Final Report

104502518 劉冠聲

一般的費氏函數：

```
def fibonacci_n(n):
    if n < 2:
        return n
    return fibonacci_n(n-1) + fibonacci_n(n-2)
```

費式數列的計算為數列中前兩個數字相加即為目前數字的值，直觀的寫法為使用遞迴函數，讓自己函數的前兩個返回值相加，例如fibonacci_n(50)就會呼叫fibonacci_n(49)和fibonacci_n(48)，但這樣會導致同樣為費氏數列第n個的值不斷被重新計算，由於會呼叫前2個數字的費氏函數，每當n增加1，計算時間便成長將近一倍。

實驗結果：fibonacci_n(39)計算時間為17.7秒

有decorators的費氏函數：

```
def cache_cut(func):
    cache_size = 50
    cache = [0] * cache_size
    def wrapper(n):
        if cache_size > n and n > 1:
            if cache[n] != 0:
                return cache[n]
            cache[n] = func(n)
            return cache[n]
        return func(n)
    return wrapper

@cache_cut
def fibonacci_n(n):
    if n < 2:
        return n
    return fibonacci_n(n-1) + fibonacci_n(n-2)
```

在原本費氏函數不變的狀況下，透過python decorators的加入，每次呼叫費氏函數時，會先經過函數cache_cut，本project中預設快取的大小為50個，將已經計算過的費氏數列儲存進快取，未來需要時可直接從快取中取得，不需要再遞迴回數列最前面，用這種以空間換時間的方式，從原本遞迴top-down的計算變成button-up的計算，加速計算時間。

實驗結果：fibonacci_n(39)計算時間為0.000007秒

附註：因為快取的緣故，在快取大小內計算時間皆差不多，因為僅單純計算快取中前兩個數字相加。