**Programming Language hw4**

何柏勤(H34031013)

簡湘霖(H24011257)

林文盛(F74037049)

洪正皇(C14031162)

王晉鋒(C14036227)

邱靖吉(F74035047):

**第12組**

實作出concurrency的矩陣乘法

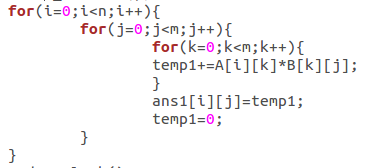
1. **一般的矩陣相乘**

一般的矩陣乘法其演算法如下:

,

則

轉變為程式碼為下：

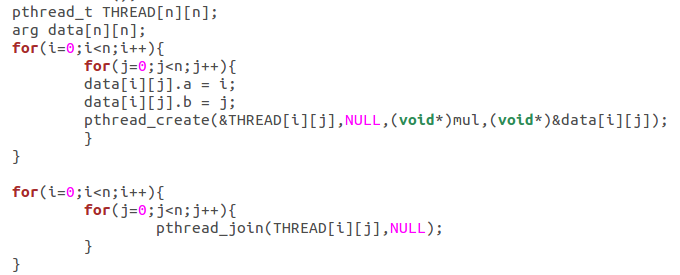


(答案ans1矩陣為及最後結果)

1. **用thread的矩陣相乘**

在迴圈中再生成thread，每一個thread分別去拿取A矩陣和B矩陣中的值，並且作相乘相加，產生出最後的結果。運用multithread是為了做平行化，用pthread\_create先產生thread，並執行參數裡面的指令，而要做平行化的話要加上pthread\_join，這是在說要等前面的thread做完後才可以再繼續做下一個。

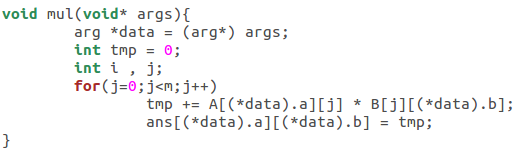
程式碼大致如下：



其中pthread\_create()中有四個參數，第一個是pthread的指標，也就是前面所宣告的pthread變數，第二個變數為thread的屬性，預設為NULL，第三個參數釋放要執行funciton的pointer，第四個參數是放此function所需要的參數。

而pthread\_join()是指等待到目標的thread執行完畢之後才會在執行目前的thread，要放入兩個參數，第一個為要等待的目標thread，第二個為目標thread的回傳值。

此作業是要做矩陣相乘，因此每個thread所執行的function為下，也就是A的行乘以B的列的結果：



運用一般的矩陣相加的演算法所需要的時間複雜度為.

1. **運用Strassen Algorithm的矩陣相乘**

Strassen Algorithm為把矩陣拆成四個部分去做運算：

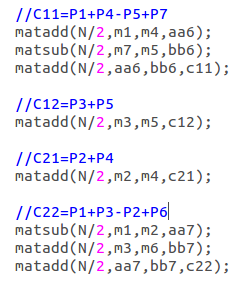
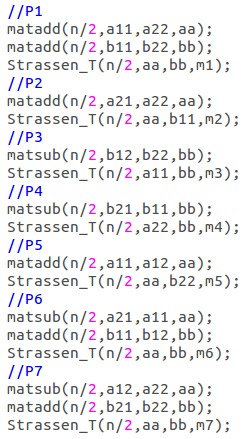
若是一般的乘法，，，，需要做8次乘法和4次加法。而Strassen Algorithm為：

總共有7次乘法，和18次加法，比一般的矩陣乘法少了一次，可以大大的降低時間複雜度，Strassen Algorithm所需要的時間複雜度為

實作的程式碼的解說如下：

matadd為兩數相加，matsub為兩數相減

Strassen(傳pointer)與Strassen\_T(傳reference)為實現上述的Strassen Algorithm，先拆成四個小矩陣再去做相乘相加，如果所要計算的矩陣無法拆成四個相同的小的矩陣的話，我們在這邊補0，把它變成可拆解的矩陣再下去做。



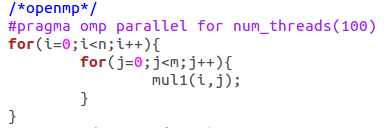
1. **Strassen Algorithm加上thread的矩陣相乘**

在main中，Strassen Algorithm的演算法中需要做7次的乘法，因此在每次的乘法中都生成一個thread，總共生成7個thread分別去做矩陣運算，每個thread中第一個參數為Strassen的函式名稱，其餘參數為要傳入Strassen的參數，但是透過thread傳參數無法與Strassen\_T一樣以reference當參數，似乎要用C語言的方式傳參數，因此在這邊用指標。隨後的join()能等待thread結束才繼續往下執行，避免thread還沒做完就發生不可預期的情況，例如return 0。



1. **Openmp平行運算方法**

OpenMP是一個平行化的library，可以實現concurrency。其程式碼如下：



mul1是我們自己寫的function，加上#pragma omp parallel for num\_threads(100)，即為產生100個thread下去做運算，以達到矩陣乘法的平行化。

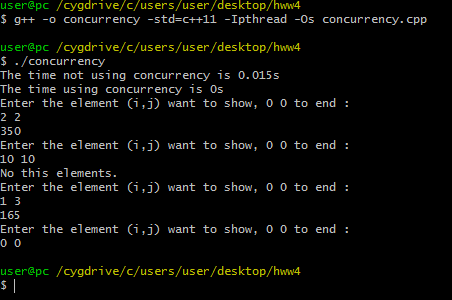
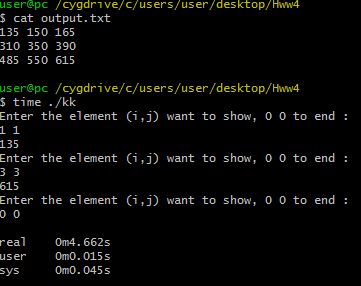
**結果：**

根據上面三種方法所做出的執行時間和結果如下：

運用Thread還需要計算生成Thread的時間，所以不一定會比較快，要看所測得測資的大小而定，若測資小也許一般的矩陣相乘會跑得比較快，而用Strassen Algorithm可以大幅的改善時間複雜度，使執行時間縮短，但是因為所測得測資矩陣大小都很小，所以其實看不太出來有差別。

* **concurrency.cpp檔以C++撰寫，可以跑出Strassen Algorithm的結果以及Strassen Algorithm加上concurrency的結果，compile時請輸入g++ -o concurrency -std=c++11 -Ipthread -Os concurrency.cpp。**
* **concurrency.c檔以C撰寫，可以跑出一般的矩陣相乘結果以及加上concurrency的結果，compile時請輸入gcc -o -pthread -fopenmp concurrency.c。**

concurrency.cpp檔執行結果：



註：這邊矩陣為3\*5太小了，用ctime的clock計時有時會為0s。

concurrency.c檔執行結果：

