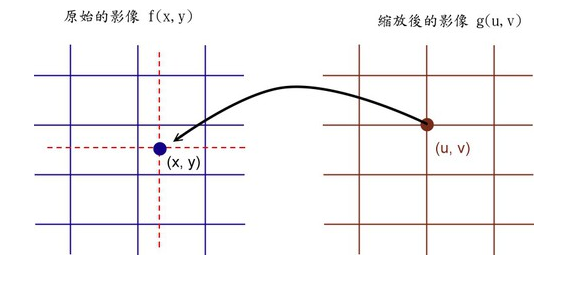
1. Zooming and shrinking

(a).



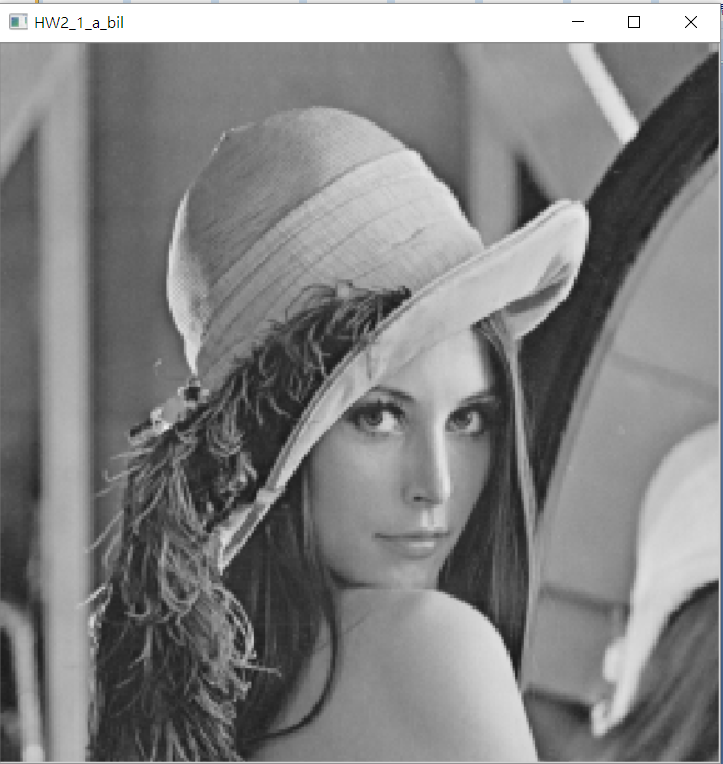
Nearest neighbor



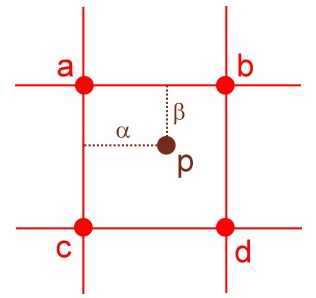
離 (x, y) 最近的整數點 (xout, yout)  
以 f(xout, yout) 代替未知的 f(x, y)  
而 f(x, y) 就是縮放後 g(u, v) 的值  
xout = int (x+0.5)  
yout = int(y+0.5)

邊界條件:

因為NNI取最接近的位置，所以用if條件句限制xy邊界大於256的話就要一直減一減到小於255。



Bilinear



a, b, c, d 四個點的灰度值分別為 f(a), f(b), f(c), f(d)  
現在要來求 f(p)，假設 p 至 a 的水平距離為 alpha，假設 p 至 a 的垂直距離為 beta，由於abcd四點都是相鄰間距為1的格子點上，所以 p 至b 的水平距離為 (1-alpha)，所以 p 至c 的垂直距離為 (1-beta)，其它依此類推...

bilinear 的精神就是影響力與距離成反比，所以

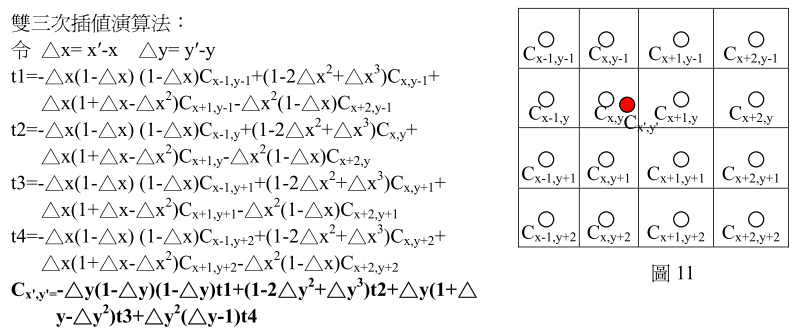
f(p) = (1-alpha)\*(1-beta)\*f(a)+alpha\*(1-beta)\*f(b)+(1-alpha)\*beta\*f(c)+alpha\*beta\*f(d)

邊界條件:

若新的點靠近邊界，相鄰位置可能會超過原始影像的邊界位置，所以必須用if條件句分別約束bcd三個點的xy位置，如果大於256的話就要一直減一減到小於255。



Bicubic



與雙線性內插法類似，將鄰近四點改為尋找鄰近周圍十六個點

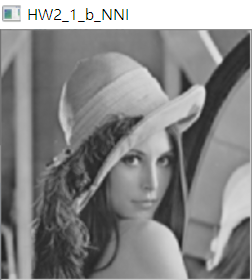
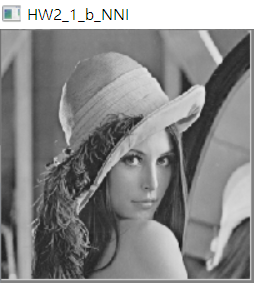
邊界條件:

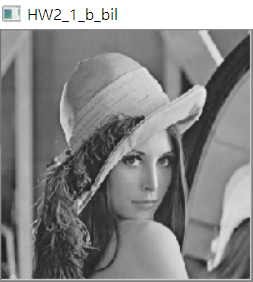
若新的點靠近邊界，相鄰位置可能會超過原始影像的邊界位置，必須用if條件句來分別判斷超出格子數，並依照格子數將圖片內移。

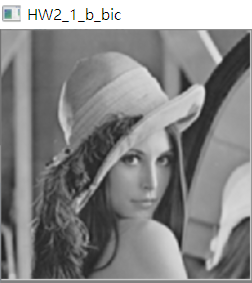
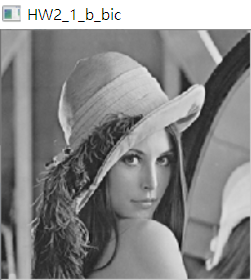
比較:理應來說 Nearest neighbor 放大後可以明顯的看出格子狀，失真許多。Bilinear 則好一點，改善Nearest neighbor 格子狀的狀況。Bicubic 雖然好像有更平滑一些，有一些值卻會被比較遠的值影響到，導致有些顏色有許多失真的情況發生。但我的三張圖片，卻沒有太大的差別，Nearest neighbor跟Bicubic兩者到是有格子狀的差距。

(b)

With blurring without blurring

縮小與放大的方法相同，只是修改倍數而已。效果則跟放大差不多，只是先模糊再縮小，不會有如此明顯的鋸齒現象。

(c)compare



Nearest neighbor 放大2.25倍



Nearest neighbor 先放大5.76倍後縮小2.56倍



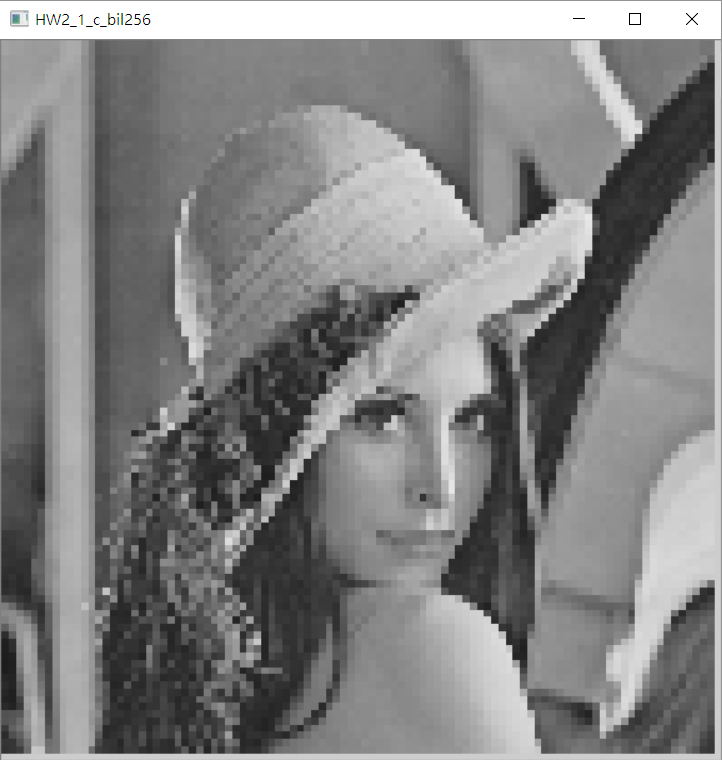
Nearest neighbor 先縮小2.56倍後放大5.76倍



bilinear放大2.25倍



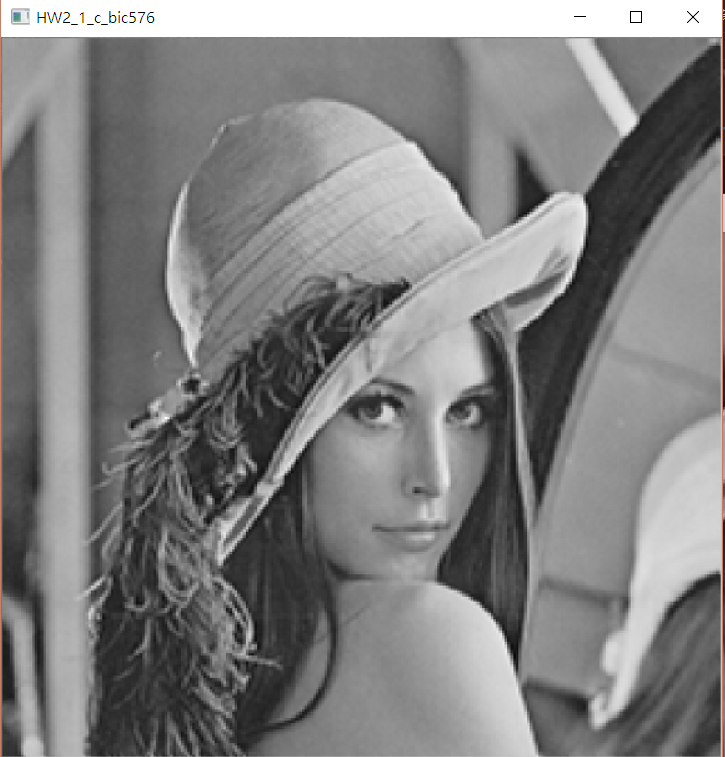
bilinear 先放大5.76倍後縮小2.56倍



bilinear 先縮小2.56倍後放大5.76倍



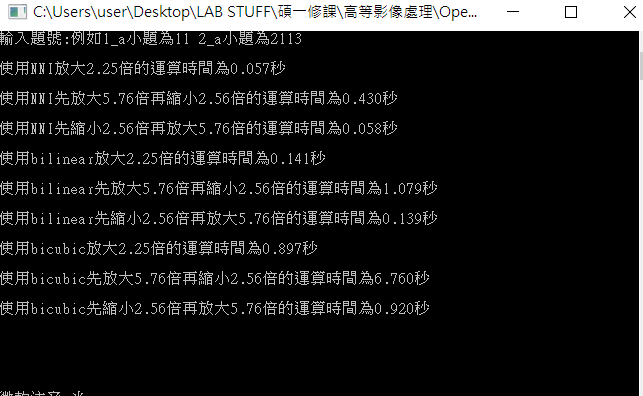
bicubic 放大2.25倍



bicubic 先放大5.76倍後縮小2.56倍



bicubic 先縮小2.56倍後放大5.76倍

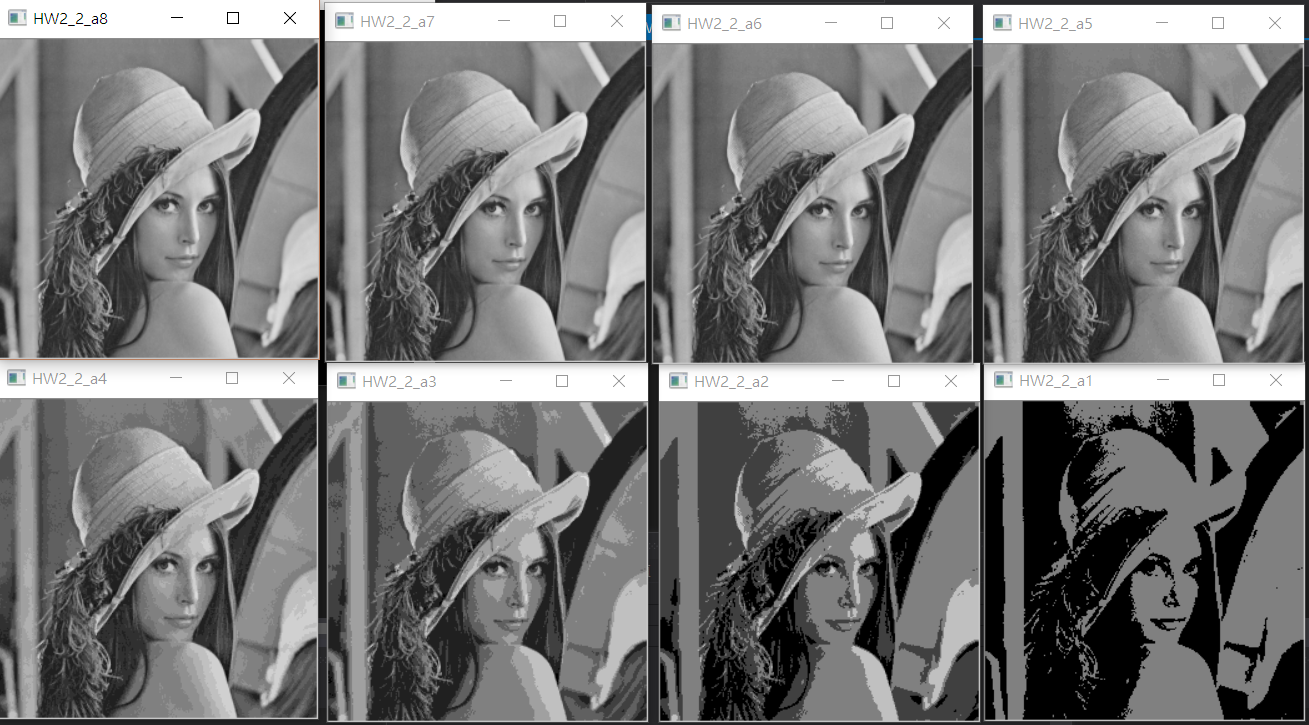


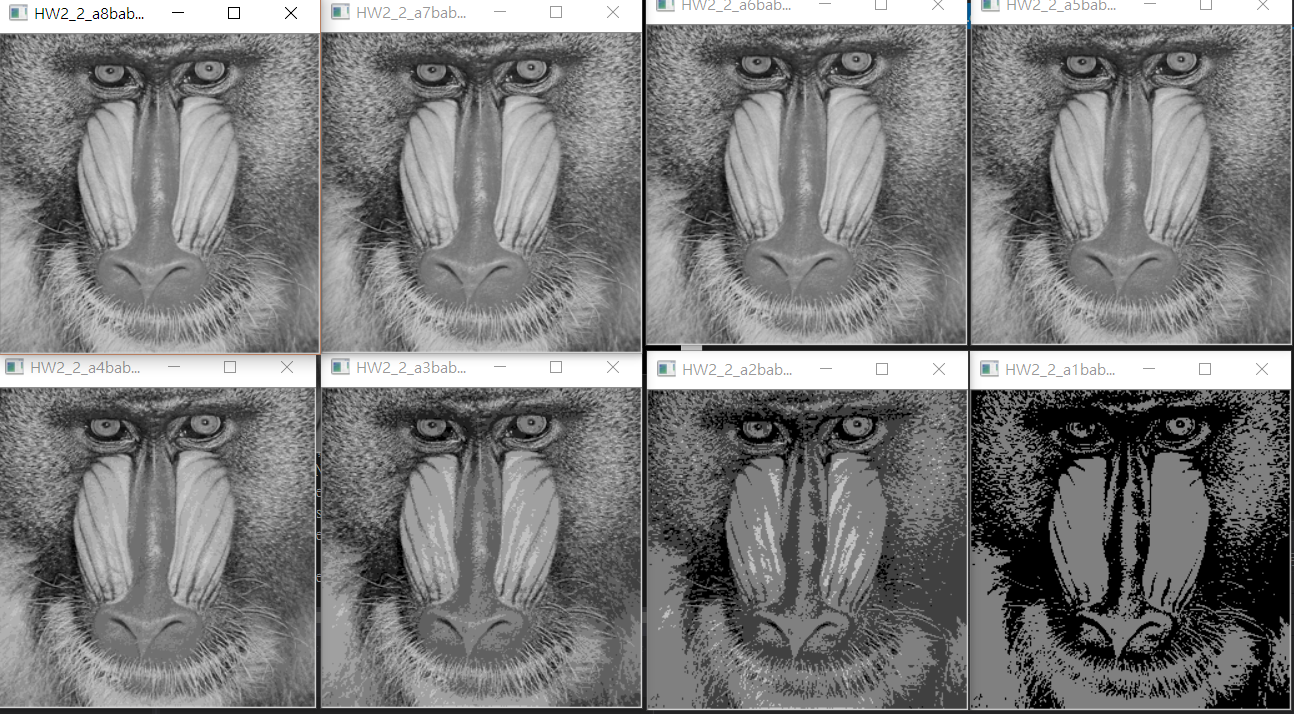
計算時間

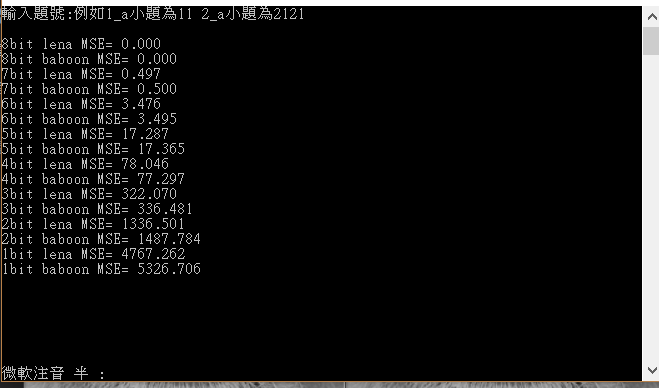
討論：不管是哪種方法，先將圖片縮小再放大，失真情形最嚴重，而直接放大則是最不會失真的，先放大在縮小則是介於兩者之間。因為先縮小會導致許多點的pixel值塞在一個點裡面，因此再放大後就會有失真的現象。

2.Gray-level resolution

(a)

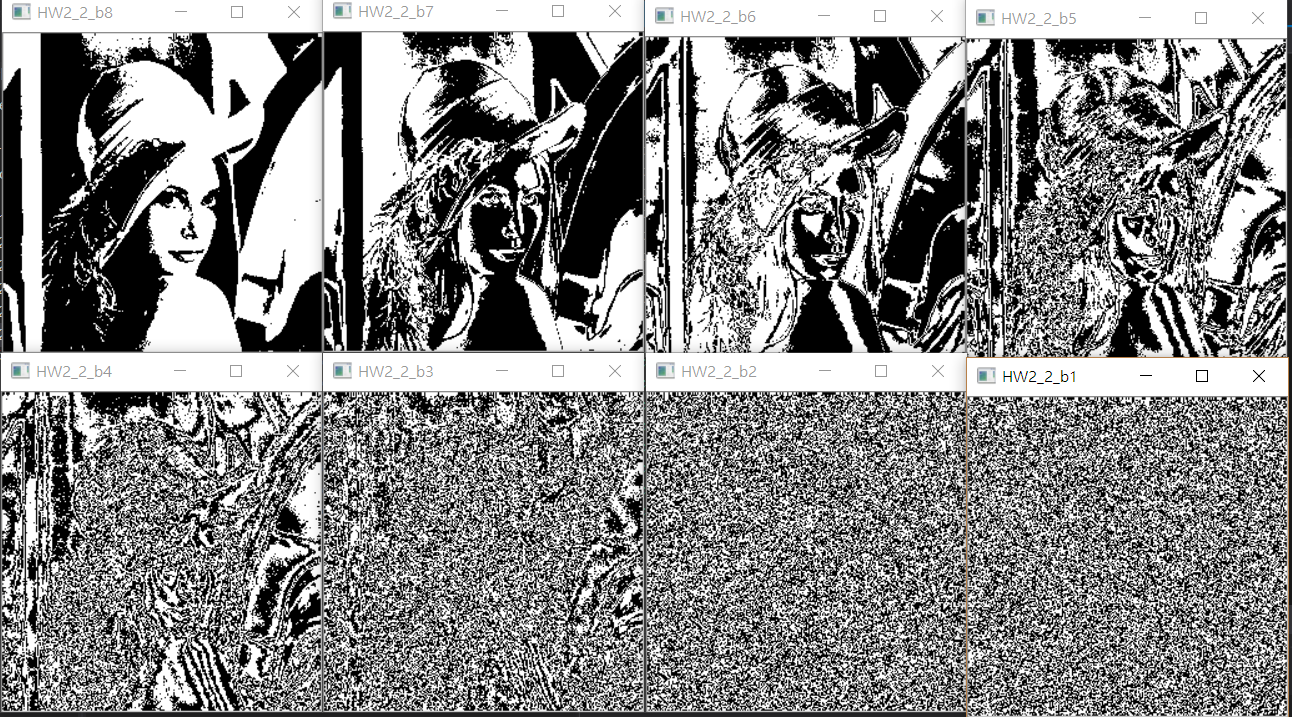


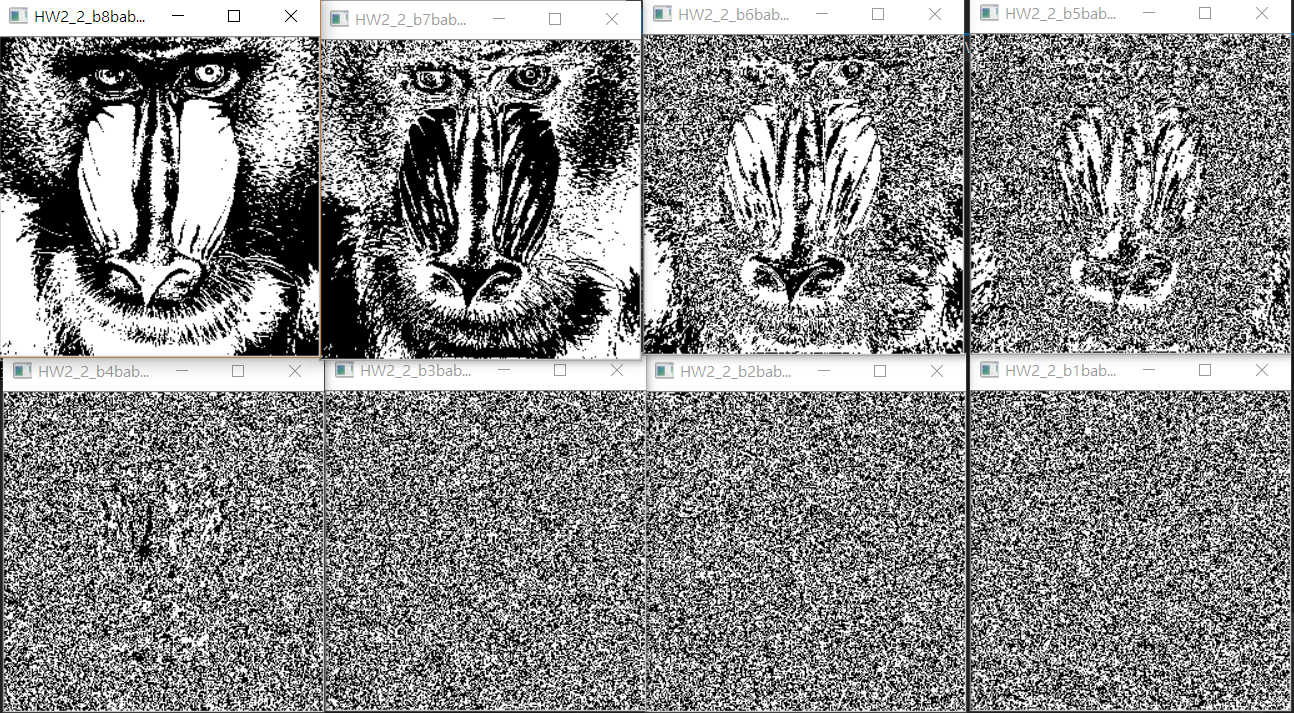




討論:不管是lena或者是baboon，圖片失真越大，MSE也越大。

(b)





討論:原先2(a)小題是使用左右位移的方式來進行quantize，現在使用更快的and判斷更好理解。從結果可以看出我們將圖片2值化後，就可以看出LSB都是雜訊。

3

討論:

(a)先將圖片上面的pixel值讀出來，並將0變成1，其他數值變成0，並判斷左、左上、上、右上、右、右下、下、左下這8個順序判斷，先上下左右，再判斷斜線方向，若有兩個以上的方向有值則設為節點。其中一個方向有值則優先將路徑走向那個方向，若都沒有則回到上一個節點。以此類推

(b)同a，只是把70~220都變成1，其他值變成0。