УДК 004.8

В.О. Максименюк
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ ODD-EVEN SORT

Вінницький національний технічний університет

Анотація

Розглянуто розробку та оптимізацію паралельного алгоритму сортування Odd–Even Sort із використанням технології OpenMP. Проведено аналіз існуючих підходів до паралельного упорядкування даних з метою підвищення продуктивності та обґрунтовано вибір засобів програмної реалізації, розроблено діаграми компонентів і діяльності програмного модуля. У роботі створено програмну реалізацію паралельного алгоритму Odd–Even Sort з використанням OpenMP та проведено тестування її ефективності на різних наборах даних і масивах різного розміру. Використання результатів дозволить покращити швидкодію і продуктивність програм та алгоритмів, які потребують обробки великих масивів даних.

Ключові слова: *Odd–Even Sort, паралельне сортування, OpenMP, багатопотоковість.*

Abstract

The development and optimization of the parallel sorting algorithm Odd–Even Sort using OpenMP technology is considered. The analysis of existing approaches to parallel data ordering to increase productivity is carried out and the choice of software implementation tools is justified, diagrams of components and activities of the software module are developed. In the work, a software implementation of the parallel algorithm Odd–Even Sort using OpenMP is created and its effectiveness is tested on various data sets and arrays of different sizes. Using the results will allow improving the speed and productivity of programs and algorithms that require processing large data arrays.

Keywords: *Odd–Even Sort, parallel sorting, OpenMP, multithreading.*

Вступ

Актуальність дослідження паралельного алгоритму сортування Odd–Even Sort визначається зростанням обсягів даних, що обробляються сучасними обчислювальними системами. У багатьох прикладних сферах – від наукового моделювання та аналізу великих даних до інженерних та бізнес-застосувань – продуктивність базових алгоритмів сортування суттєво впливає на ефективність програмних комплексів. Оскільки послідовні методи не завжди забезпечують необхідну швидкодію, зростає потреба у паралельних підходах, які дозволяють скоротити час опрацювання великих масивів даних.

Використання технології OpenMP дає змогу ефективно розподіляти обчислювальне навантаження між потоками, забезпечуючи прискорення роботи алгоритму при мінімальних змінах його базової структури.

Метою роботи є розробка паралельного алгоритму Odd–Even Sort із використанням OpenMP для підвищення продуктивності обчислювальних процесів під час сортування великих наборів даних.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- аналіз структури та ключових особливостей алгоритму Odd–Even Sort;
- побудова модельного представлення та діаграм роботи алгоритму;
- реалізація послідовної та паралельної версій алгоритму з використанням OpenMP;
- проведення тестування продуктивності на різних наборах даних;
- порівняння результатів та оцінка ефективності паралельної обробки.

Виклад основного матеріалу

Алгоритми сортування відіграють ключову роль у великій кількості прикладних задач, включаючи аналіз даних, моделювання, інформаційний пошук та оптимізаційні процеси. Їхня продуктивність суттєво впливає на ефективність сучасних програмних систем, особливо при роботі з великими масивами даних [1]. Одним із підходів до прискорення сортування є використання паралельних обчислень, які дозволяють розподілити роботу між кількома потоками й скоротити загальний час обробки. У цьому контексті особливу увагу становлять алгоритми, внутрішня структура яких природно підходить для розпаралелювання. До таких належить Odd–Even Sort – порівняльний алгоритм із чіткою фазовою організацією, який є модифікацією методу бульбашкового сортування.

Вхідними даними для алгоритму Odd-Even Sort є одновимірний масив елементів

$$A = [a_0, a_1, \dots, a_{n-1}],$$

де ключем порівняння виступає певний тип даних (найчастіше – цілі або дійсні числа) і розмір масиву n. Результатом роботи алгоритму є масив A, відсортований за неспаданням або спаданням, залежно від заданої умови.

Алгоритм Odd–Even Sort розділяє проходи на непарні фази (Odd) та парні фази (Even):

1. Odd-фаза – працює з парами елементів, які мають непарний індекс: (A[1], A[2]), (A[3], A[4]), (A[5], A[6]) … ;
2. Even-фаза – працює з парами елементів, які мають парний індекс: (A[0], A[1]), (A[2], A[3]), (A[4], A[5]) … .

Паралельність досягається за рахунок того, що жодна пара не перетинається за індексами, тобто не має спільних елементів [2].

Цей алгоритм є стійким, оскільки не виконує перестановок між рівними елементами (за умови використання порівняння A[j] > A[j+1]).

Алгоритм на кожну з n фаз виконує (n – 1)/2 порівнянь сусідніх елементів, загальна кількість виконаних порівнянь становить: $(n^2 - n)/2$.

У найгіршому випадку, масив упорядкований у зворотньому порядку або таким чином, що майже кожне порівняння призводить до обміну, загальна кількість операцій рівна $(n^2 - n)/2$, що зростає квадратично. Таким чином, алгоритм Odd–Even Sort має часову складність $O(n^2)$.

Середнім випадком вважається ситуація, коли елементи масиву розташовані без певного порядку, хаотично. Однак характер вхідних даних не впливає на те, скільки фаз виконає алгоритм: Odd–Even Sort проходить всі n фаз і в кожній виконує (n – 1)/2 порівнянь. Тому середній випадок, так само як і найгірший, приводить до квадратичної кількості операцій. Отже, середня оцінка часу виконання алгоритму дорівнює $O(n^2)$ [4].

Найсприятливішим є випадок, коли масив уже відсортований. Оскільки, алгоритм проводить порівняння незалежно від вхідних даних, часова складність також рівна $O(n^2)$.

Odd–Even Sort працює in-place, тобто виконує впорядкування без створення додаткових структур даних. Єдиним допоміжним елементом є тимчасова змінна, що використовується під час обміну двох сусідніх елементів, а також лічильники циклів. Кількість додаткової пам'яті не залежить від розміру масиву і залишається сталою: для вхідного масиву розміром n – O(n), для додаткової змінної – O(1) [4].

Сортування Odd–Even Sort має характерні переваги, що визначають його застосовність у задачах паралельного впорядкування. Він є простим у реалізації, оскільки базується на елементарних операціях порівняння та обміну сусідніх елементів. Завдяки чіткому поділу роботи на непарні та парні фази алгоритм демонструє схильність до паралельного виконання: усі операції в межах однієї фази є незалежними та можуть бути розподілені між потоками або обчислювальними ядрами без додаткових засобів синхронізації. Це дозволяє скорочувати час однієї ітерації та отримувати відчутний виграш на багатоядерних системах. Структурна простота робить Odd–Even Sort привабливим для апаратної реалізації та дослідницьких завдань, пов'язаних з вивченням основ паралельних моделей обчислень [3].

Проте алгоритм має і недоліки, що обмежують його практичне використання. Квадратична обчислювальна складність робить Odd–Even Sort малоефективним для великих масивів і

поступається більш сучасним методам, таким як швидке сортування чи сортування злиттям. Хоча паралелізація скорочує час обробки, вона не змінює асимптотичної складності й потребує синхронізації між фазами, що може спричиняти накладні витрати в реальних паралельних системах. Додатковим обмеженням є те, що алгоритм виконує обміни виключно між сусідніми елементами, через що упорядкованість поширюється масивом поступово, а кількість необхідних ітерацій може бути значною, особливо для слабко впорядкованих або хаотичних даних. У сукупності ці особливості роблять Odd–Even Sort корисним насамперед у навчальних, дослідницьких або вузькоспеціалізованих задачах, де ключову роль відіграє саме можливість зручної та прозорої паралельної реалізації [3].

Паралельна модифікація алгоритму знижує фактичний час виконання за рахунок одночасного опрацювання незалежних порівнянь. За використання технології OpenMP блоки пар елементів можуть бути розподілені між потоками за допомогою директиви parallel for. Хоча асимптотична складність формально залишається $O(n^2)$, прискорення досягається за рахунок зменшення часу обробки однієї фази: замість послідовного проходження всіх пар елементів, вони порівнюються паралельно у доступних потоках. Теоретичний виграш обмежується максимальною кількістю непересічних пар у масиві, тобто приблизно n/2, однак реальне прискорення визначається числом доступних апаратних потоків і накладними витратами синхронізації [5-6].

Програмна реалізація паралельного алгоритму сортування Odd–Even Sort створена на мові програмування C++, яка є ефективним інструментом для створення високопродуктивних обчислювальних програм, наприклад таких, що реалізують алгоритми сортування та багатопотокові обчислення [7-8]. Програмно реалізовано задану задачу та описано всі компоненти коду.

Аналіз результатів тестування програми виконано для різної кількості елементів масиву (табл.1-2).

Таблиця 1 – Порівняння часу сортування алгоритмом Odd–Even Sort з використанням потоків та без них.

Розмір, n	Час сортування з потоками, с	Час сортування без потоків, с
1 000	0,01093	0,01128
10 000	1,11421	1,13974
50 000	22,25730	24,17340
100 000	85,27920	86,48880

Таблиця 2 – Порівняння часу сортування алгоритмом Odd–Even Sort з використанням потоків та без них для найкращого, середнього та найгіршого випадків.

Розмір n	Час сортування	1 000	10 000	50 000	100 000
Найкращий випадок	Без потоків	0,0000123	0,000127	0,00064	0,00130
	З потоками	0,0000121	0,000125	0,00079	0,00127
Середній випадок	Без потоків	0,0088100	1,165770	24,05170	99,84100
	З потоками	0,0079300	1,147650	22,02560	91,656100
Найгірший випадок	Без потоків	0,0115900	1,591030	30,54310	121,64300
	З потоками	0,0110300	1,537170	29,65810	119,53700

Проведені експериментальні дослідження демонструють, що паралельна реалізація Odd–Even Sort помітно зменшує загальний час виконання для великих масивів. Найбільший приріст швидкодії спостерігається при роботі з випадково згенерованими даними та при збільшенні розміру масивів, для яких кількість внутрішніх ітерацій стає значною. Результати підтверджують, що алгоритм є ефективним навчальним прикладом розпаралелювання порівняльних методів сортування та може служити основою для дослідження впливу багатопотоковості на продуктивність алгоритмів обробки даних.

Висновки

Досліджено алгоритм Odd–Even Sort як приклад порівняльного методу сортування, визначено його місце серед класичних алгоритмів та обґрунтовано доцільність паралельної реалізації. Розглянуто принцип роботи алгоритму на основі чергування непарних і парних фаз, визначено його часову складність та особливості розпаралелювання. На основі цих досліджень побудовано діаграми, що відображають архітектуру програмного модуля, та потоковий граф, який ілюструє послідовність операцій алгоритму.

Реалізовано послідовну та паралельну версії алгоритму засобами C++ та OpenMP. Проведено тестування продуктивності на масивах різного типу та розміру. Отримані результати підтверджують коректність роботи алгоритму та демонструють суттєве скорочення часу виконання паралельної версії порівняно з послідовною, особливо при збільшенні обсягу даних. Встановлено, що ефективність прискорення залежить від кількості потоків і характеру вхідних даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Методичні вказівки до лабораторної роботи № 13. Київський політехнічний інститут. URL: https://ee.kpi.ua/~golubeva/PC_Lab/LR_PC_13.pdf
2. Кузьменко І. М., Дацюк О. А. Базові алгоритми та структури даних: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського, 2022. 137 с.
3. Звіт “Алгоритми сортування: парне-непарне, за розрядами”. URL: https://surl.lt/umwzxh
4. Odd-Even Transposition Sort. URL: https://www.baeldung.com/cs/odd-even-transposition-sort
5. Яковлева І. Д., Лісовенко І. Д. Підхід до побудови потокового графа алгоритму. Чернівці: ЧНУ імені Юрія Федьковича, 2005. 4 с.
6. Інструкція, як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575/
7. Введення в C++. URL: https://metanit.com/cpp/tutorial/1.1.php
8. Введення в OpenMP. URL: https://metanit.com/c/tutorial/11.6.php

Максименюк Вікторія Олександрівна – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м. Вінниця, e-mail: v.maksimenjuk@gmail.com;

Денисюк Валерій Олександрович – канд. техн. наук, доцент, доцент кафедри комп’ютерних наук, Вінницький національний технічний університет, м. Вінниця, e-mail: vad64@i.ua.

Maksymeniuk Viktoria Olexandrivna – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: v.maksimenjuk@gmail.com;

Denysiuk Valerii Olexandrovich – Ph.D., Assistant Professor, Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: vad64@i.ua