Documentatie Tema 2

# Cerinte

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.

1. Program secvential
2. Program paralel: folositi **p** threaduri pentru calcul.
   1. Descompunere geometrica pe orizontala (mai multe linii alocate unui thread)

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

Fisierul trebuie creat anterior prin adaugarea de numere generate aleator.

Toate rularile trebuie executate cu acelasi fisier.

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

Postconditie: Matricea initiala contine imaginea filtrata.

Constrangere: NU se aloca o alta matrice rezultat si nici o matrice temporara!

Se pot folosi/aloca doar vectori temporari pentru care complexitatea spatiu se incadreaza in O(n).

Atentie si implementarea secventiala trebuie adaptata la constrangerea specificata.

Va fi nevoie sa se foloseasca vectori auxiliari temporari.

Obiectiv: optimizarea complexitatii-spatiu in conditiile obtinerii unei performante ridicate.

# Functii utilizate

## aplicareConvolutie(int[][] matrice, int[][] matriceConvolutie, int k, int linieElement, int coloanaElement)

* Aplică operația de convoluție pe un element specific din matricea matrice.
* matriceConvolutie este un filtru de convoluție (de dimensiune k x k) aplicat în jurul elementului aflat la poziția (linieElement, coloanaElement).
* Returnează valoarea calculată a elementului după aplicarea convoluției.

## bordareMatrice(int[][] matrice, int n, int m, int k)

* Extinde (bordează) matricea matrice astfel încât să permită aplicarea convoluției pe elementele din margini.
* Adaugă valori replicate din marginile matrice în jurul părților de nord, sud, vest și est.
* Aceasta permite aplicarea unui filtru de convoluție de dimensiune k x k fără a ieși din limitele matricei originale.

## parcurgereSecventialaInplace(int[][] matrice, int n, int m, int[][] matriceConvolutie, int k)

* Realizează o convoluție secvențială pe matricea matrice, modificând-o direct (in-place).
* Utilizează un vector temporar linieTemporara pentru a stoca rezultatele convoluției pentru fiecare linie înainte de a le copia în matrice.
* Aplică filtrul matriceConvolutie asupra fiecărui element.

## afisareMatrice(int[][] matrice, int n, int m)

* Afișează elementele matricei matrice cu n linii și m coloane în consolă.

## genereazaMatriceRandom(int n, int m, int k, String numeFisier)

* Generează o matrice aleatorie de dimensiuni n x m și o matrice de convoluție k x k cu valori aleatorii între 0 și 5999.
* Salvează matricea și matricea de convoluție într-un fișier specificat de numeFisier.

## parcurgereParalela(int[][] matrice, int n, int m, int[][] matriceConvolutie, int k, int nrThreaduri)

* Realizează operația de convoluție în mod paralel folosind mai multe fire de execuție.
* Creează o CyclicBarrier pentru sincronizarea firelor, astfel încât fiecare thread să aștepte finalizarea operației de copiere înainte de a aplica convoluția.
* Creează și rulează fire de tip PersonalThread pentru a parcurge și calcula rezultatele convoluției pe liniile/coloanele matricei.

## scrieInFisier(int[][] matrice, int n, int m, String numeFisier)

* Scrie matricea matrice în fișierul specificat de numeFisier, utilizând dimensiunile n și m.

## main(String[] args)

* Citește matricea principală și matricea de convoluție dintr-un fișier (implicit data.txt).
* În funcție de parametrii de intrare (args), execută convoluția fie în mod secvențial, fie în mod paralel.
* Măsoară timpul de execuție pentru fiecare metodă și scrie rezultatul într-un fișier de ieșire specificat de utilizator.

## Clasa PersonalThread

* Fiecare instanță de PersonalThread aplică convoluția pe secțiuni ale matricei în funcție de parametrii start și step, permițând divizarea sarcinilor între mai multe thread-uri.
* Copiază vecinii necesari pentru calculul convoluției într-un vector vecini înainte de a trece la calculul convoluției, utilizând CyclicBarrier pentru a sincroniza thread-urile.

# Java

|  |  |  |
| --- | --- | --- |
| Tip Matrice | Numar thread | Timp mediu de executie |
| N=M=10  K=3 | Secvential | 48 |
| 2 linie | 48.8 |
| N=M=1000  K=3 | Secvential | 50.6 |
| 2 linie | 52 |
| 4 linie | 52.9 |
| 8 linie | 52.7 |
| 16 linie | 51 |
| N=M=10000  K=3 | Secvential | 53.7 |
| 2 linie | 52.9 |
| 4 linie | 51.6 |
| 8 linie | 52.9 |
| 16 linie | 53.4 |

# C++

|  |  |  |
| --- | --- | --- |
| Tip Matrice | Numar thread | Timp mediu de executie |
| N=M=10  K=3 | Secvential | 51.1 |
| 2 linie | 51.5 |
| N=M=1000  K=3 | Secvential | 51.2 |
| 2 linie | 52.3 |
| 4 linie | 51.8 |
| 8 linie | 52.2 |
| 16 linie | 52.7 |
| N=M=10000  K=3 | Secvential | 53 |
| 2 linie | 51.4 |
| 4 linie | 52.8 |
| 8 linie | 52.7 |
| 16 linie | 51.1 |