x Laborator 1: Aplicarea unei matrici de convolutie

Timp de lucru: 2 saptamani (deadline – saptamana 4)

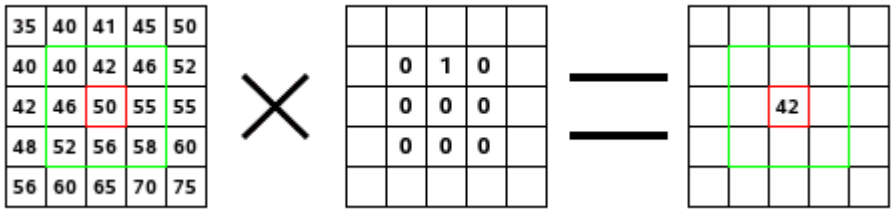
Multe dintre filtrele pe imagini utilizează operatia de convolutie bazata pe matrice de convoluție.

Simplu spus convoluția este tratamentul aplicat unei matrice F de dimensiune NxM (care poate fi o matrice care reprezinta o imagine reprezentata prin pixeli) prin intermediul altei matrice numita matricea de convolutie - C.

De cele mai multe ori se folosesc matrice de convolutie de dimensiune 3x3 sau 5x5. Generalizand putem considera o matrice de convolutie de dimensiune kxk unde k este numar impar.w

Prin aplicarea convolutiei se modifica fiecare element al matricei date initial. Putem sa consideram si cazul in care rezultatul convolutiei se pastreaza intr-o matrice noua V de acealeasi dimensiuni cu matricea initiala F (nxm).

Exemplu pentru matrice de convolutie C de dimensiune 3x3(k=3) aplicat unei matrice date F de dimensiune 5x5 (m=5, n=5)



Calculul efectuat pentru elementul marcat cu rosu cu valoarea 50 este urmatorul:

(40*0)+(42*1)+(46*0) + (46*0)+(50*0)+(55*0) + (52*0)+(56*0)+(58*0) = 42

Generalizand si considerand ca rezulta o matrice noua rezultata in urma aplicarii convolutiei pe care o notam cu V , putem exprima calculul efectuat cu o matrice de convolutie de 3x3 astfel:

```
v[i,j] =
        f[i,j] *c[1,1]+
        f[i-1,j]* c[0,1]+
                f[i,j-1]* c[1,0]+
        f[i-1,j-1]* c[0,0]+
```

f[i+1,j]* c[2,1]+
f[i,j+1]* c[1,2]+
f[i+1,j+1]* c[2,2]

unde 0<=i<n si 0<=j<m.

Se considera ca matricea initiala este bordata virtual cu elemente egale cu elementele de pe frontiera. Astfel f[-1,j]=f[0,j] , f[n,j]=f[n-1,j], f[i,-1]=f[i,0], f[i,m]=f[i,m-1], f[-1,-1]=f[0,0], f[-1,m]=f[0,m-1], f[n,-1]=f[n-1,0], f[n,m]=f[n-1,m-1].

TEMA:

Considerand ca se da o matrice F(n,m) si o matrice de convolutie C(k,k) se cere sa se calculeze matricea V(n,m) rezultata in urma aplicarii convolutiei cu matricea de convolutie C pe matricea F.

A) Program secvential

B) 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 va folosi descompunere geometrica cu urmatoarele 2 variante:

- Pe orizontala (mai multe linii alocate unui thread)

si

- Pe verticala (mai multe coloane alocate unui thread)

Pentru urmatoarele variante se acorda suplimentar cate 2 puncte la nota finala.

1) Bloc – submatrici alocate unui thread (se face impartire si pe orizontala si pe verticala)
2) Distributie bazata pe o **functie de distributie delta** prin care unui thread cu index t i se atribuie o submultime de indecsi din matrice:

delta : NXM->P, N={0,1,2,...n-1}, M=0,1,2...m-1}, P={0,1,2...p-1}

delta(i,j)= t, 0<=i<n, 0<=j<m

testarea se va face definind functia de distributie astfel incat sa obtinem:

- distributie liniara (indici alaturati la acelasi thread) sau
- distributie ciclica(cu pas/step 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!!!

Date de iesire: output.txt fisier care contine matricea rezultat

Implementare:

a) Java

b) C++ (cel putin C++11)

   i. matricile sunt alocate static (int f[MAX][MAX])

   ii. 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
5) N=10000 M=10000 si n=m=5; p=2,4,8,16

Documentarea performantei

Pentru fiecare dintre variantele de testare rezultate din diferitele implementari si din diferitele date de test analizati performata folosind tabele similare celor evidentiate mai jos.

Aceste tabele trebuie adaugate in documentatie!

Java:

Tip matrice	**Nr threads**	**Timp executie**

N=M=10 n=m=3	secvential	
	4	…
N=M=1000 n=m=5	sequential	….
	1	….
	2	….
	4	….
	8	….
	16	….
N=10 M=10000 n=m=5		….

C++

Tip matrice	**Tip alocare**	**Nr threads**	**Timp executie**
N=M=10 n=m=3	Static	4	….
	dynamic	4	….
N=M=1000 n=m=5	static	1	….
		2	….
		4	….
		8	….
		16	….
	dynamic	1	….
		2	….
		4	….

		8	….
		16	….
….	….	….	….

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 (secventiala, paralela) folositi acelasi fisier “date.txt”;

Folositi recomandarile din fisierele “Testare” si “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.