**Laborator 1**

**Deadline:** saptamana 4

Se considera o imagine reprezentata printr-o matrice de pixeli, F , de dimensiune (MxN).

Se cere transformarea ei aplicand o filtrare cu o fereastra definita de multimea de indici

Ind[n,m] = {(k,l) | -n/2<=k<=n/2, -m/2<=l<=m/2}

si de coeficientii *wkl*

n,m impare si n<N, m<M.

Transformarea unui pixel:

V[i,j] = {**+** (k,l): -n/2<=k<=n/2, -m/2<=l<=m/2: w[k,l] \* F[i-k ,j-l] }

unde

De exemplu:

multimea de indici este

Ind [3,3] ={ (-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (-0,1), (1,-1), (1,0), (1,1)}

Si ponderile asociate



Actualizarea unui pixel de pe pozitia (i,j)

v[i,j] =

f[i,j] \*1/9+

f[i-1,j]\* 1/9+

f[i,j-1]\* 1/9+

f[i-1,j-1]\* 1/9+

f[i+1,j]\* 1/9+

f[i,j+1]\* 1/9+

f[i+1,j+1]\* 1/9

Pentru frontiere se considera ca un element este egal cu elemental din celula vecina din matrice   
f[-1,-1]= f[0,0]; f[-1,j]= f[0,j]; f[i,-1]=f[i,0]; f[M,j]= f[M-1,j]; f[i,N]=f[i,N-1];

Exemplificare -> https://de.wikipedia.org/wiki/Datei:2D\_Convolution\_Animation.gif

Se cere asigurarea urmatoarei postconditii:

Postconditie: Matricea rezultat V contine imaginea filtrata a imaginii initiale F (unde V != F)

1. Program secvential
2. Program paralel: folositi **p** threaduri pentru calcul.

**Obiectiv:** Impartire cat mai echilibrata si eficienta a calculul pe threaduri!

Pentru impartirea sarcinilor de calcul (taskuri) se foloseste descompunere geometrica care poate fi  
 (puteti alege o varianta sau sa incercati mai multe si sa o identificati pe cea mai buna):

* Pe orizontala (mai multe linii alocate unui thread)
* Pe verticala (mai multe coloane alocate unui thread)
* Bloc – submatrici alocate unui thread
* bazat pe o functie de distributie prin care unui index al unui thread i se distribuie o submultime de indecsi din matrice;

distributia se poate face prin:

- distributie liniara (indici alaturati la acelasi thread) sau

- distributie ciclica( cu pas egal cu p).

**Datele de intrare** se citesc dintr-un fisier de intrare “date.txt”.

(Fisierul trebuie creat anterior prin adaugare de numere generate aleator. Toate rularile trebuie executate cu acelasi fisier.)

**Implementare:**

1. Java
2. C++ ( cel putin C++11 )
   1. matricile sunt alocate static (int f[MAX][MAX] )
   2. matricile sunt alocate dynamic (new…)

Folosire directa a threadurilor (creare explicita) => nu se permite folosirea executorilor.

**Testare:** masurati timpul de executie pentru

1. N=M=10 si n=m=3; p=4;
2. N=M=1000 si n=m=5; p=2,4,8,16
3. N=10 M=10000 si n=m=5; p=2,4,8,16
4. N=10000 M=10 si n=m=5; p=2,4,8,16

Rezultatele acestor teste trebuie sa fie reflectate in documentatie in tabele

**Java:**

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp executie** |
| N=M=10  n=m=3 | secvential | 0,05431 |
| 4 | 1,60328 |
| N=M=1000  n=m=5 | secvential | 88,4006 |
| 1 | 81,62894 |
| 2 | 58,50882 |
| 4 | 42,94364 |
| 8 | 45,62096 |
| 16 | 76,91551 |
| N=10  M=10000  n=m=5 | secvential | 32,93158 |
| 1 | 37,66228 |
| 2 | 49,49241 |
| 4 | 52,96885 |
| 8 | 66,25325 |
| 16 | 94,67041 |
| N=10000  M=10  n=m=5 | secvential | 22,57198 |
| 1 | 18,6589 |
| 2 | 18,63336 |
| 4 | 22,75454 |
| 8 | 35,32658 |
| 16 | 60,21252 |

**C++**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Tip alocare** | **Nr threads** | **Timp executie** |
| N=M=10  n=m=3 | Static | secvential | 0,00691 |
| 4 | 0,70453 |
| dinamic | secvential | 0,00731 |
| 4 | 0,53611 |
| N=M=1000  n=m=5 | static | secvential | 131,0763 |
| 1 | 136,687 |
| 2 | 76,14148 |
| 4 | 40,16231 |
| 8 | 37,19738 |
| 16 | 32,32051 |
| dinamic | secvential | 161,0611 |
| 1 | 162,9107 |
| 2 | 70,109 |
| 4 | 40,23968 |
| 8 | 32,06299 |
| 16 | 30,67815 |
| N=10  M=10000  n=m=5 | static | secvential | 12,66436 |
| 1 | 13,58294 |
| 2 | 6,72848 |
| 4 | 4,19436 |
| 8 | 3,80257 |
| 16 | 3,42035 |
| dinamic | secvential | 13,79221 |
| 1 | 14,70523 |
| 2 | 7,56057 |
| 4 | 4,65615 |
| 8 | 3,79861 |
| 16 | 4,12402 |
| N=10000  M=10  n=m=5 | static | secvential | 13,34775 |
| 1 | 13,84281 |
| 2 | 7,3485 |
| 4 | 4,74231 |
| 8 | 3,94617 |
| 16 | 3,71737 |
| dinamic | secvential | 14,1095 |
| 1 | 14,87931 |
| 2 | 7,72651 |
| 4 | 5,10003 |
| 8 | 4,18721 |
| 16 | 3,94695 |

**ObservatII:**

* Fiecare test trebuie repetat de 10 ori si pentru evaluarea timpul de executie se considera media aritmetica a celor 10 rulari.
* Pentru fiecare varianta (secvential, paralele) folositi acelasi fisier “date.txt”;

Folositi recomandarile din fisierele “Testare” “Verificare corectitudii”.

**Analiza**

Comparati performanta pentru fiecare caz – secvential versus paralel si variantele paralele intre ele.

Comparati timpii de executie obtinuti cu implementarea Java versus implementarea C++.

Comparati cele doua variante pentru implementarea C++.

Analiza trebuie evidentiata in documentatie.