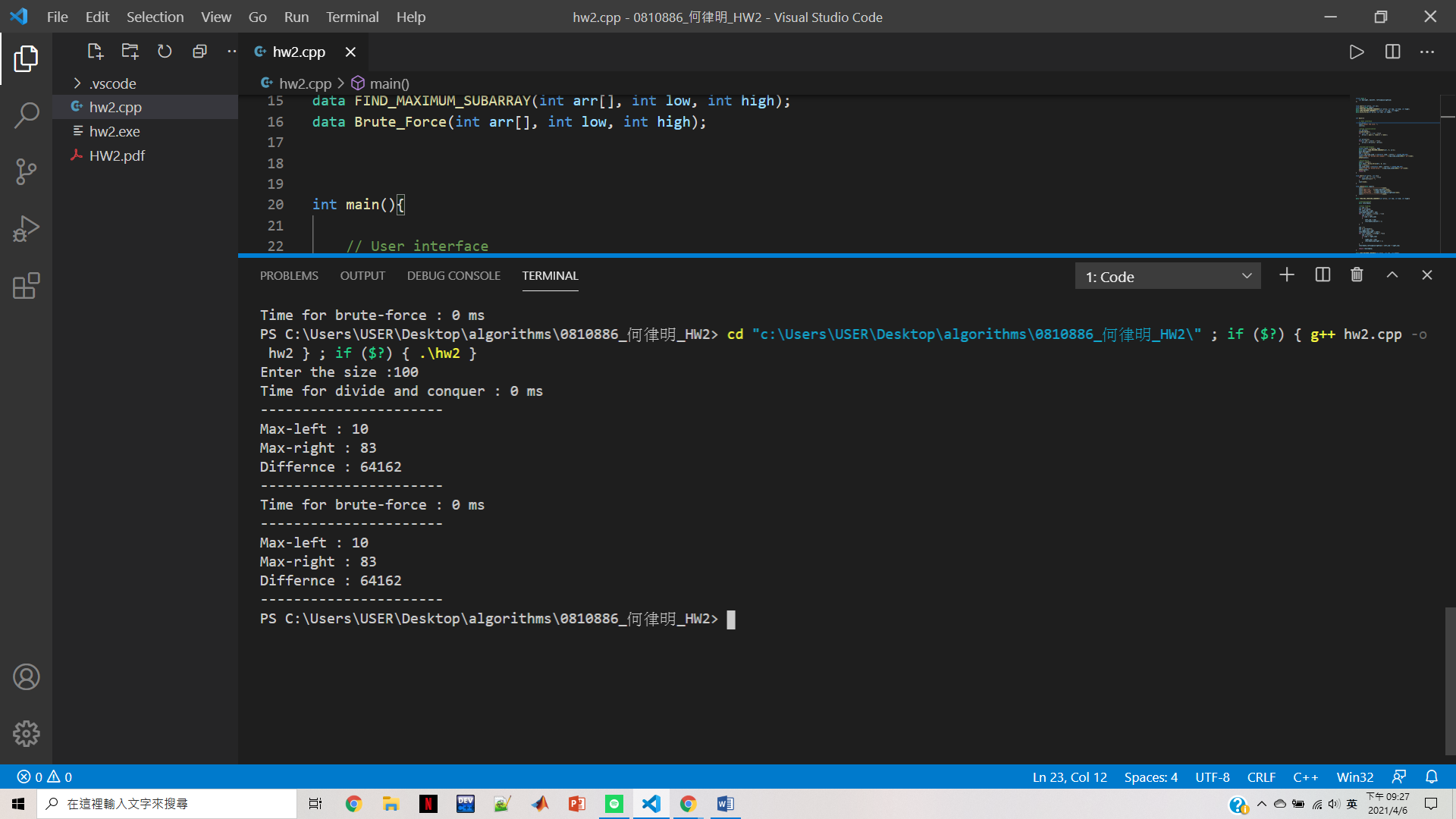
Time for size 100 :

Divide-and-Conquer : 0ms

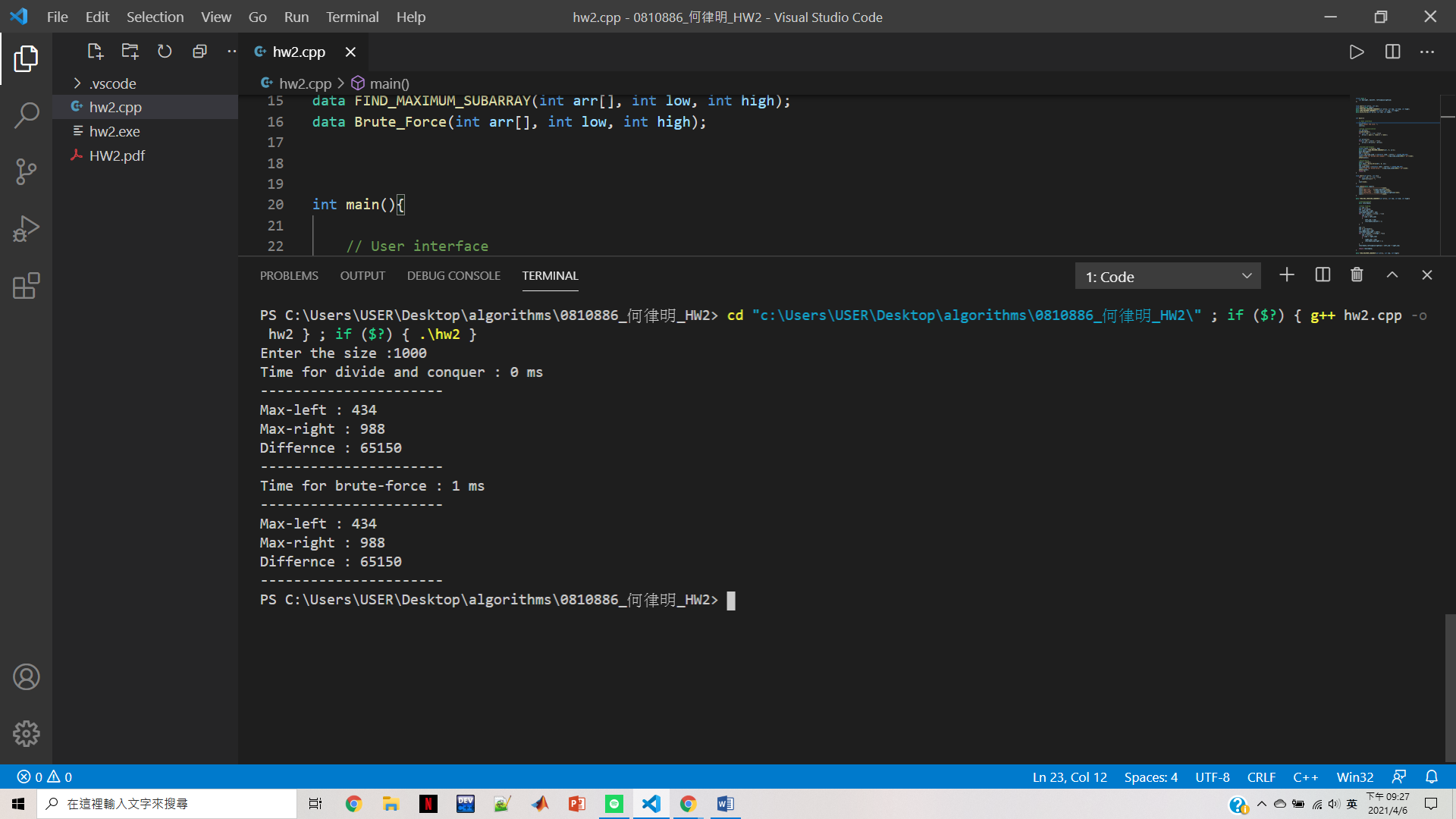
Brute-Force : 0ms



Time for size 1000 :

Divide-and-Conquer : 0ms

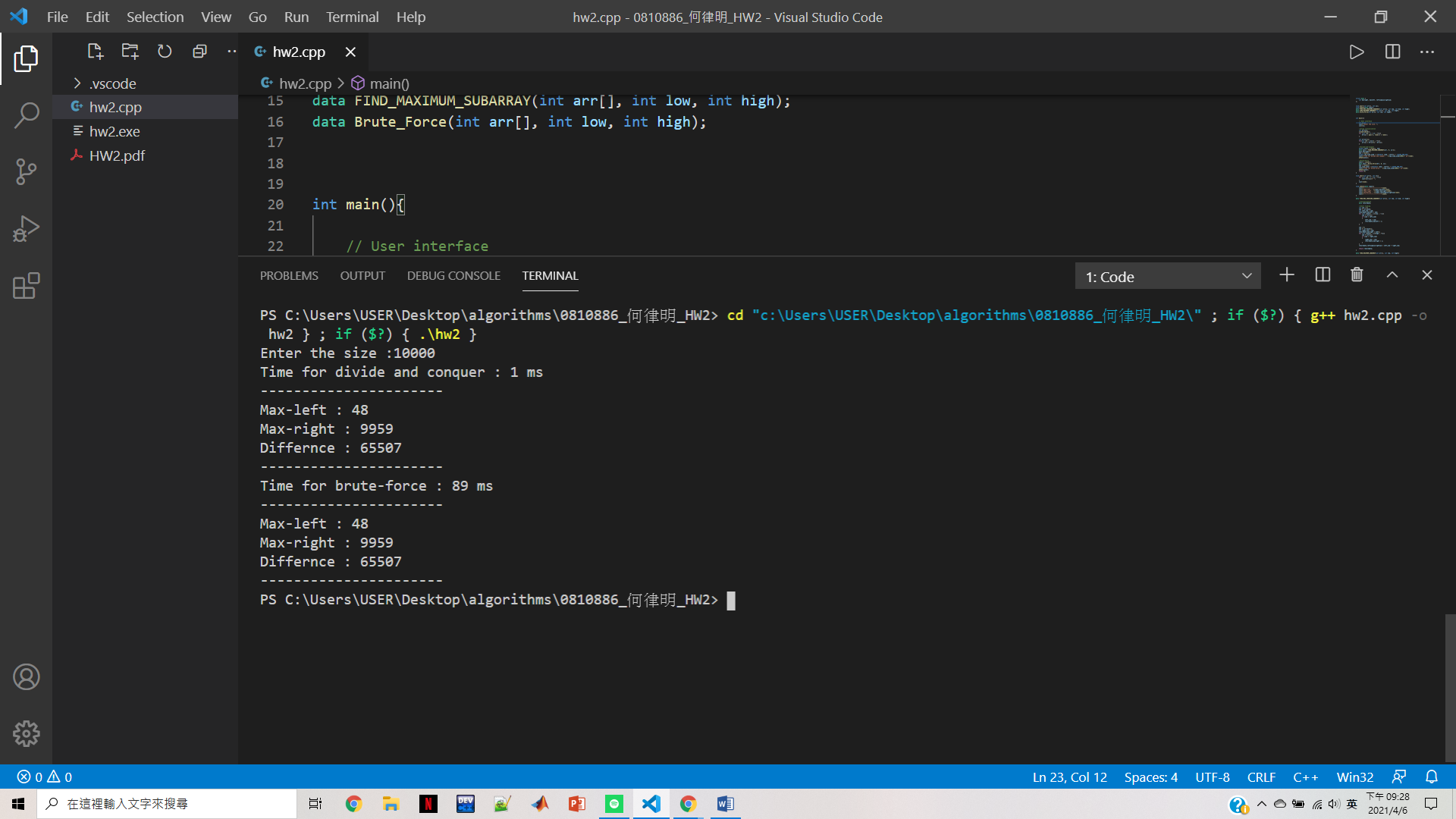
Brute-Force : 1ms



Time for size 10000 :

Divide-and-Conquer : 1ms

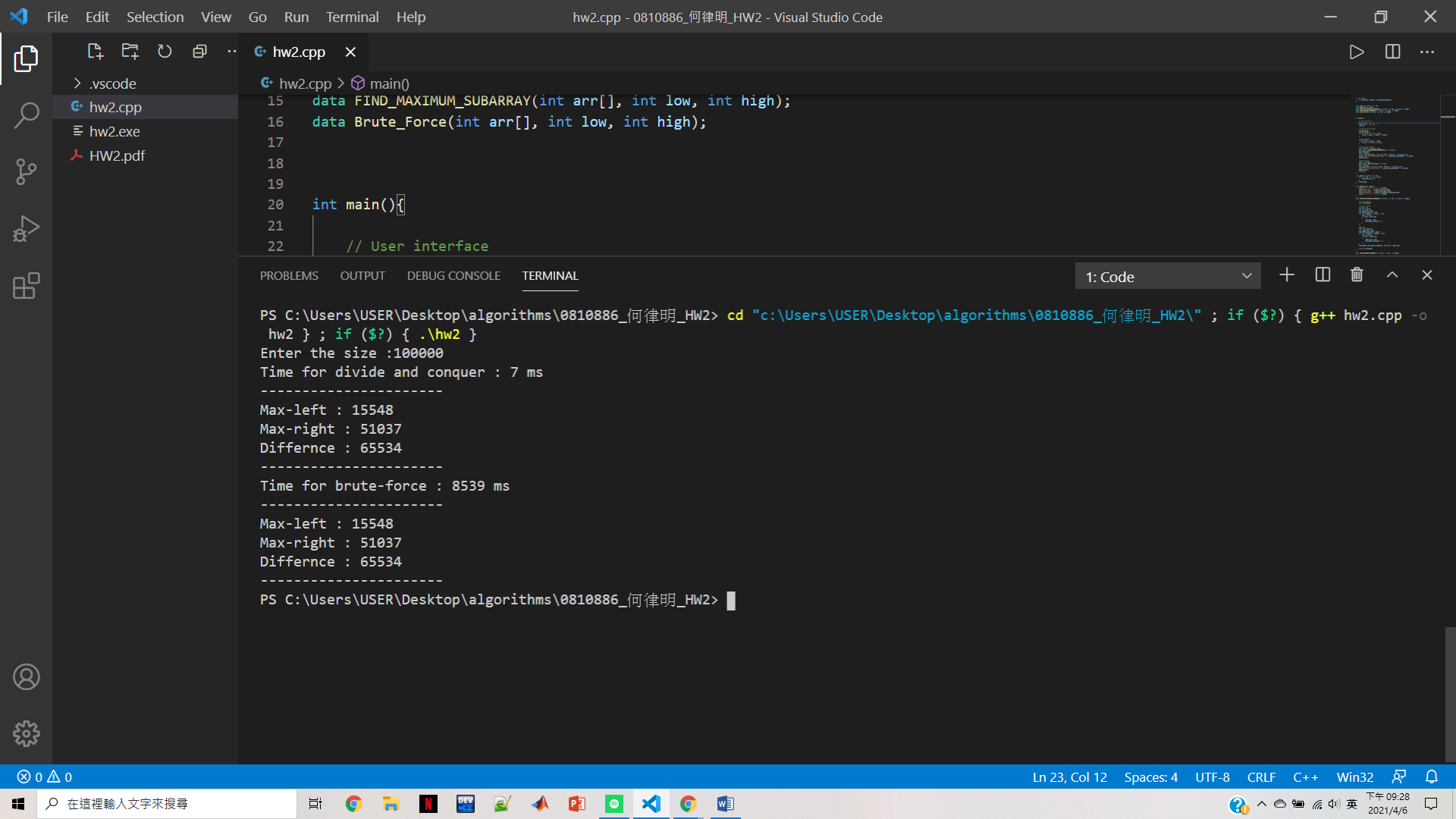
Brute-Force : 89ms



Time for size 100000 :

Divide-and-Conquer : 7ms

Brute-Force : 8539ms



Functions Introduction:

struct data {

    int maxright, maxleft, leftsumplusrightsum;

};

這個struct是我拿來回傳答案的，叫做data。

void ptdata(data input){

    cout<<"----------------------"<<endl;

    cout<<"Max-left : "<<input.maxleft<<endl;

    cout<<"Max-right : "<<input.maxright<<endl;

    cout<<"Difference : "<<input.leftsumplusrightsum<<endl;

    cout<<"----------------------"<<endl;

}

此function傳入一個data type的input，就會印出Max-left & Max-right & Difference。

data Brute\_Force(int arr[], int low, int high){

    data tmp;

    int diff = INT32\_MIN;

    for(int i=low ; i<high ; ++i){

        for(int j=i+1 ; j<high ; ++j){

            if(arr[j] - arr[i] > diff){

                diff = arr[j] - arr[i];

                tmp.maxleft = i;

                tmp.maxright = j;

                tmp.leftsumplusrightsum = diff;

            }

        }

    }

    return tmp;

}

暴力解法，雙重for迴圈，任選兩個找出最大差值的組合。

data FIND\_MAX\_CROSSING\_SUBARRAY(int arr[], int low, int mid, int high){

    //Initialization

    data returndata;

    //Start finding

    int sum = 0;

    sum += arr[mid];

    int left\_sum = sum;

    returndata.maxleft = mid;

    for (int i=mid-1 ; i>=low ; --i){

        sum += arr[i];

        if (sum > left\_sum)

        {

            left\_sum = sum;

            returndata.maxleft = i;

        }

    }

    sum = 0;

    sum += arr[mid+1];

    int right\_sum = sum;

    returndata.maxright = mid+1;

    for (int j=mid+2 ; j<=high ; ++j){

        sum += arr[j];

        if (sum > right\_sum)

        {

            right\_sum = sum;

            returndata.maxright = j;

        }

    }

    returndata.leftsumplusrightsum = left\_sum + right\_sum;

    return returndata;

}

根據課本pseudo code打出來的，這適用於max subarray橫跨中間那個值所調用的function。

data FIND\_MAXIMUM\_SUBARRAY(int arr[], int low, int high){

    if (low == high){

        data tmp;

        tmp.maxleft = low;

        tmp.maxright = high;

        tmp.leftsumplusrightsum = arr[low];

        return tmp;

    }

    else{

        int mid = floor((low + high)/2);

        data left, right, cross;

        left =  FIND\_MAXIMUM\_SUBARRAY(arr, low, mid);

        right = FIND\_MAXIMUM\_SUBARRAY(arr, mid+1, high);

        cross = FIND\_MAX\_CROSSING\_SUBARRAY(arr, low, mid, high);

        if(left.leftsumplusrightsum >= right.leftsumplusrightsum && left.leftsumplusrightsum >= cross.leftsumplusrightsum){

            return left;

        }

        else if(right.leftsumplusrightsum >= left.leftsumplusrightsum && right.leftsumplusrightsum >= cross.leftsumplusrightsum){

            return right;

        }

        else{

            return cross;

        }

    }

}

這個也是根據課本pseudo code打出來的，分成三個部分，找max subarray全部在左半邊的，以及max subarray全部在右半邊的，還有max subarray橫跨中間的(需調用上一頁的function)，最終返回一個type為data(包含三個members)的值。

int main(){

    // User interface

    int sz;

    cout<<"Enter the size :";

    cin>>sz;

    //Array initialization

    int Arr[sz];

    srand(time(0));

    for (int i=0 ; i < sz ; ++i){

        Arr[i] = pow(-1, rand()) \* rand();

    }

    int arr[sz-1];

    for(int i=0 ; i<sz-1 ; ++i){

        arr[i] = Arr[i+1] - Arr[i];

    }

    //Divide-and-Couquer

    double start = clock(), end;

    data ans1 = FIND\_MAXIMUM\_SUBARRAY(arr, 0, sz-2);

    ans1.maxright++;

    end = clock();

    double cpu\_time\_used = ((double) (end - start)) / CLOCKS\_PER\_SEC;

    cout<<"Time for divide and conquer : "<<cpu\_time\_used\*1000<<" ms"<<endl;

    ptdata(ans1);

    //Brute-Force

    start = clock();

    data ans2 = Brute\_Force(Arr, 0, sz);

    end = clock();

    cpu\_time\_used = ((double) (end - start)) / CLOCKS\_PER\_SEC;

    cout<<"Time for brute-force : "<<cpu\_time\_used\*1000<<" ms"<<endl;

    ptdata(ans2);

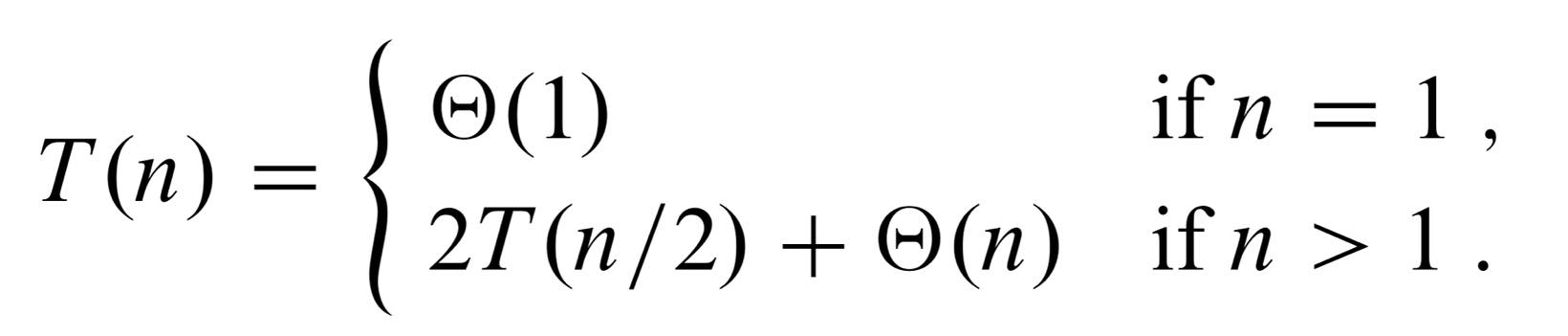
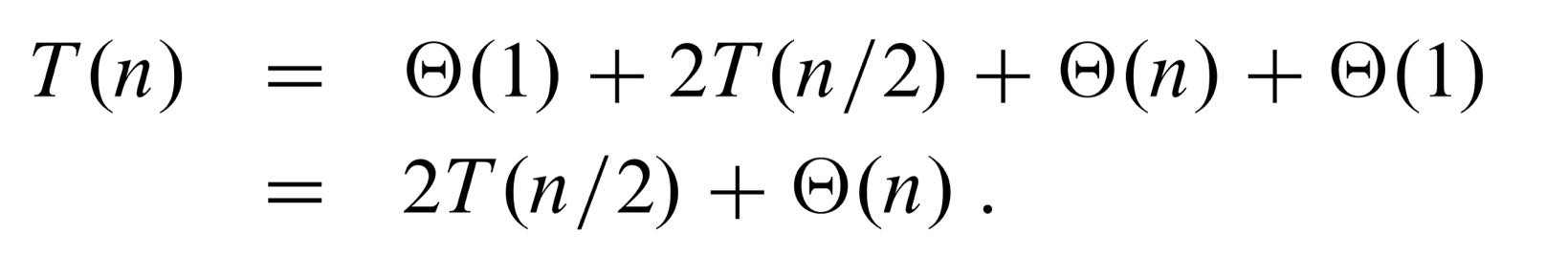
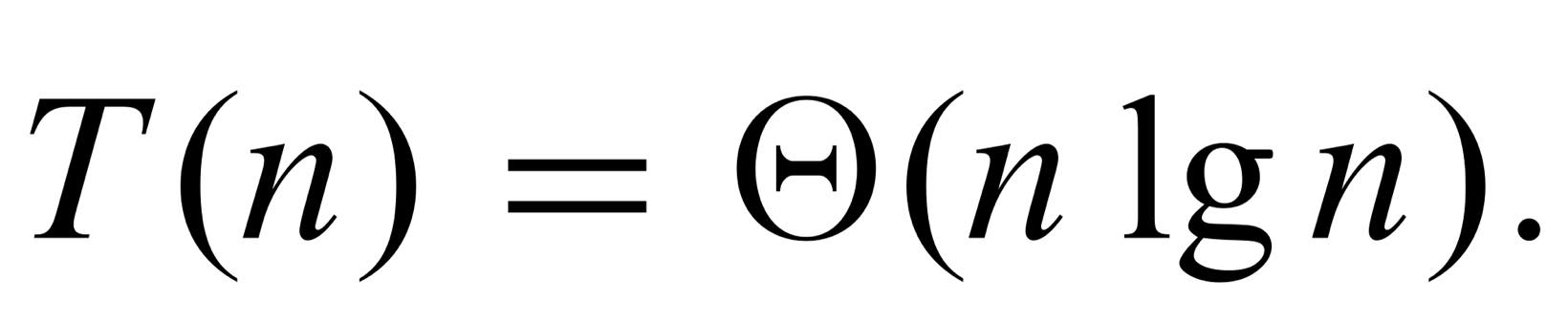
    return 0;

}

以上為主程式，會先讓user輸入input size，並隨機在整數最大值以及整數最小值中隨機產生出size為指定大小的integer array，並輸出divide and conquer & brute force各自所花的時間以及答案。

Observation:

我的程式碼都是根據課本的pseudo code去打的。所以running time 應該課本描述得差不多，以下為Divide-and-Conquer的running time:

最終經過化簡以及驗證，running time的Big Theta為nlogn。

根據我寫的程式，divide and conquer在input size小於等於1000的時候，因為執行時間都低於1ms，所以都顯示0秒，當n = 10000時，我執行出時間大部分為0ms偶爾出現為1ms，以我測量的次數去做平均大概是5:1，也因此平均時間我抓0.2ms，當n = 100000時，執行時間為幾乎都差不多7ms，我有多做n = 200000如下:

Enter the size :200000

Time for divide and conquer : 17 ms

----------------------

Max-left : 5475

Max-right : 103979

Difference : 65534

----------------------

Time for brute-force : 34086 ms

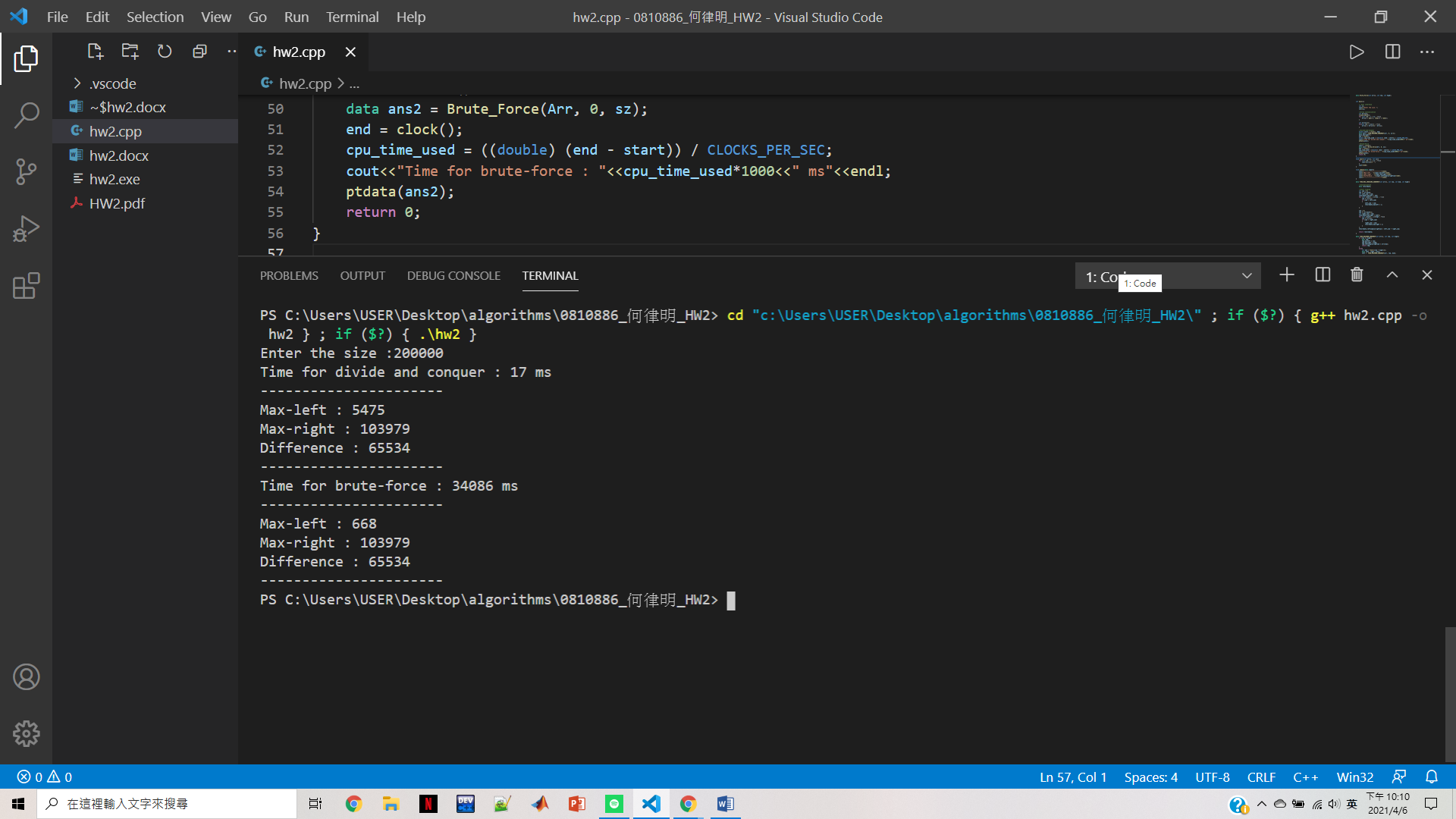
----------------------

Max-left : 668

Max-right : 103979

Difference : 65534

----------------------



當n = 10000，n\*logn = 40000，平均為0.2ms-----------------------(1)

當n = 100000，n\*logn = 500000，平均為7ms----------------------(2)

當n = 200000，n\*logn = 1060205，平均為17ms-------------------(3)

由此可見n\*logn跟平均時間成正比關係，所以用divide and conquer方法去找maximum subarray所花費的時間的確是Big Theta(n\*logn)。

同理Brute-Force:

因為裡面有雙重迴圈，而且第一層的i是從0 ~ n，第二層的j是從i ~ n，很明顯根據經驗部難看出為n\*n。

當n = 1000，n\*n = 1e6，平均為1ms-----------------------(1)

當n = 10000，n\*n = 1e8，平均為89ms-----------------------(1)

當n = 100000，n\*n = 1e10，平均為8539ms-----------------------(1)

當n = 200000，n\*n = 4e10，平均為34086ms-----------------------(1)

也很明顯的n\*n跟平均時間也差不多在正比關係的那條直線上。

Conclusion:

Divide and conquer用來找maximum subarray用的是recursive的概念，跟上ˋ作業一模一樣為n\*logn，而brute force則採用雙重迴圈的方法暴力去尋找答案，故Big Theta為n平方。