**Laborator 5 - PPD**

De Miros Razvan-Andrei

Grupa 234, semigrupa 2

19.12.2022

**Analiza Cerintelor**

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Polinoamele sunt salvate in fisierere text, in directorul “data/test\_case1” sau “data/test\_case2”.

Coeficientii si exponentii au fost generati random in limitele impuse de problema. Coeficientii pot fi si negativi, expoentii nu pot fi 0 in fisier.

Rezultatul se afla in fisierul “result”.

Description: Text

Description automatically generated with medium confidence Description: A picture containing calendar

Description automatically generated

**Proiectare:**

Scriptul se ruleaza prin comanda “.\scriptJ.ps1 Main (nr\_threaduri) (nr\_readeri) (test\_case) (nr\_rulari)”.

Daca $nr\_threaduri == 0 se ruleaza varianta secventiala.

Description: Graphical user interface

Description automatically generated

**Varianta Secventiala:**

**Description: Text

Description automatically generated**

Se citeste cate o linie din fisier reprezentata de “(coefficient) (exponent)”, se creeza un nod de adaugat in lista inlantuita prin instantierea unui Monom.

Se adauga treptat in lista inlantuita sortata dupa exponent, In care nu putem avea coeficienti nuli.

Prin adaugarea unui nod exista 2 posibilitati :

1. Exponentul monomului din nod se regaseste in lista caz in care la coeficientul monomului din nodul existent se aduga coeficientul monomului din nodul de adaugat.

OBS!! Daca noul coefficient este 0, nodul se scoate din lista

Description: Text

Description automatically generated

1. Exponentul nou nu se regaseste in lista, deci se insereaza in pozitia corespunzatoare.

Description: Graphical user interface, text

Description automatically generated

**Varianta Paralela:**

**Text

Description automatically generated**

Functia primeste cazul de testare, numarul de threaduri, lista inaltuita rezultat si o coada de unde workerii extrag nodurile aduagate de reader.

Fata de laboratorul 4, introducem o bariera prin care sincronizam finalizarea readerilor.

**ReaderThread - urile** primesc cate un “chunk” de fisiere din care trebuie sa citeasca monoamele.

La final, cand termina de citit toate threadurile (eveniment semnalat de asteptarea la bariera), se apeleaza o functie “**setIsDone()**”din interfata cozii, care seteaza un flag, indicand ca procesul de citire a fost finalizat, astfel incat workerii sa fie notificati in cazul in care asteapta inca informatie.

Text

Description automatically generated

**WorkerThread** – urile extrag pe rand cate un Node din coada, atat timp cat **ReaderThread**- urile nu au terminat procesul de citire (flagul inca nu a fost setat pentru a indica terminarea).

Dar in cazul in care flagul este setat inainte ca coada sa se goleasca, WorkerThread-urile parcurg in continuarea coada si extrag pana cand aceasta se goleste.

Description: A screenshot of a computer

Description automatically generated with medium confidence

In interfata cozii, functiile sunt sincronizate si se foloseste sablonul producator-consumator(wait-notify)

Description: Text

Description automatically generated

Description: Text

Description automatically generated

Fata de laboratorul 4, implementarea adaugarii in LinkedList este diferita, intrucat sincronizarea se face la nivel de Node si nu pe intreaga lista. Fiecare Node are cate un lock asociat in interfata sa.

Text

Description automatically generated

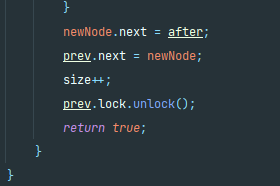
Parcurgem lista, tinand cont de un previous Node si un after Node, apeland functiile lock() si unlock() pe nodul previous pana cand ajungem pe pozitia corecta

Stergerea se face tot in fucntia de add(), prin desfacerea legaturii co nodul acolo unde coeficientul devine 0.

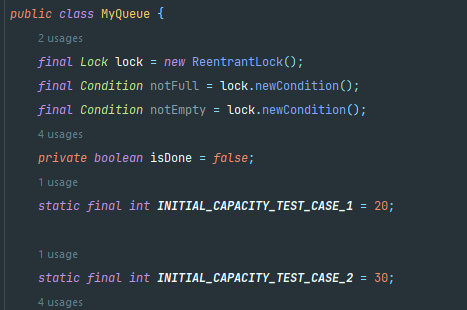
Text

Description automatically generated





Coada are o dimensiunea maxima data de cazul de testare.



Text

Description automatically generated

**Analiza Performanta:**

|  |  |  |
| --- | --- | --- |
| Caz testare | Nr Threaduri / nr Readeri | Timp executie(ms) |
| 1)  nrPolinoame : 10  gradMaxim : 1000  nrMaxMonoame: 50  (capacity) MAX = 20 | 0 (secvential) | 46.62023 |
| 4 / 2 | 45.70067 |
| 4 / 3 | 46.070255 |
| 6 / 2 | 45.93392 |
| 6 / 3 | 45.275055 |
| 8 / 2 | 45.268835 |
| 8 / 3 | 46.034535 |
| 2)  nrPolinoame : 5  gradMaxim : 10000  nrMaxMonoame: 100  (capacity) MAX = 30 | 0 (secvential) | 47.50678 |
| 4 / 2 | 47.58927 |
| 4 / 3 | 48.46259 |
| 6 / 2 | 48.43029 |
| 6 / 3 | 48.882535 |
| 8 / 2 | 48.31753 |
| 8 / 3 | 49.029425 |

**Analiza Performanta - Observatii:**

1. Se observa ca pentru primul caz (nr date mai mici) varianta secventiala are cea mai slaba performanta, iar pentru cel de-al doilea – cea mai buna. Dubios
2. In primul caz, flucteaza timpii de executie de la un numar de threaduri la altul cand avem 2 vs 3 readeri. Cel mai bun timp il avem atunci cand avem 6 threaduri cu 3 readeri sau 8 threaduri cu 2 readeri.
3. In schimb pentru al doilea caz de testate (nr date mari) variantele paralele sunt mai slabe decat varianta secventiala.
4. Intre cele doua in schimb (2 vs 3 readeri), varianta pe 2 readeri este in fiecare caz de testare mult mai eficienta decat cea pe 3 readeri.
5. Timpii au fost obtinuti in urma rularii de 20 de ori a fiecarui numar de threaduri si readeri, inclusiv secvential.

**Analiza Performanta Lab4 vs Lab5:**

**Analiza Performanta Lab4 vs Lab5 - Observatii:**

1. Se observa ca pentru primul caz (nr date mai mici) varianta implementata in Lab4 este mult mai eficienta in toate cazurile.
2. Aproximativ la fel si in cel de-al doilea caz de testare, cu exceptia variantei secventiale unde se observa ca modificarea adaugarii in lista inlantuita aduce cu sine imbunatatirea performantei.
3. Lab4 ramane cea mai eficienta implementare overall.