(1) A = $\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$, B = $\begin{bmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{bmatrix}$

Let *b11, b21, b31* is B1, *b12, b22, b32* is B2, and *b13, b23, b33* is B3. Then B = $[B1 \quad B2 \quad B3]$.

AB = $[AB1 \quad AB2 \quad AB3]$

Let *a11, a12, a13* is A1, *a21, a22, a23* is A2, and *a31, a32, a33* is A3. Then A = $\begin{bmatrix} A1 \\ A2 \\ A3 \end{bmatrix}$.

AB = $\begin{bmatrix} A1B \\ A2B \\ A3B \end{bmatrix}$.

Let A = $[A1 \quad A2]$, A1 is m*s, A2 is m*(n-s), and B = $\begin{bmatrix} B1 \\ B2 \end{bmatrix}$, B1 is s*n, B2 = (n-s)*n.

Then $Cij = \sum_{k=1}^{n} Aik * Bkj = \sum_{k=1}^{s} Aik * Bkj + \sum_{k=s+1}^{n} Aik * Bkj$.

If a matrix X = $\begin{bmatrix} X11 & X12 \\ X21 & X22 \end{bmatrix}$, and a matrix Y = $\begin{bmatrix} Y11 & Y12 \\ Y21 & Y22 \end{bmatrix}$, then if matrix C = XY, by above proof, we know that X1 = $\begin{bmatrix} X11 \\ X21 \end{bmatrix}$, X2 = $\begin{bmatrix} X12 \\ X22 \end{bmatrix}$, Y1 = $[Y11 \quad Y12]$, Y2 = $[Y21 \quad Y22]$, and

XY = $[X1 \quad X2]\begin{bmatrix} Y1 \\ Y2 \end{bmatrix}$ => X1*Y1 = $\begin{bmatrix} X11 * Y1 \\ X21 * Y1 \end{bmatrix}$ = $\begin{bmatrix} X11 * Y11 & X11 * Y12 \\ X21 * Y11 & X21 * Y12 \end{bmatrix}$.

And so on, we can get C = $\begin{bmatrix} X11 * Y11 & \cdots & X1k * Ykn \\ \vdots & \ddots & \vdots \\ Xm1 * Ym1 & \cdots & Xmk * Ykn \end{bmatrix}$

= $\begin{bmatrix} C11 & \cdots & C1n \\ \vdots & \ddots & \vdots \\ Cm1 & \cdots & Cmn \end{bmatrix}$.

(2) Equation 1:

By the question, we know the number of additions is n^3 and number of multiples is n^3.

Equation 2:

M = n/q, and M is new columns of A (new rows of B). Similarly, N = n/p, and N is new row of A (new columns of B). For Cij, it will be added and multiplied M times, and for C, it exists N*N element, so it need calculated N^2 times.

For Cij, there are p*p elements and for each element we need calculated M*q (=n) times. Hence, for each Cij, the number of added and multiplied is p*p*n (n * n^2/N^2). Totally, there are N^2 elements, so the number of added is N^2 * n * n^2/N^2 = n^3, and the number of multiplied is N^2 * n * n^2/N^2 = n^3

(3)

執行情形：

```
4.642643  4.033143  3.859925  3.865853  4.616160  5.022504  5.099792
3.655306  3.604086  3.503014  3.608221  4.483868  4.912957  5.043349
3.534409  3.464788  3.466462  3.606980  4.423390  4.880773  4.985149
3.586524  3.505883  3.542294  3.601915  4.396093  4.798906  4.897858
3.947464  3.542253  3.470878  3.575130  4.364373  4.769017  4.820594
3.970912  3.545517  3.459854  3.572529  4.352968  4.762901  4.888511
3.982609  3.540711  3.685007  3.917209  4.966024  5.064197  4.960946
Program ended with exit code: 0
```

環境：

OS : macOS 10.15.5

CPU : 2.4GHz 4 cores intel Core i5

IDE : Xcode v12.0.1

(4)

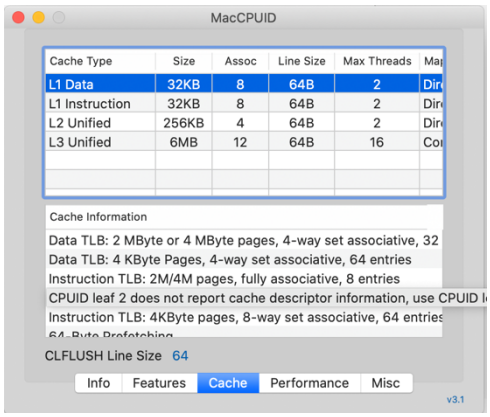
p \ q	4	8	16	32	64	128	256
4	4.642643	4.033143	3.859925	3.865853	4.616160	5.022504	5.099792
8	3.655306	3.604086	3.503014	3.608221	4.483868	4.912957	5.043349
16	3.534409	3.464788	3.466462	3.606980	4.423390	4.880773	4.985149
32	3.586524	3.505883	3.542294	3.601915	4.396093	4.798906	4.897858
64	3.947464	3.542253	3.470878	3.575130	4.364373	4.769017	4.820594
128	3.970912	3.545517	3.459854	3.572529	4.352968	4.762901	4.888511
256	3.982609	3.540711	3.685007	3.917209	4.966024	5.064197	4.960946

(5)

a.

快取(cache)是一種暫存的記憶體。在 CPU 處理資料的時候會先從 cache 中找尋資料，如果從中有找到暫存的資料或指令，就不需要到 main memory 中找尋，因為 CPU 的處理速度大多比 maim memory 讀取速度快，使用 cache 較能有效的利用資源。此外，現今的 cache 還會分成 L1、L2、L3 等等，依據層級的不同，cache 的儲存量及存取速度也都不同。

b. cache size of my computer.



c. Explain the reason of the performance differences for various p and q.

因為 matrix 的乘法會不斷讀取到 cache，但一般的 matrix multiple 無法有效的利用從 cache 拿出來的資料，造成時間的浪費，因此使用 block matrix multiple。

以我的電腦來看 cache L1 Data 的 size 是 32KB，而 size of cache line size 是 64byte，也就是共有 512 個 line，而一條 line 可以放 16 個 int。因此可以看出來當 p 相同的時候，q 是 16 的使用時間是最少的，這是因為可以一次拿出一條 cache 所有的資料並全數用完。當 q 大於 16，就需要讀到第二條 cache。而如果 q 小於 16，同一條 line 則需要重複讀取，浪費掉了資源。