

Міністерство освіти і науки України
Національний технічний університет України «Київський
політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 4 з дисципліни
«Технології паралельних обчислень»

«Розробка паралельних програм з використанням пулів потоків,
екзекуторів та ForkJoinFramework»

Виконав(ла)

ІП-14 Сергієнко Ю. В.
(шифр, прізвище, ім'я, по батькові)

Перевірив

Дифучина О. Ю.
(прізвище, ім'я, по батькові)

Київ 2024

Комп'ютерний практикум 4

Тема: Розробка паралельних програм з використанням пулів потоків, екзекуторів та ForkJoinFramework.

Виконання:

1. Побудуйте алгоритм статистичного аналізу тексту та визначте характеристики випадкової величини «довжина слова в символах» з використанням ForkJoinFramework. **20 балів.** Дослідіть побудований алгоритм аналізу текстових документів на ефективність експериментально. **10 балів.**

Для виконання даного завдання було використано приклад реалізації пошуку слова у структурі файлів від Oracle. У ньому реалізований однопотоковий пошук (перебиранням файлів) та багатопотоковий пошук з використанням ForkJoinFramework. Основним готовим функціоналом є конвертація тексту усіх файлів у слова з видаленням пунктуаційних символів.

Для знаходження середньої довжини слова було додано клас WordLength, у якому описані avg та count параметри. З кожною ітерацією слів будемо розраховувати avg параметр. Для проведення коректного експерименту було створено структуру файлів із книг у текстовому форматі (рисунок 1).

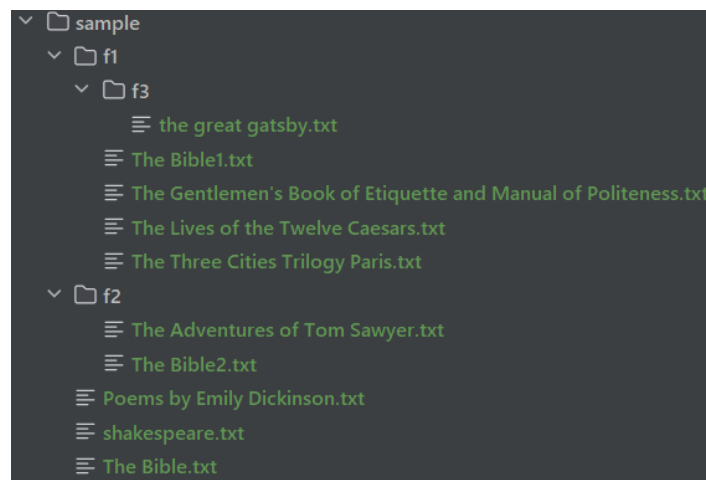


Рисунок 1 – Структура файлів

Результати замірів часу роботи алгоритмів (1 замір – по 10 ітерацій) можна бачити у таблиці 1.

Таблиця 1 – Результати експерименту

№	Single, ms	Parallel, ms	Acc, t
1	1188	459	2.42
2	1160	350	3.31
3	1137	271	4.2

2. Реалізуйте один з алгоритмів комп'ютерного практикуму 2 або 3 з використанням ForkJoinFramework та визначте прискорення, яке отримане за рахунок використання ForkJoinFramework. **20 балів.**

Було створено реалізацію алгоритму множення матриць Фокса за допомогою ForkJoin, адже тут можливо розбити на підзадачі основну дію – розбиття матриці на блоки. У рекурсивній задачі MultiplyMatrixTask будуть запускатись $q * q$ підзадач (де q – к-сть блоків діагоналі), поки к-сть елементів блоку m більша за поріг або поточний блок ще можливо поділити. Коли алгоритм доходить порогу, відбувається множення поточних блоків та повернення даного результату.

Таблиця 2 – Результати роботи алгоритму Фокса

Size	Threads	Fox, ms	FJ Fox		Acc, t
			Threshold	Time, ms	
1000	4	324	125	294	1.1
	16	224	250	1550	0.15
2048	4	4674	128	2059	2.27
			32	2084	2.24
	16	2556	8	3195	0.8
			32	1716	1.49
3000	4	21256	375	13268	1.6
	16	12814	750	198448	0.064

У результаті спостережень виявлено, що алгоритм множення з використанням ForkJoinFramework справляється із задачею краще при більшій кількості поділів, але не при найменшій. При мінімальній кількості поділів (при вхідних матрицях, розмір яких неподільний на кількість потоків) більш оптимальним є звичайний алгоритм.

3. Розробіть та реалізуйте алгоритм пошуку спільних слів в текстових документах з використанням ForkJoinFramework. **20 балів.**

Для виконання даної задачі було взято рішення Oracle для пошуку слів. Щоб знайти спільні слова вхідних документів, необхідно пройти кожний з них, записати усі слова та кількість їх повторів (найкращий варіант – структура Map). Далі, після ітерації, з'єднаємо поточну структуру та знайдену. В результаті отримаємо список слів з к-стями їх повторів (рисунок 2).

```
Drives=2, throne=718, throng=54, Ear's=1, beguile=9, sorceress=6, Casum=1, Ebooks=1, proprietary=11, unwearily=2, andirons=1, Grew=7, Jessica=32, Grey=27, fluorine=1, Eager=1, Dream=1, MARK=4, Travel=1, booths=1, russet=3, advertise=5, claying=1, humility=132, potent=20, rivage=1, "fight=1, polykoiraniae=1, "behold=1, inclosure=1, deboshed=1, Cassi=1, Dainties=1, insolvable=1, engirts=1, rightty=65, MATE=1, horseman's=1, consummated=11, naves=5, aquiline=2, answerable=9, CHILD=3, queen's=20, cygnet's=1, navel=14, "girdling=1, Annihilate=1, chirruping=1, flag=22, blabbed=1, insignificant=5, perceptible=11, botany=1, Mephostophilus=1, MARTIAL=1, there't=1, praetor's=1, there's=247, derive=28, Grins=3, flays=5, dimidiates=1, flax=42, archbishopric=1, flaw=7, Begins=5, flat=96, Afforded=1, skinless=1, zephyrs=1, flap=9, there'd=3, brazen=90, sickening=2, occupying=4, thudding=1, twisted=70, enamel=1, suggest=20, averted=4, drummer=2, Play=20, inexorable=6, litigants=2, plotters=1, Camerinos=1, Assur=24, Glick=1, responses=1, poniards=6, perditit=1, argument=120, rud'et=2, hopeful=15, Equality=1, Member=1, tenure=3, Inherit=1, effusion=8, cheerly=9, Jerusa=6, engender'd=3, deliveries=1, broach=4, herself=514, favorites=2, shouting-before=1, VALERIUS=8, instances=38, Locusta=4, flae=377, Asson=3, flad=768, Assos=6, flae=11, Porpentine=5, Locusts=6, instanced=3, taxings=2, flae=55, limitless=2, man-well=1, Severus=9, conceit's=1, CHIEF=57, response=10, responsive=2, LABOUR=1, rival=49, throws=5)
process finished with exit code 0
```

Рисунок 2 – Результат пошуку спільних слів

4. Розробіть та реалізуйте алгоритм пошуку текстових документів, які відповідають заданим ключовим словам (належать до області «Інформаційні технології»), з використанням ForkJoinFramework. **30 балів.**

Щоб виконати цю задачу необхідно модифікувати початковий код, а саме додати перевірку під час ітерації слів документу, чи воно співпадає зі словами області ІТ. Якщо так, будемо додавати назву поточного документу в результуючий список (рисунок 3).

```
the great gatsby.txt
The Bible1.txt
The Gentlemen's Book of Etiquette and Manual of Politeness.txt
The Lives of the Twelve Caesars.txt
The Three Cities Trilogy Paris.txt
The Adventures of Tom Sawyer.txt
The Bible2.txt
Poems by Emily Dickinson.txt
shakespeare.txt
The Bible.txt

Process finished with exit code 0
```

Рисунок 3 – Результируючі документи

Висновок

Під час виконання даного комп'ютерного практикуму я закріпив знання та навички щодо розробки паралельних алгоритмів з використанням ForkJoinFramework. Даний фреймворк дає змогу оптимізувати будь-який процес чи алгоритм, який можна перебудувати рекурсивно. На прикладі даних задач бачимо, що шукати по файлах можна значно швидше за допомогою пулу потоків та рекурсії. Алгоритм Фокса з використанням ForkJoin є більш оптимальним на об'ємах даних, розміри яких є подільними, тобто є змога зробити більше розгалудження задач для зменшення навантаження системи.

Код програми доступний на [Github](#).