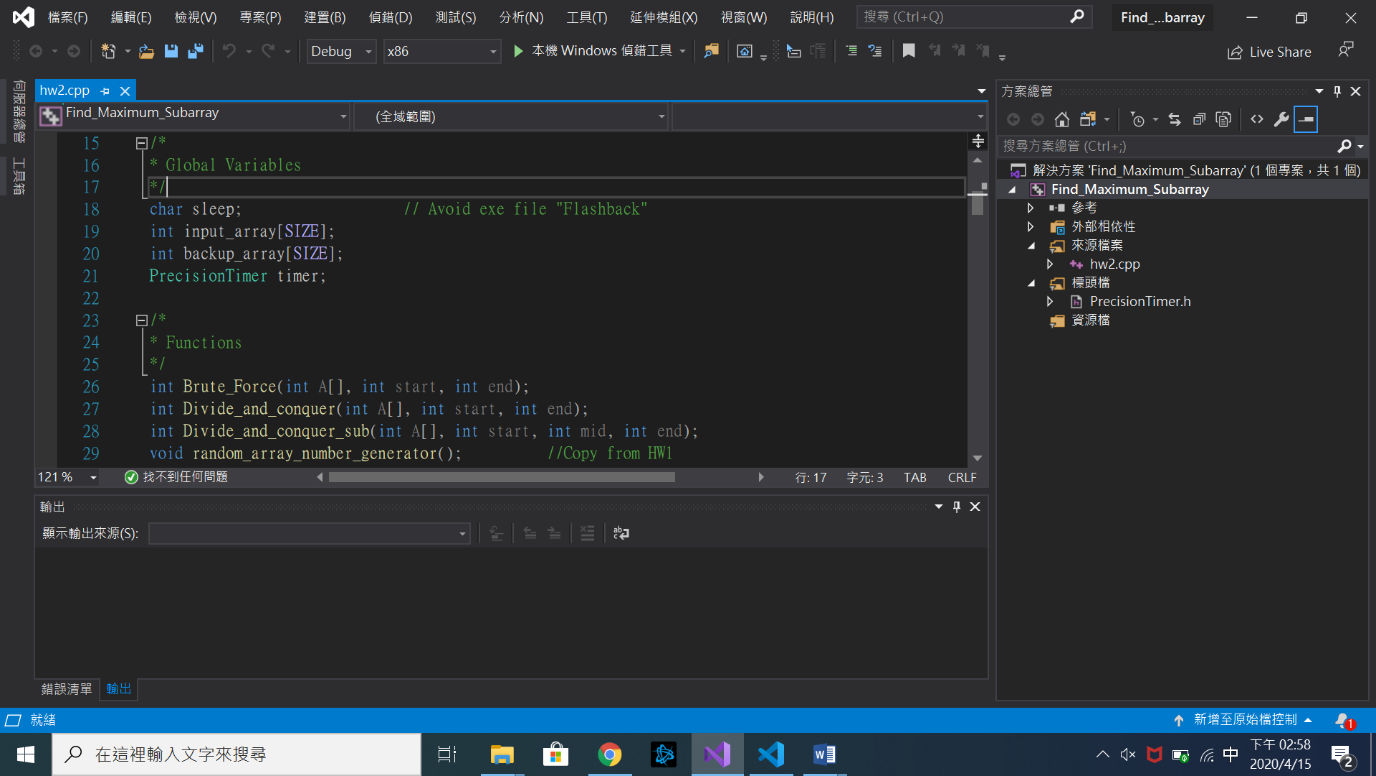
演算法\_HW2 0617052 林明佑

**Description of implementation:**

4 small functions + 1 main function + 1 header file



PrecisionTimer.h : Much more precision timer. Return value is **sec.**

Brute\_Force : Use Brute-Force to do maximum subarray problem ,**O(n^2)** algorithm.

Divide\_and\_conquer : Use Divide\_and\_conquer to do maximum subarray problem, it is a

**O(nlogn)** algorithm.

Divide\_and\_conquer\_sub : Use to calculate and return the maximum of the sums of sequences that cross the midpoint of a given array.(linear time algorithm)

Random\_array\_number\_generator : Use to generate random numbers. Copy from HW1.

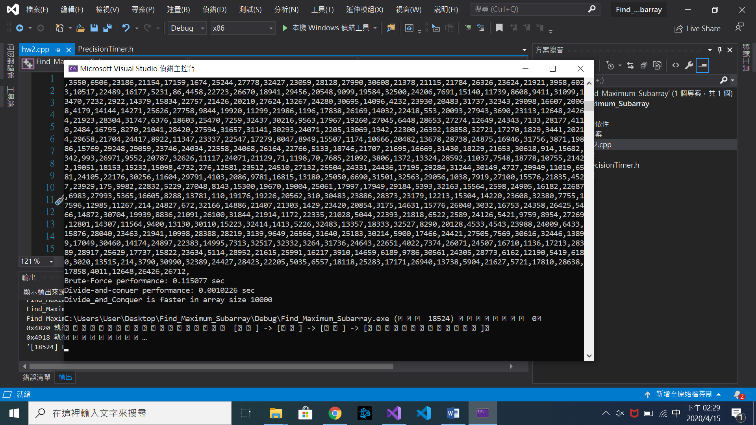
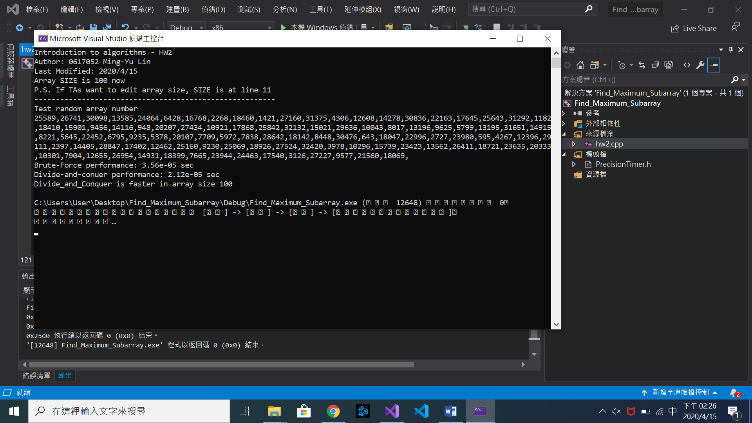
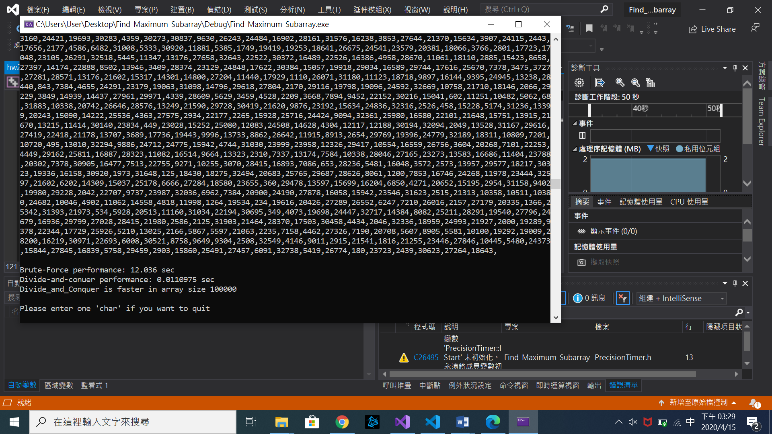
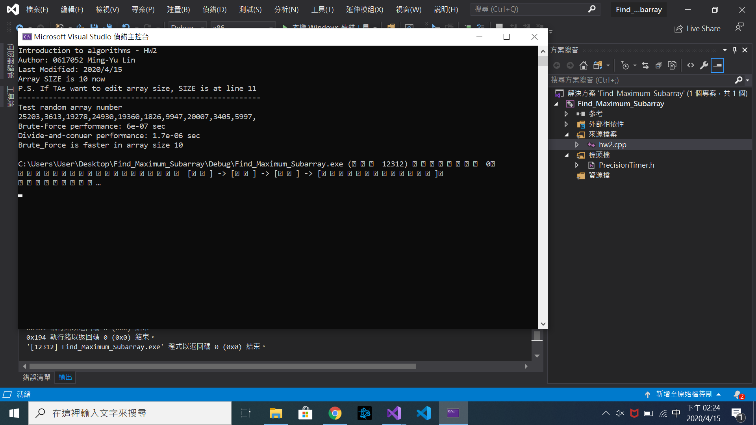
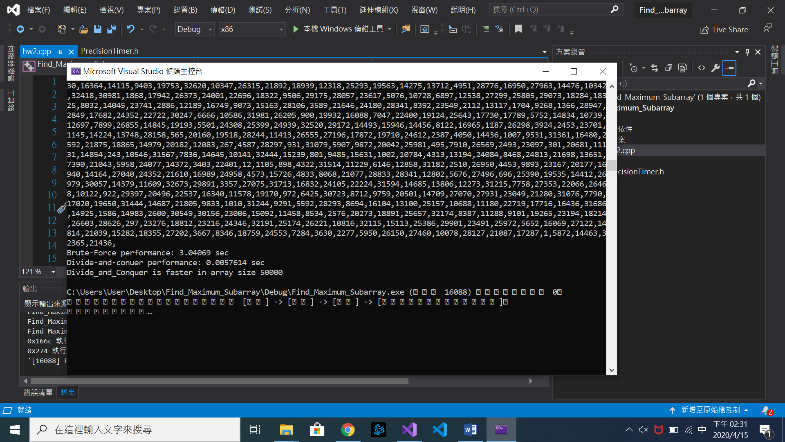
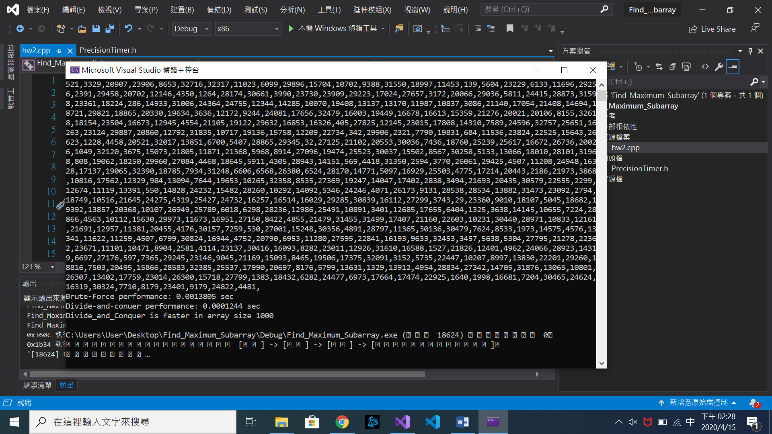
**Observation of different input\_array[] sizes:**

|  |  |  |
| --- | --- | --- |
| **SIZE** | **Brute\_Force** | **Divide and conquer** |
| **10** | **6e-07 sec** | **1.7e-06 sec** |
| **100** | **3.56e-05 sec** | **2.12e-05 sec** |
| **1000** | **0.0013825 sec** | **0.0001244 sec** |
| **10000** | **0.115077 sec** | **0.0010226 sec** |
| **50000** | **3.04869 sec** | **0.0057614 sec** |
| **100000** | **11.9797 sec** | **0.0107442 sec** |

Brute\_Force is slower than Divide and conquer algorithm when the array size is larger than 100.

After array size is larger than 100, Divide and conquer is always faster than Brute Force.

Screenshots of each array size are in the next page.(6 pictures).

**Summary:**

**From class notes:**

Brute Force is **O(n^2)** algorithm in maximum subarray problem.

Divide and conquer is **O(nlogn)** algorithm in maximum subarray problem.

**From the definition of O:**

We can find n for **T\_brute\_force(n) >= T\_Divide\_and\_conquer(n)** is always true because (n^2) has the faster growth rate than (nlogn).

According to my observation of different array sizes, when n is larger than 100, divide and conquer always spend less time than brute force to solve the problem.