國立雲林科技大學

112學年度第二學期

**數位影像處理**

第一次作業

班 級： 四電子四A

學 號： B10813029

學 生： 李睿聰

授課教師： 藍呂興 老師

繳交日期： 112年5月1日

目錄

Problem 1

題目說明. 3

本程式之演算法 3

製作方式 4

主程式碼 5

程式碼解說 8

討論與分析 10

Problem 2

題目說明.

本程式之演算法

製作方式

主程式碼

程式碼解說

討論與分析

Problem 3

題目說明.

本程式之演算法

製作方式

主程式碼

程式碼解說

討論與分析

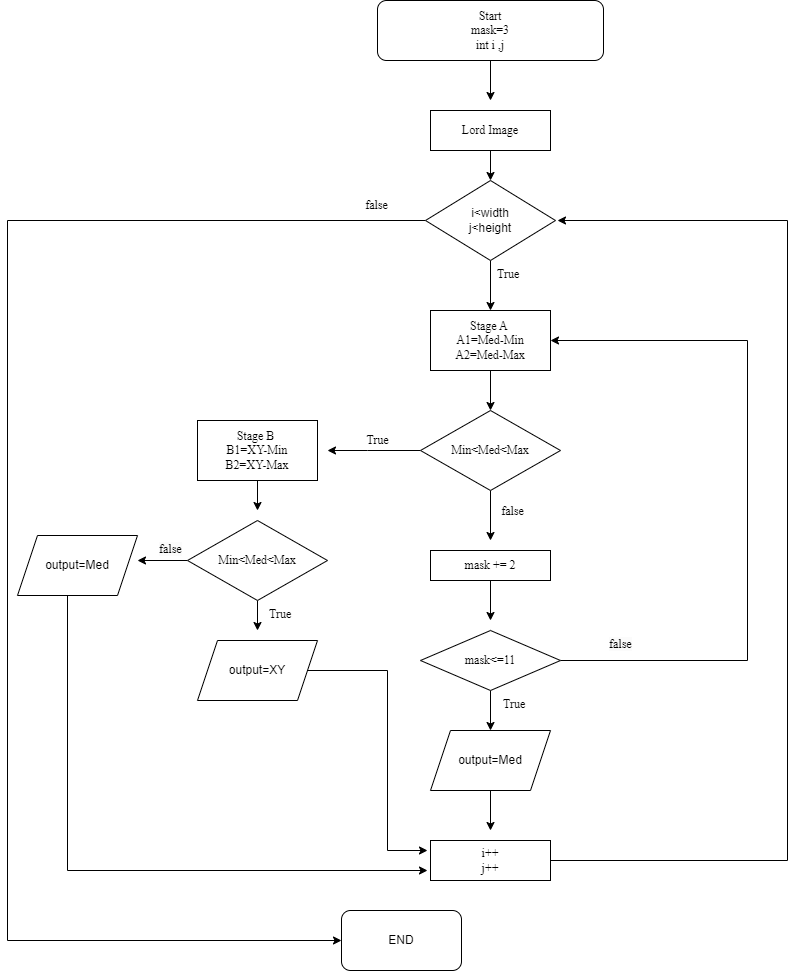
心得分享

1. Problem 1

1.題目說明

使用Adaptive Median Filtering來達到目的，其方式為採用兩個stage去選擇輸出的值，此方法常使用於解決pepper-and-salt 的雜訊，後面將會詳細說說明演算法的架構。

2.本程式之演算法



3.製作方式

初始的Mask為3\*3，從起始點(Figure1.)開始做Mask並將值依大小排列，找出最大值(Max)、最小值(Min)與中間值(Med)並保存原圖點上的值(xy)，若mask超出圖範圍一律將值設為0，將排列完的值開始做Adaptive Median Filtering (Figure2. & Figure 3.)。

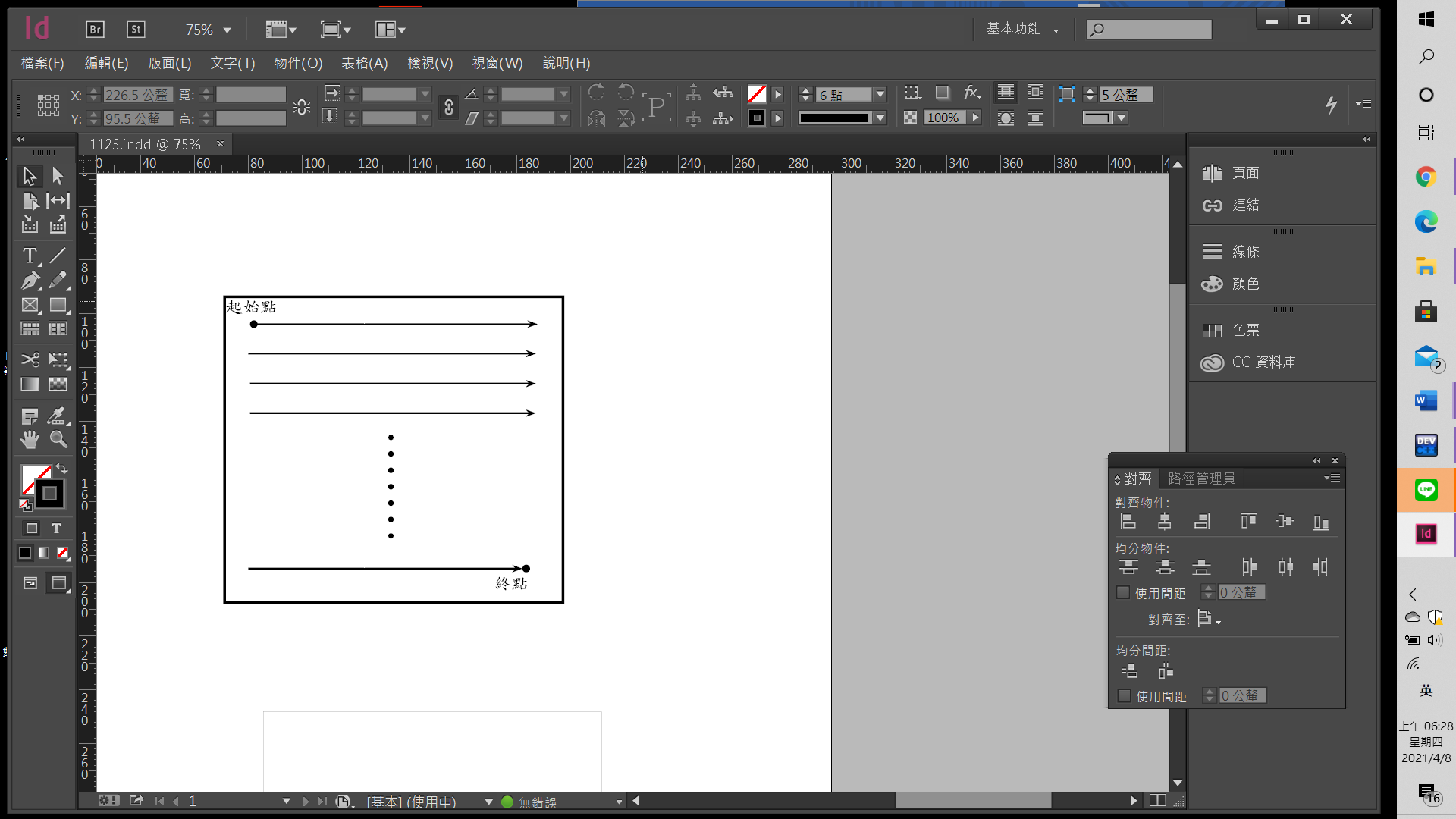
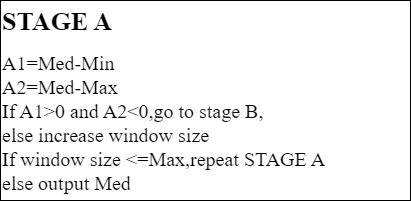
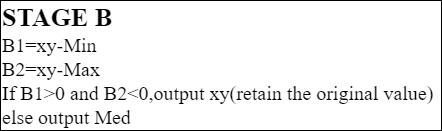


Figure 1. 陣列中運算順序



STAGE A



STAGE B

4.主程式碼

|  |
| --- |
| /\*===========================================================================  This demonstrative example is provided by the teaching assistant,  Mr. Shih-Hung Liao (廖世宏), and modified by the instructor, Prof. Lan.  (1) It can be compiled and executed correctly under the DEV-C++, and Visual C++  environments.  (2) In order to run this program, you should also have the "bmp.h" and  "bmp.cpp" files installed in your current directory or whichever directory  the C++ compiler is directed to search for.  (3) The DEV-C++ is a free C++ development environment that is recommended for  this course. Refer to http://www.bloodshed.net/dev/devcpp.html.  Apr. 3, 2006  ============================================================================\*/  #include <iostream>  #include <stdlib.h>  #include "bmp.h"  using namespace std;  int R[MaxBMPSizeX][MaxBMPSizeY]; // MaxBMPSizeX and MaxBMPSizeY are defined in "bmp.h"  //int G[1024][1024];  //int B[1024][1024];  int r[MaxBMPSizeX][MaxBMPSizeY];  //int g[1024][1024];  //int b[1024][1024];  void sort(int\* , int );  int main(int argc, char\* argv[])  {  int width, height;  open\_bmp((char \*)"lena\_pepper\_and\_salt\_noise10%.bmp", R, R, R, width, height); // for gray images  int arrTmp[121];  int appendData = 0;  int flag\_oversize = 0;  int min, max, idx;  for (int i=0; i < width; i++){  for (int j=0; j < height; j++){  flag\_oversize = 0;  int mask = 3;  // ====================== stage A ======================  while(mask <= 11){  min = 255;  max = 0;  // run the mask windows  idx = 0;  for (int m = -1\*((mask - 1) / 2); m < ((mask - 1) / 2) + 1; m++) {  for (int n = -1 \* ((mask - 1) / 2); n < ((mask - 1) / 2) + 1; n++) {  if (i + m >= width || i + m <= -1 || j + n >= width || j + n <= -1)appendData = 0;  else appendData = R[i + m][j + n];    //max & min  if (appendData > max)max = appendData;  if (appendData < min)min = appendData;  arrTmp[idx++] = appendData;    }  }  A  sort(arrTmp, mask\* mask);    //printf("%d %d %d\n", max, min, arrTmp[int(mask \* mask / 2)]);  if ((arrTmp[int(mask \* mask / 2)] - min > 0) && (arrTmp[int(mask \* mask / 2)] - max < 0)) {  break;// go to B  }  else {  if (mask == 11) {  flag\_oversize = 1;  r[i][j] = arrTmp[int(mask \* mask / 2)];  }  }  mask += 2;  }  // ====================== stage B ======================  if (flag\_oversize == 0) {  if((R[i][j] - min > 0) && (R[i][j] - max < 0))r[i][j] = R[i][j];  else r[i][j] = arrTmp[int(mask \* mask / 2)];  }  }  }    save\_bmp((char\*)"new\_lena\_pepper\_and\_salt\_noise10%.bmp", r, r, r); // for gray images  printf("Job Finished!\n");  close\_bmp();  system("PAUSE"); /\* so that the command window holds a while \*/  return 0;  }  void sort(int\* arrIn, int len) {  for (int i = 0; i < len; i++) {  for (int j = 0; j < len; j++) {  if (\*(arrIn + i) < \*(arrIn + j)) {  \*(arrIn + i) = \*(arrIn + i) ^ \*(arrIn + j);  \*(arrIn + j) = \*(arrIn + i) ^ \*(arrIn + j);  \*(arrIn + i) = \*(arrIn + i) ^ \*(arrIn + j);  }  }  }  }  //a=a xor b  //b=a xor b =(a xor b) xor b = a  //a=a xor b =(a xor b) xor a = b |

5.程式碼解說

取3\*3中Mask的值

圖中m為x軸、n為y軸，若Mask為3則x軸從-1取到1，y軸也是從-1取到1，若取到的點位置超出Mask則取0，若沒有則取原值。

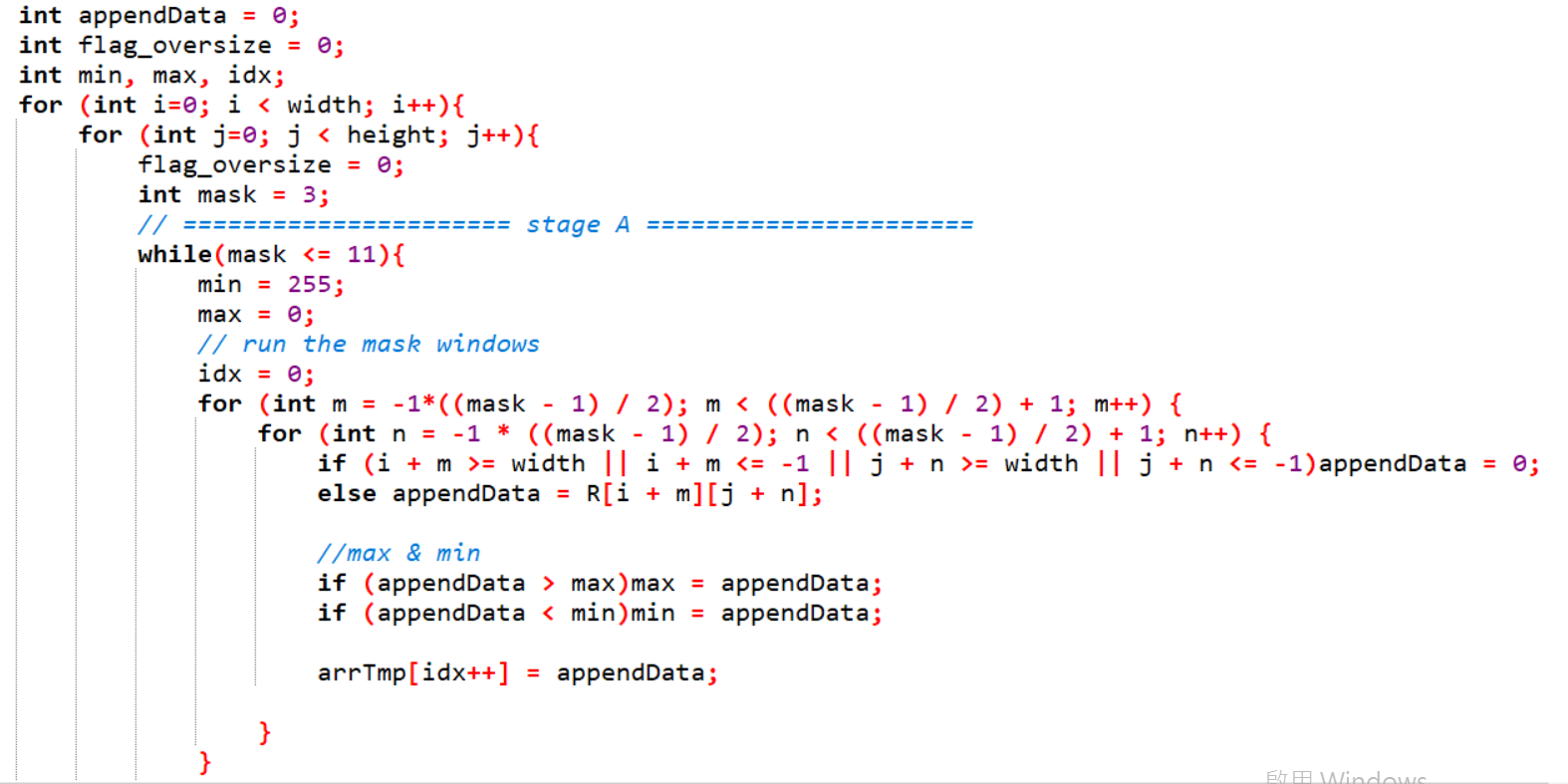
arrTmp為將原值儲存起來。

Figure 5.

Sort 為排序大到小排序，並使用xor的方法做排序

例：

a= a xor b

b= a xor b = (a xor b) xor b = a

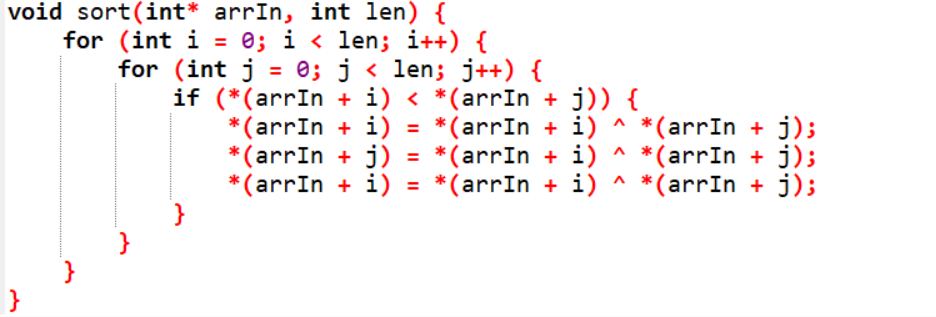
a= a xor b = (a xor b) xor a = b

Figure 6.

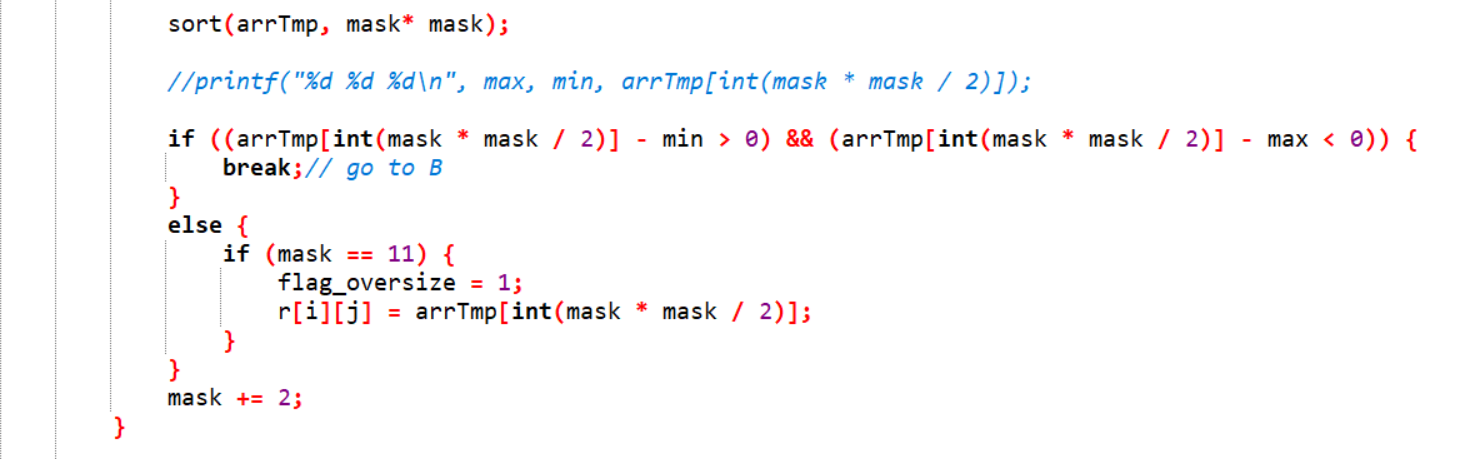
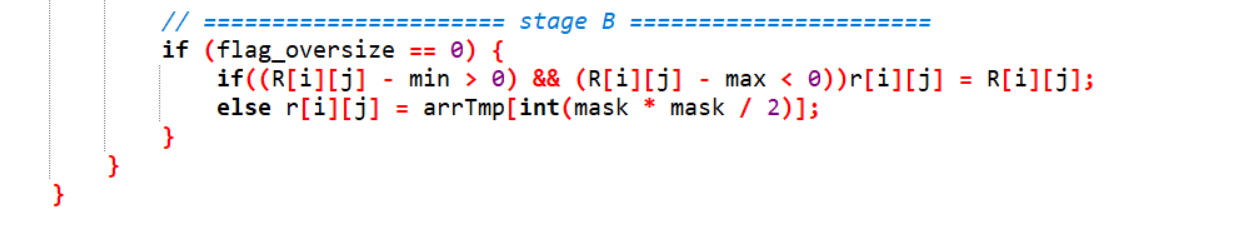
Stage A 如果Min<Med<Max則go to stage B，如果沒有則增加window size並重複stage A，若window size等於最大，則輸出Med值。

Figure 7.

flag\_oversize為判斷window size是否為最大

若Min<原值<Max則輸出原值，反之則輸出Med



6.討論與分析

使用Adaptive Median Filtering來達成本次目的

Figure 8.原始圖 Figure 9. Adaptive Median Filtering

心得分享

在做第一次作業時覺得要把一些演算法用程式表現出來並不是這麼容易，應該要好好的先把架構寫出來再一步一步的換成程式，以及花費更多的時間去嘗試，雖然理論方面能夠很直覺的大概知道結果的樣子，但跟實際用程式模擬的結果感覺更不一樣，希望我下次作業能完整做完。