多處理機平行程式設計 2021 fall 作業四說明

題目

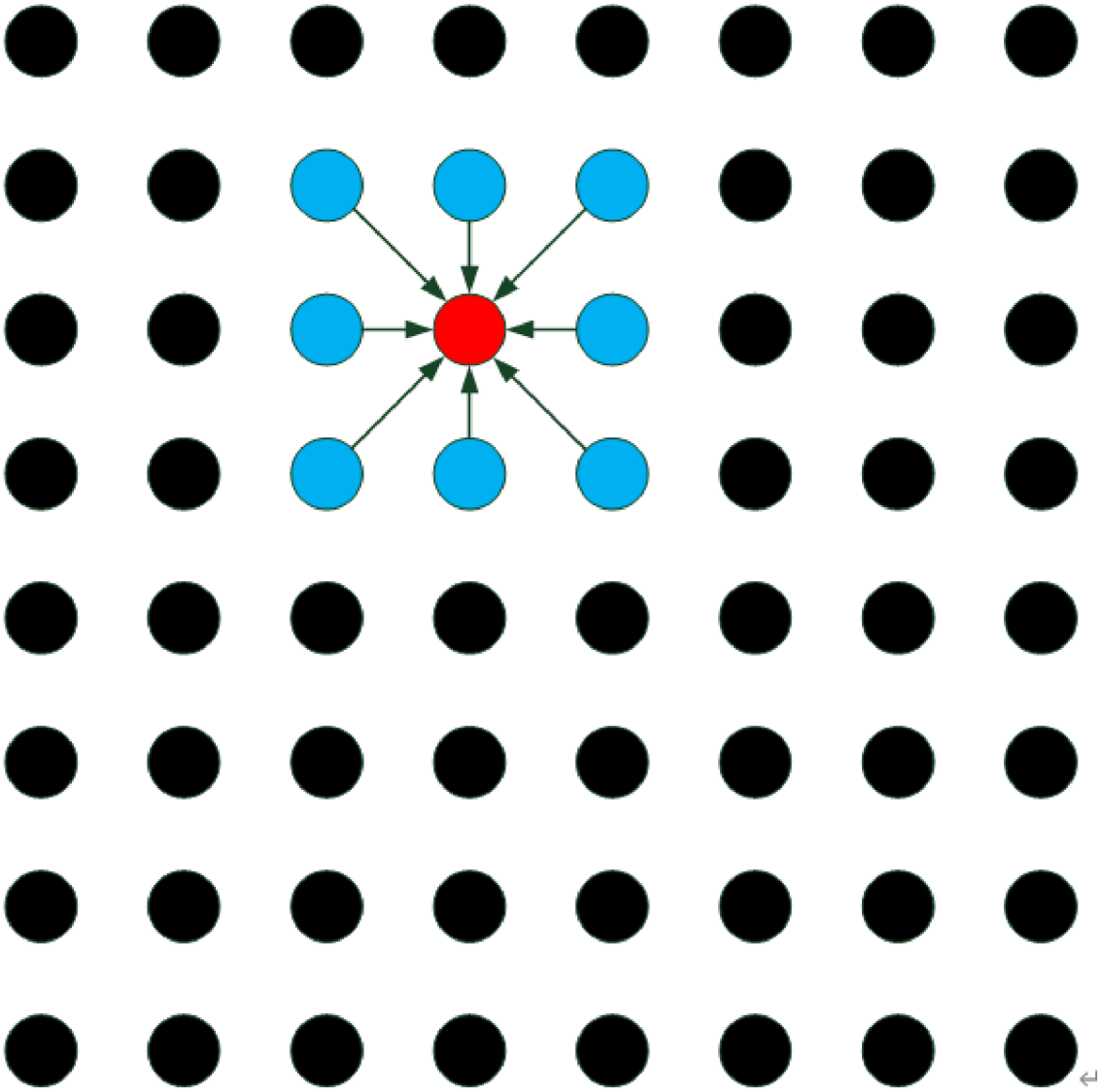
1. 影像平滑

請使用Pthreads來實作，Barriers使用：

1. Busy-waiting and a Mutex
2. semaphores
3. Condition Variables

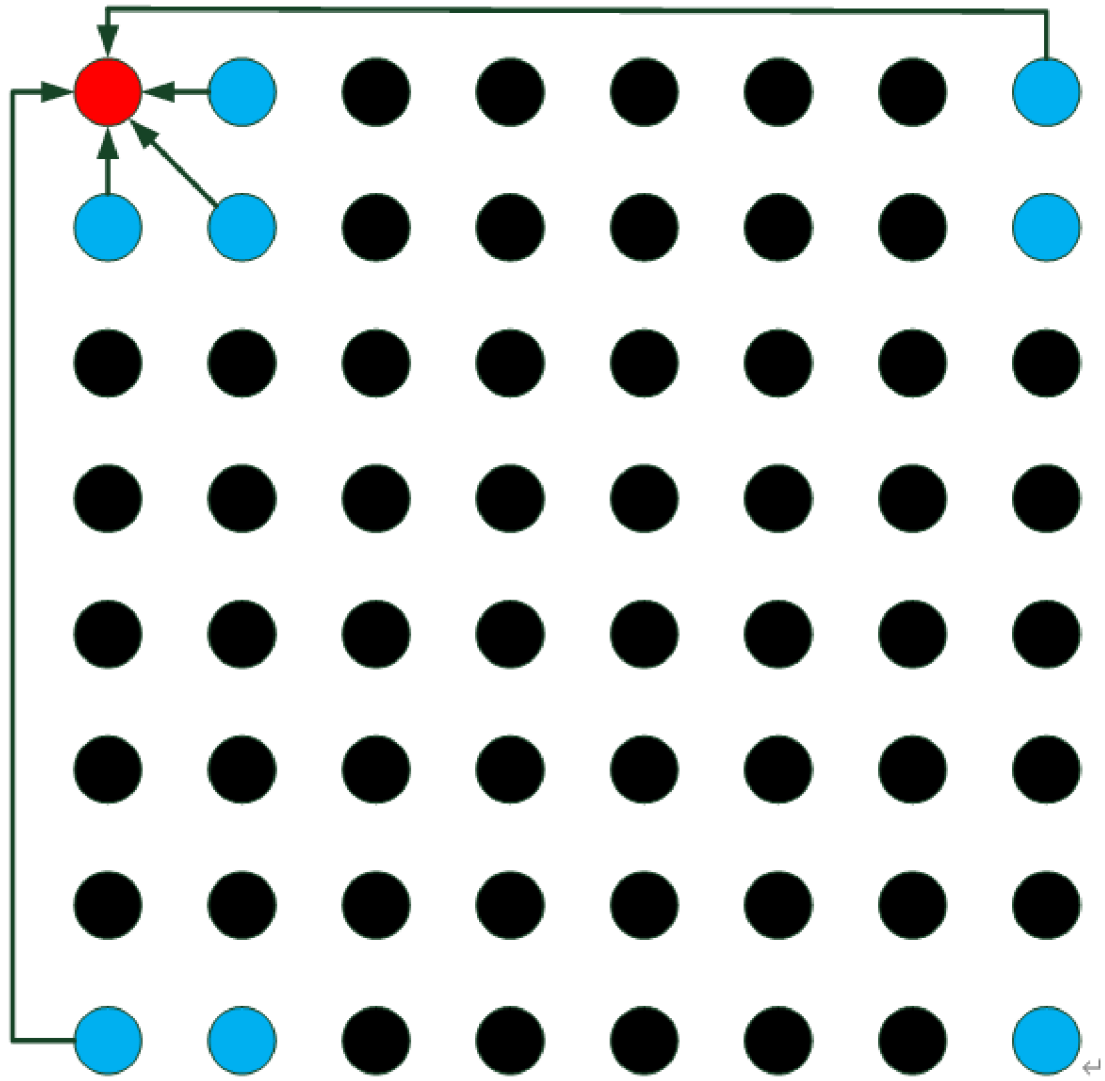
- 三選一即可 -

數位化影像處理 (Digital Image Processing) 中的影像平滑運算 (Image smoothing) 可以消除或衰減影像中的雜訊或輪廓。其中一種方式為空間領域（局部）平均法，其方法如圖一所示：



▲圖一：像素平均示意圖

每一個點裡的像素都會與其周圍的像素進行平均。如果該像素點在邊界中，如圖二所示，則會與對面的像素進行平均運算。



▲圖二：邊界像素平均示意圖

現在我們有一個單機的平滑化運算程式，此程式讀取一個 BMP 圖檔 (位元深度 24 bits)，其像素資料儲存在一個結構矩陣裡：

$$A[Height][Width] = \begin{bmatrix} BGR & \dots & BGR \\ \vdots & \ddots & \vdots \\ BGR & \dots & BGR \end{bmatrix}$$

其中 B 儲存像素 Blue 的資料， G 儲存像素 Green 的資料，R 儲存像素 Red 的資料，且 BGR 的資料型態均為位元 (Char)。每次平滑運算會分別把 BGR 與周圍的 BGR 進行平均運算。舉例來說，當我們把圖三進行 1000 次平滑化運算後，會得到圖四的結果。請試著將此程式進行平行化，並且印出執行時間。程式碼與圖片請至 moodle 下載。原程式碼使用 C++ 撰寫，可以繼續

使用 C++ 改寫，或是自行改寫成 C。

▲圖三：原始圖片

▲圖四：1000次平滑化後的結果

如何交作業

1. 把作業交到你的家目錄底下 (`mv yourfile.c ~`) 並使用正確的檔名: `h4_problem1.c` (or `.cpp`)，想用資料夾稍作整理也可以，但是請確保只有一個 `h4_problem1.c` NEW。 請勿抄襲。
2. 你可以使用自己的電腦寫此次作業。課程之伺服器 `gcc` 版本為 `4.5.1` ，若使用不同版本應該不會有影響，不過還是請在 report 中註明你的 `gcc` 版本以利發生問題時 debug。

3. 上傳你的 report (使用 `.md` 或是 `.pdf` 格式，或是包含連結至你的 HackMD 頁面 / GitHub Readme 的文字檔 NEW)(in Chinese or English) 至 moodle 並包含：
    - What have you done
    - Analysis on your result
    - Any difficulties?
    - (optional) Feedback to TAs

截止日期： 2021/12/10 23:59:59

Please report any server mis-configuration you found. TAs are new to System/Network administration. We will appreciate your report.

另外，pn1 上已安裝 `perf` ，有在 pn1 上測試的同學可以試試看。NEW

計分方式

- 程式 style 25%
    1. 巢狀結構需用階層式編排。
    2. 適當地使用空白列來區隔功能上無關的程式碼，使你的程式段落分明。
    3. 清楚且詳細的註解。
- 結果正確無誤 20%
  平行程式執行的結果需與循序程式執行的結果一致。可用 `diff` 指令驗證輸出結果是否正確， `diff` 會印出 2 個檔案的不同處。假設平行程式的輸出檔為 `output1.bmp` ，循序程式的輸出檔為 `output2.bmp` ，執行 `diff output1.bmp output2.bmp` 的結果應該沒有任何輸出
- 效能 30%
- 平行程式的觀察報告 25%
  請觀察作業中，改變不同的平滑化的次數與不同數量的處理程序對執行時間有何影響。請寫出觀察結果，並回答為何會有這種結果，寫在結果分析中

Useful links

- 並行程式設計: POSIX Thread
  (https://hackmd.io/@sysprog/concurrency/https%3A%2F%2Fhackmd.io%2F%40sysprog%2Fposix-threads)
    - Still many useful links inside