* **Analiza cerintelor**

Tema:

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

-M linii ,N coloane

-Se considera un „kernel” de dimensiunei m x n , unde m si n sunt numere impare,iar m < M si n < N

-Se cere matricea rezultata in urma aplicarii kernel-ului peste matricea originala

- Pentru explicatii suplimentare a se vedea exemplul : <https://de.wikipedia.org/wiki/Datei:2D_Convolution_Animation.gif>

Constrangeri : Pentru frontiere se considera ca un element este egal cu elementul 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];

In cazul unui kernel cu m,n > 3 pentru frontierele care nu sunt imediat urmatoare frontierei matricii F se va aplica recursiv algoritmul de atribuire a valori vecinului celulei pana se ajunge la o valoare de pe frontiera imediat urmatoare matricii F.

* **Proiectare**
* **Structuri de date folosite:** lista cu valori de tip primitiv „double” dar si liste bidiminesionale ex. double[][]
* **P**entru varianta de executie pe thread-uri am folosit paritionarea pe nr\_threads intervale
* Clasele utilizate:

\ 1.DataStore

O clasa care ajuta la citirea matricilor de input(F si kernel) dintr-un fisier text „date.txt”

-ofera o functie „convertMatrixToList” care permite convertirea unei Matrici la o Lista

- Cand un obiect al aceste clase este instantiat se si citesc deja matricile si lungimile lor din fisierul „date.txt”

\ 2.Sequential

O clasa care are ca si campuri

-o lista rezultata prin convertirea matricii F la o lista(„listMatrix”),

- un „listKernel” care este reprezentarea liniara a matricii kernel citita din fisier in urma instantierii unui obiect DataStore

-un obiect de tipul DataStore

-Aceasta clasa are urmatoarele functii definite:

a.”getGuardedMatrixElement(int i,int j)” - returns a value that is always valid for the original F matrix taking into account the frontier constraints;

i and j are indexes within (not necessarly) a matrix;

b. mapListToMatrixCoordinates(int index,int \_m,int \_n)

maps a list index to it’s matrix elemetn corespondant coordinates

c.” getResultMatrixCorespondentElement(int index)”

in functie de indexul din reprezentarea liniara a unei matrici functia intoarce valoare trecuta prin filtrul kernel-ului care corespunde acestei pozitii si care trebuie trecuta in rezultat

d.” computeResultListMatrix(int start,int end,double[] result)”

-dandu-se o pozitie de start si una de sfarsit in reprezentarea liniara a matricii F

Se calculeaza pentru fiecare pozitie din acest interval [start,end] elementul corespondent care va aparea in rezultatul final dupa aplicarea” filtrului” matricii „kernel ului”

e.”convertListToMatrix”

-converteste lista de valori la matricea corespondenta

\3.Main

a.”printResult” – functie care matricea rezultata in urma calculelor intr-un fisier

b „ computeResultMatrixUsingThreads” – instantiaza p thread-uri,la care atribuie sarcini pe intervale in reprezentarea liniara a matricii originale F asupra careia se doreste transformarea cu kernel ul

c.main-primeste ca si parametrii ,args[0] = nr d threaduri pe care se doreste a rula

args[1] = fisierul de output in care se va afisa rezultatul

un flow complet al solutie prezentate ar fi:

Citire matrici (f si kernel) din fisierul „date.txt” -> convertirea matricilor la liste ->

Parcurgerea elementelor listei si calcularea corespondentului in lista rezultat (in cazul threads pe intervale se atrbuie cate un interval fiecarui thread ) -> convertirea listei rezultat la o matrice rezultat