УДК 004.8

А.Ю. Сухоцька
В.О. Денисюк

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ BINGO SORT

Вінницький національний технічний університет

Анотація

Розглянуто алгоритм Bingo Sort, досліджено його послідовну та паралельну реалізацію, обґрунтовано засоби розробки програмного модуля на Python з використанням бібліотеки Tkinter для візуалізації. Розроблено діаграми класів програмного модуля, обґрунтовано вибір програмного засобу реалізації та створено програмну реалізацію алгоритму з можливістю візуалізації та порівняння часу виконання. Проведено тестування модулю на різних типах масивів. Реалізація програмного модуля дозволяє підвищити наочність алгоритму, оцінити переваги паралельного сортування та зручність його використання в навчальних та практичних завданнях.

Ключові слова: *Bingo Sort, паралельне сортування, Python, візуалізація алгоритму, порівняння часу виконання.*

Abstract

The work examines the Bingo Sort algorithm, investigates its sequential and parallel implementations, and justifies the tools used for developing the software module in Python with the Tkinter library for visualization. Class diagrams of the software module were developed, the choice of the programming tool was substantiated, and a software implementation of the algorithm was created with the capability of visualization and time-performance comparison. The module was tested on different types of arrays. The implementation of the software module allows enhancing the algorithm's clarity, evaluating the advantages of parallel sorting, and facilitating its use in educational and practical tasks.

Keywords: *Bingo Sort, parallel sorting, Python, algorithm visualization, performance comparison.*

Вступ

Актуальність розробки паралельного алгоритму Bingo Sort зумовлена зростанням обсягів даних, які потребують швидкої та ефективної обробки в сучасних обчислювальних системах. У багатьох прикладних задачах — від аналізу даних до моделювання та комп'ютерної графіки — сортування є базовою операцією, що суттєво впливає на продуктивність програм [1].

Алгоритм Bingo Sort належить до вибіркових методів сортування та базується на послідовному пошуку максимального елемента і перенесенні його у відсортовану частину масиву. Послідовне виконання таких операцій обмежує швидкодію при роботі з великими наборами даних. Паралельна обробка дає можливість одночасно виконувати пошук у декількох сегментах масиву, що скорочує загальний час виконання та робить алгоритм ефективнішим для багатоядерних систем.

Метою роботи є створення та дослідження паралельної реалізації Bingo Sort для підвищення швидкодії сортування великих масивів даних. У роботі розглянуто принципи роботи алгоритму, особливості паралельного підходу, програмну реалізацію та результати експериментального тестування.

Постановка задачі дослідження

Задачі дослідження передбачають вирішення таких питань:

- дослідження принципів роботи алгоритму Bingo Sort;
- створення послідовної та паралельної реалізацій алгоритму;
- тестування продуктивності на різних наборах даних;
- порівняння результатів та оцінка прискорення паралельного підходу.

Виклад основного матеріалу

Алгоритми сортування відіграють ключову роль у сучасних програмних системах, забезпечуючи впорядкування великих масивів даних, що необхідно для пошуку, фільтрації, класифікації та аналітичної обробки інформації [2, 5]. Підвищення швидкодії таких операцій набуває особливої важливості в умовах великих обсягів даних та багатоядерних обчислювальних систем, де традиційні послідовні алгоритми виявляються недостатньо ефективними [3]. Одним із таких алгоритмів є Bingo Sort — модифікація методу вибірки, що працює шляхом визначення максимального або мінімального елемента (bingo-значення) та переміщення всіх його копій на відповідні позиції масиву[1].

У класичному варіанті Bingo Sort виконується послідовно: кожне нове bingo-значення визначається шляхом проходу по всьому масиву, після чого елементи з цим значенням переміщуються на початок або кінець масиву. Такий підхід простий, але має часову складність $O(n^2)$, що обмежує ефективність при роботі з великими наборами даних [1]. Особливо це помітно у випадках, коли кількість унікальних значень масиву наближається до його розміру.

Паралельна реалізація Bingo Sort усуває цей недолік завдяки можливості виконувати незалежні операції одночасно. Масив даних розподіляється між потоками, кожен з яких виконує пошук локальних екстремумів у своєму підмасиві [3, 4]. Після завершення локальних обчислень виконується редукція — глобальне визначення bingo-значення за результатами потоків. Наступним етапом усі потоки паралельно переміщують елементи, рівні знайденому значенню, у відповідну позицію масиву. Після завершення переміщення оновлюються межі невідсортованої частини, і процес повторюється.

Завдяки такій структурі алгоритму одночасно прискорюються три ключові етапи:

- пошук локальних мінімумів / максимумів;
- переміщення рівних елементів;
- оновлення робочих меж масиву [3].

Ефективність паралельного Bingo Sort особливо помітна тоді, коли масив містить багато однакових значень, оскільки алгоритм за одну ітерацію переставляє цілу групу елементів [1]. Це зменшує кількість проходів та дозволяє краще розподілити навантаження між потоками.

Під час реалізації алгоритму важливу роль відіграють механізми синхронізації — бар'єри, м'ютекси або семафори. Вони забезпечують узгодженість виконання потоків, не допускають конфліктів доступу до спільної пам'яті та гарантують правильність результату [4]. Однак надмірна кількість синхронізацій може знизити продуктивність, тому оптимальне балансування потоків та підмасивів є критично важливим аспектом.

З математичної точки зору час виконання паралельного алгоритму Bingo Sort можна подати як суму часу розподілу даних, обчислення та синхронізації. Прискорення визначається співвідношенням часу послідовного й паралельного виконання, а ефективність — часткою корисного обчислювального часу кожного потоку [4]. Практичні дослідження показують, що при великій кількості елементів і достатній кількості повторів Bingo Sort здатний масштабуватися майже лінійно до певного рівня, після якого ефективність зменшується через накладні витрати на синхронізацію [3, 5].

Паралельний Bingo Sort має низку переваг: можливість значного прискорення сортування великих масивів, ефективне використання багатоядерних систем, здатність одночасно обробляти групи однакових значень. Проте існують і недоліки, пов'язані з необхідністю синхронізації потоків, можливими конфліктами доступу до пам'яті та складністю балансування навантаження [4]. Порівняльний аналіз свідчить, що оптимальний ефект досягається для масивів великого розміру з достатньою кількістю повторів значень [2].

Таким чином, паралельний Bingo Sort є перспективним інструментом для сортування великих наборів даних у багатоядерних архітектурах. Він поєднує простоту базового алгоритму з можливістю значного скорочення часу виконання за рахунок одночасної роботи кількох потоків, що робить його ефективним у багатьох практичних сценаріях.

Програмно реалізовано задачу та описано всі компоненти коду [6].

Аналіз результатів тестування програми виконано для різної кількості елементів масиву та різного типу масиву (табл.1-3).

Таблиця 1 – Час виконання програми на різних вхідних даних

Тип масиву	Загальна розмірність матриці					
	1 000		5 000		10 000	
	паралельне	послідовне	паралельне	послідовне	паралельне	послідовне
Random	0,020537	0,019697	0,487061	0,503636	1,975035	2,028673
Reverse	0,022061	0,020017	0,501647	0,507668	2,037168	2,086421
Sorted	0,022037	0,019025	0,490376	0,504408	2,013514	2,069115

Таблиця 2 – Коефіцієнти прискорення

Тип масиву	Загальна розмірність матриці		
	1 000	5 000	10 000
Random	0.9589	1.0340	1.0271
Reverse	0.9078	1.0120	1.0242
Sorted	0.8634	1.0287	1.0276

Таблиця 3 – Коефіцієнти ефективності (P = 4)

Тип масиву	Загальна розмірність матриці		
	1000	5000	10000
Random	0.2397	0.2585	0.2568
Reverse	0.2269	0.2530	0.2560
Sorted	0.2158	0.2572	0.2569

Проаналізувавши отримані часи виконання, коефіцієнти прискорення та коефіцієнти ефективності для різних типів масивів (Random, Reverse, Sorted) для різних розмірів матриці (1000, 5000, 10000) (табл. 1–3), можна зробити висновки.

1. Для невеликих масивів (розмір 1000) значення коефіцієнта прискорення близькі до 1 або менші, а коефіцієнт ефективності дуже низький (E = 0,2158–0,2397). Це означає, що кількість потоків при виконанні програми з невеликими вхідними даними фактично не впливає на швидкість виконання.

2. Для більших масивів (5000–10000) збільшення кількості потоків призводить до незначного зростання коефіцієнта прискорення (S ≈ 1,012–1,034), проте значне зниження ефективності (E ≈ 0,253–0,258), оскільки накладні витрати на створення потоків перевищують виграш від паралельності.

3. Найменший час виконання спостерігається для відсортованого масиву (Sorted), що пояснюється швидким визначенням максимумів алгоритмом Bingo Sort.

4. Найбільший час виконання фіксується для Reverse-масиву, що відповідає теоретично найгіршому випадку для алгоритму.

5. Структура вхідних даних має прямий вплив на час виконання та коефіцієнти ефективності: Random і Sorted масиви демонструють приблизно однакові результати, Reverse-масив показує найбільш тривалий час обчислень.

Отже, для даного алгоритму паралельна обробка не дає значного прискорення, а ефективність знижується через накладні витрати на створення потоків та обмеження GIL у Python.

Висновки

Досліджено алгоритм Bingo Sort, як приклад сортування масивів, визначено особливості його роботи на різних типах вхідних даних (Random, Reverse, Sorted). Розглянуто паралельну та

послідовну реалізації алгоритму, описано їхню логіку та компоненти коду. Програмно реалізовано задачу сортування, проведено тестування та UML-діаграму для структури програми.

Проаналізовано час виконання, коефіцієнти прискорення та ефективності. Встановлено, що збільшення кількості потоків призводить до незначного підвищення коефіцієнта прискорення та суттєвого зниження коефіцієнта ефективності через накладні витрати та обмеження паралельності в Python.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Шинкаренко В. І., Макаров О. В. Комбіновані алгоритми сортування. Наукові праці НМетАУ, 2018. Електронний ресурс. URL: https://journals.nmetau.edu.ua/index.php/itmm/article/download/1640/941
2. Макаров О. В., Шинкаренко В. І. Конструювання алгоритмів сортування. Освітній портал USTU, 2016. URL: https://crust.ust.edu.ua/items/15a69fe9-0cec-46e9-9f82-7d765ec04035
3. Singh R., Sharma P. Parallel Sorting Algorithms: Performance Evaluation and Comparative Analysis. IEEE Xplore, 2022.
4. Rajasekaran S., Reif J. (Eds.) Handbook of Parallel Computing: Models, Algorithms and Applications. Boca Raton: CRC Press, 2010. 800 c.
5. Akl S. G. Parallel Sorting Algorithms. San Diego: Academic Press, 1993. 300 c.
6. Tkinter — Python Interface to Tcl/Tk. Python.org. URL: https://docs.python.org/3/library/tkinter.html

Сухоцька Аліна Юріївна – студентка кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: sukhotskaalina@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.

Sukhotska Alina Yurievna – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: sukhotskaalina@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