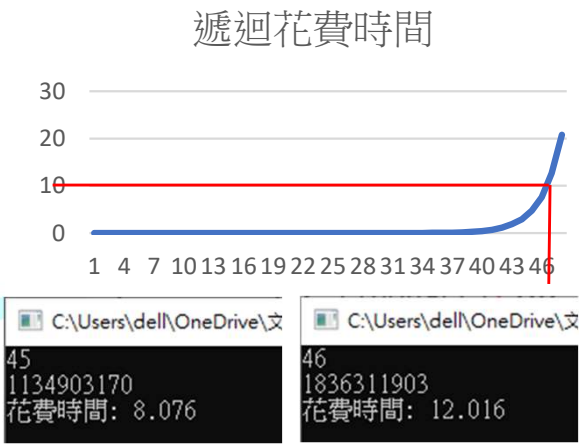
第二次程式作業　資工2A 109502510 張芷烜

遞迴程式碼:

```cpp
#include<bits/stdc++.h>

using namespace std;
int fibonacci(int n){
    if(n==0) return 0;
    if(n==1) return 1;

    return fibonacci(n-1)+fibonacci(n-2);
}
int main(){
    double START,END;
    ofstream ofs;
    ofs.open("recursive.txt");
    for(int i=0;i<48;i++){
        START = clock();
        cout<<fibonacci(i)<<endl;
        END = clock();
        ofs <<(END - START) / CLOCKS_PER_SEC  << "\n";
    }
    return 0;
}
```



時間過40後直線上升，測試出n=46時，執行時間超過10sec(12.016sec)

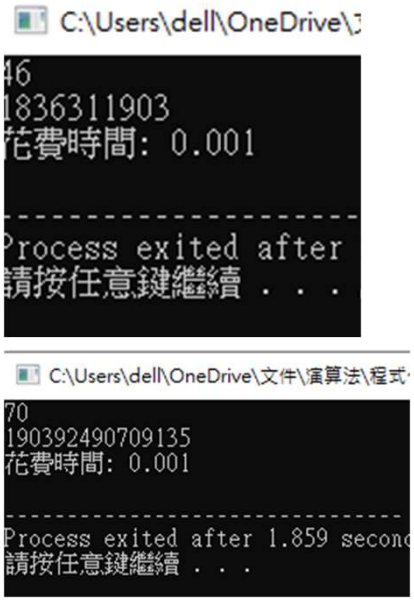
DP程式碼:

```cpp
#include<bits/stdc++.h>

using namespace std;

int main(){
    int n;
    double START,END;
    cin>>n;
    long long int *fibonacci=new long long int[n];
    fibonacci[0]=0;
    fibonacci[1]=1;
    START = clock();
    for(int i=2;i<=n;i++)
        fibonacci[i]=fibonacci[i-1]+fibonacci[i-2];

    cout<<fibonacci[n]<<endl;
    END = clock();
    cout<<"花費時間: "<<(END - START) / CLOCKS_PER_SEC<<endl;
}
```



由實驗數據可知
用DP，儲存已經算過的數字後
尋找fibonacci[n]的時間將大幅下降
Bottom up 計算數列，不須浪費時間算已經算過的數值
前面的數字可以直接取用array中已經計算好的數值

結論:
若用遞迴實作在計算F[n]時須先計算 F[n-1],F[n-2]
　　計算F[n-1]須先計算 F[n-2],F[n-3]
可以發現會重複計算F[n-2]，重複計算導致執行時間過長，浪費時間
用DP的方法解決重複計算，並且使用bottom up 算好前面的值儲存，
便可節省時間直接取用值，故可以看到DP執行結果快速許多