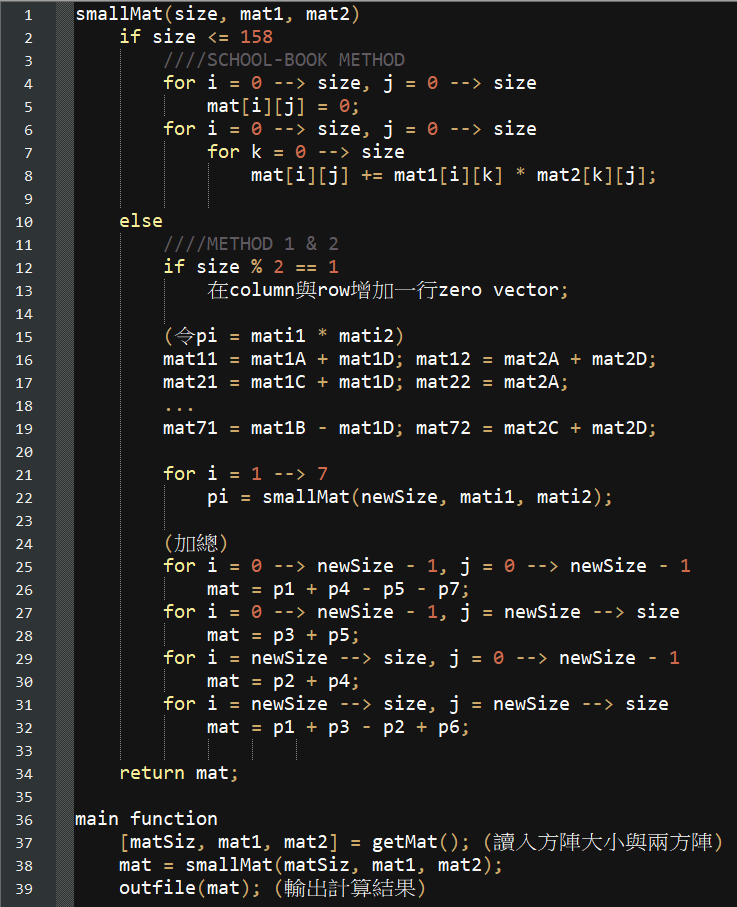
Programming Assignment #1 0510006 電資09 張若暄

Multiplication of Two Matrices

1. 作業目標

分別以*School-Book Method*、*Method 1*、*Method 2*計算兩方陣相乘結果並輸出於指定文字檔，分析並比較三種方式的performance；此外，試結合三種方式達到最高效率。

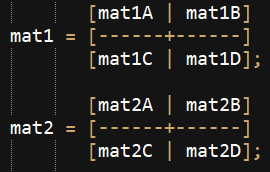
1. Method Introduction
   1. Psuedo Code

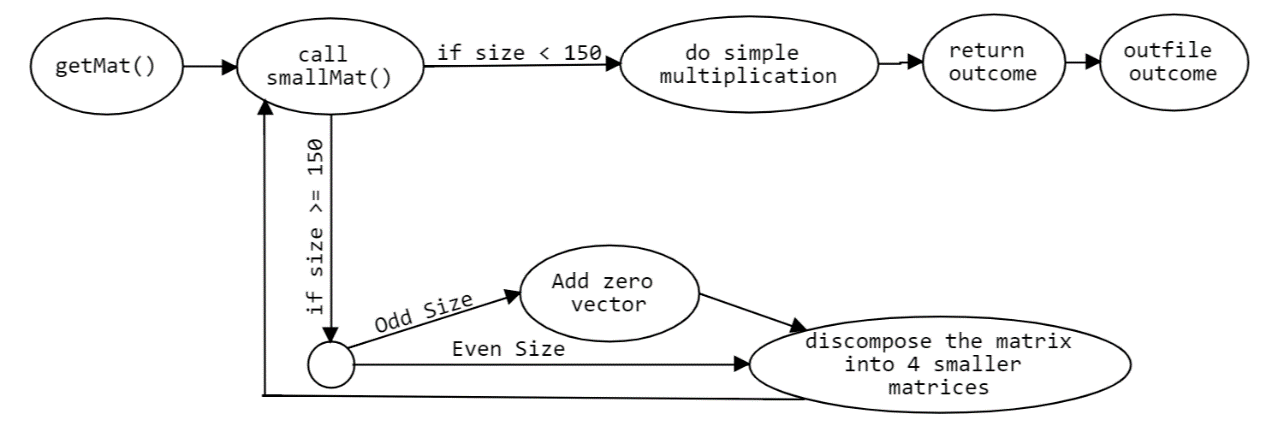
程式整體可以簡單分成三部分：

* + - * **主程式：line 36 ~ 39的main func.**
      * **Recursive終止區：line 2 ~ 8**

處理size ≤ 158的狀況，直接利用*School-Book Method*即可達到較高效率。

* + - * **Recursive區塊：line 12 ~ 32**

處理size > 158的狀況，處理size可能為奇數的問題後，參考*Method 1*將mat1、mat2按照下圖分成4個小方陣後按照*Method 2*的優化算法得出P1 ~ P7，進而相加減得到相乘矩陣。

* 1. Flowchart
  2. 遭遇困難：**MatSiz為奇數時如何拆矩陣？**

解決方法：

為了使拆矩陣的行為次數降至最低，小方陣的size必須為大方陣的一半，對於size為奇數的矩陣，拆開之前先額外加上一個column與一個row的全zero vector，使size由原本的2k+1轉為2k+2，進而拆成size = k + 1之小方陣。

(對照Pseudo Code中line 12 ~ 13的部分)

1. Analyzation
   1. ***School-Book Method***

簡單分析Pseodo Code中line 4 ~ 8，第二個for迴圈i、j、k都要從0增加至size – 1，故當size = n，需要n3次iterations，可得到**Order of Growth = Θ(n3)**。

* 1. ***Method 1***

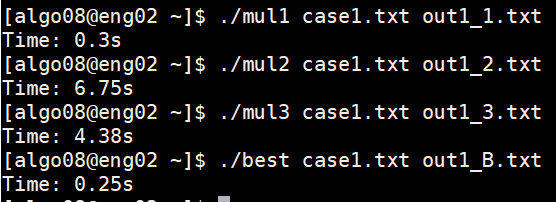
將*Method 1*簡單分析如下：

|  |  |
| --- | --- |
| 步驟 | 次數 \* 時間 |
| 1 \* 1 mat | T(1) 🡪 Θ(1) |
| 補入zero vector (size為奇數時才需要補) | Θ(1) or Θ(0) 🡪 Θ(1) |
| 呼叫smallMat() | 8 \* T(⌈(n+1) / 2⌉) |
| 矩陣相加 | Θ((⌈(n+1) / 2⌉)2) 🡪 Θ(n2) |

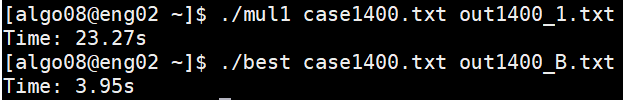
* 1. ***Method 2***

與*Method 1*相似，唯一不同處為呼叫smallMat()之次數減為7。

略為調整T(n)的計算過程如下：

1. Experiment Results
   1. **Size = 500** (tests on provided data: *case1.txt*)

右圖中，*mul1*、*mul2*、*mul3*分別為使用*School-Book Method*、*Method 1*、*Method 2*產生的執行檔；*best*為依照不同size大小選用不同method的程式(如Pseudo Code所描述)。由執行時間可以觀察出以下幾個特性：

* + 1. *School-Book Method*的效率較*Method 1*、*Method 2*高的原因為*Method 1*相較於*School-Book Method*不僅沒有減少計算量，更需花額外時間呼叫記憶體；*Method 2*雖然少了1/8次的計算量，但在拆至size較小時，額外呼叫記憶體的時間大於減少的計算時間，因此效率依舊較*School-Book Method*低。
    2. *Method 2* 效率高於*Method 1*顯示每八次乘法減少一次可以有效增加程式效率。
    3. 優化程式(*best*)的效率明顯高於*Method 1*、*Method 2*，但與*School-Book Method* 相較並沒有加快非常多，因為size還相當小，在拆成原本的1/4後即直接使用*School-Book Method*了，因此在執行時間上並沒有明顯差異。
  1. **Size = 1400** (自行產生的data file)

在提高Matrices的大小後，執行時間明顯的減少。與*School-Book Method*相較，size較大時，呼叫記憶體的時間小於減少一次乘法所節省的時間，使程式速度加快許多；且size較小時，改採用*School-Book Method*以避免呼叫記憶體耗去過多時間，直接算出矩陣乘積以達最高效率。