DOI: 10.34185/1991-7848.itmm.2025.01.045

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЧАСОВОЇ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ПОШУКУ ПІДРЯДКА

Клименко І.В.[1], Лебідь Є.А.[2]

[1]УДУНТ, к.е.н., доцент, Україна

[2]УДУНТ, аспірант, Україна

Анотація. *У статті проведено порівняльний аналіз часової ефективності чотирьох класичних алгоритмів пошуку підрядка (string searching algorithms), що намагаються знайти позицію, де один або декілька текстових рядків (зразків) входять у довший рядок або текст. Порівнювались наступні алгоритми: наївного пошуку (перебору, або ж брут-форс), Кнута-Морріса-Пратта (КМП), Боєра-Мура та Рабіна-Карпа. Дослідження проводилось на трьох різних архітектурах процесорів та з трьома наборами вхідних даних різного обсягу. Алгоритми тестувались за умов як холодного, так і прогрітого кешу. Для зниження впливу сторонніх чинників було реалізовано запуск на одному процесорному ядрі та примусове очищення пам'яті після кожного тесту. Результати експериментів проаналізовано за допомогою розрахунків S- та R-показників ефективності, надано рекомендації щодо доцільності застосування кожного окремого алгоритму в різних умовах.*

Ключові слова: *інформаційна технологія, машинний експеримент, S- та R-показники, алгоритм пошуку підрядка, часова ефективність алгоритму*

Пошук підрядка в рядку є однією з базових задач інформатики та комп'ютерних наук, з широким спектром застосування – від пошукових систем, обробки природної мови та аналізу великих даних до компіляторів і текстових редакторів. Попри однакову функціональність, існуючі алгоритми пошуку суттєво відрізняються за структурою, стратегією пошуку та ефективністю в різних умовах. Наявність великої кількості алгоритмів пошуку та їх різноманітність виводить задачу вибору оптимального алгоритму для кожного окремого випадку на новий рівень – визначення критеріїв, оцінок та метрик, які дають змогу враховувати теоретичні оцінки складності [2], емпіричні характеристики у реальному середовищі виконання і практично оцінити його якість/ефективність для заданих вхідних даних.

Метою дослідження є експериментальне порівняння часової ефективності функціонально еквівалентних алгоритмів пошуку підрядка: наївного пошуку

(брут-форс), Кнута-Морріса-Пратта [3], Боєра-Мура [4] та Рабіна-Карпа [5]. Основне завдання – оцінити вплив програмно-апаратних засобів, обсягу вхідних даних та стану кешу на продуктивність алгоритмів і визначити найбільш доцільні умови їх застосування.

Усі алгоритми реалізовані мовою Go із забезпеченням однакових умов запуску. Тестування виконувалось на трьох різних апаратних платформах: Apple M1 (MacBook Pro M1 2020 - ARM, 8 vCPU (4+4), 16 GB RAM), GCP VM c2-standard-4 (Intel Cascade Lake, 4 vCPU, 16 GB RAM), GCP VM c2d-highcpu-4 (AMD Milan, 4 vCPU, 8 GB RAM).

Для зменшення впливу багатозадачності та отримання більш достовірних результатів експериментів на всіх архітектурах примусово обмежувалось виконання операцій тільки на одному ядрі (за допомогою інструкції GOMAXPROCS(1)); додатково перед кожною ітерацією експерименту проводилось очищення пам’яті (використовуючи runtime.GC()). Оскільки часова ефективність алгоритмів залежить від об’єму вхідних даних [1], для отримання об’єктивних результатів експерименти проводились над трьома різними наборами текстових даних – маленькому (500 – 1000 символів), середньому (приблизно 10000 символів) та великому (більше 100000 символів). Щоб оцінити вплив кешування на час виконання алгоритмів окремо було проведено серію виконань для різних режимів кешу: так званого «прогрітого» (100 попередніх виконань алгоритму результати яких не замірюються) та холодного (без попередніх виконань).

Для кожної конфігурації (архітектури/об’єму даних/режиму кешу) виконувалось 100, 500 та 1000 виконань алгоритмів, оскільки виконання серії випробувань підвищує статистичну точність результату. Після збору результатів обчислювались середні значення часу виконання, медіана та довірчий інтервал, а також розраховувались S- і R-показники ефективності відповідно до запропонованої та обґрунтованої методики [1].

Розрахунок показника S – середньої переваги одного алгоритму над іншим, базується на методі попарного порівняння часових витрат та виконується наступним чином:

$$S = \frac{1}{N}\sum_{i=1}^{N}\frac{t_i'' - t_i'}{max(t_i', t_i'')} \cdot 100\%, \quad (1)$$

де t_i'', t_i' – час виконання для кожного з алгоритмів у і-му випробуванні, N – кількість випробувань. Для кожної пари алгоритмів обчислюється, скільки разів один із них виконався швидше за інший, та виводиться відсоткове співвідношення (S-показник).

Показник R – визначає відносну частку області, де спостерігається перевага одного з алгоритмів над іншим та може бути представлений як:

$$R = \frac{1}{N}\sum_{i=1}^{N} sign(t_i'' - t_i') \cdot 100\%, \; sign(a) = \begin{cases} 1, \text{якщо } a > 0 \\ 0, \text{якщо } a \leq 0 \end{cases} \quad (2)$$

На рис. 1 графічно зображено час виконання алгоритмів пошуку підрядка в рядку (Бойера-Мура, перебором, КМП, Рабіна-Карпа) у середовищі GCP VM (c2-standard-4, Intel) з великим набором даних у режимі прогрітого кешу, серія випробувань складала 500 ітерацій.

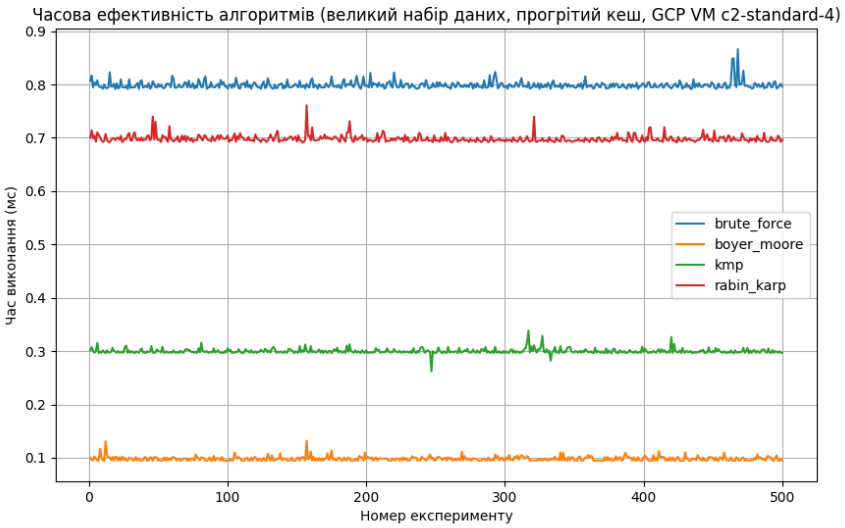

Рисунок 5 – час виконання алгоритмів у середовищі c2-standard-4

У табл. 1 наведено значення середніх часів виконання, медіани та довірчого інтервалу для усіх чотирьох алгоритмів у середовищі GCP VM (c2-standard-4, Intel) з великим набором даних у режимі прогрітого кешу, серія випробувань складала 500 разів. Як бачимо, алгоритм Бойера-Мура у цьому сценарії найшвидший і має найменший довірчий інтервал, що свідчить про його стабільність. Алгоритм Кнута-Морріса-Пратта приблизно втричі повільніший за Бойера-Мура, а алгоритм наївного пошуку (брут-форс) найповільніший за даних умов. Алгоритм Рабіна-Карпа має дещо кращий час виконання аніж алгоритм наївного пошуку, але в рази більший довірчий інтервал ніж у інших алгоритмів (дуже нестабільний).

Таблиця 1

Час виконання алгоритмів пошуку (Intel)

Параметр	boyer_moore	brute_force	kmp	rabin_karp
Середнє значення (мс)	0.098188	0.798881	0.299812	0.698225
Медіана (мс)	0.097596	0.796274	0.298501	0.695973
Довірчий інтервал (± мс)	0.000296	0.000546	0.000506	0.002880

У табл. 2 наведено значення S- та R-показників часової ефективності алгоритмів пошуку підрядка в рядку (Бойера-Мура, наївного пошуку, Кнута-Морріса-Пратта, Рабіна-Карпа). Експеримент проводився у середовищі GCP VM (c2d-highcpu-4, AMD) на великому наборі даних, в режимі прогрітого кешу, серія випробувань складала 500 ітерацій. Як бачимо з таблиці, алгоритм Бойера-Мура демонструє найвищу часову ефективність серед усіх алгоритмів: має перевагу над алгоритмом наївного пошуку (84.9 ±0.1%), КМП (60.0 ±0.7%) і Рабіна-Карпа (87.0 ±0.1%), причому перевага була зафіксована у 100% випадків (R: 100). Алгоритм КМП посідає друге місце в рейтингу – він кращий за алгоритм наївного пошуку (61.9 ±0.9%) та Рабіна-Карпа (67.3 ±0.8%), і програє тільки Бойера-Мура (-60.0 ±0.7%). Згідно з наведених даних найгіршим алгоритмом у цьому сценарії є алгоритм Рабіна-Карпа.

Таблиця 2

S- та R-показники ефективності (великий набір даних)

Алгоритм	boyer_moore	brute_force	kmp	rabin_karp
boyer_moore	X	S: 84.9 ±0.1	S: 60.0 ±0.7	S: 87.0 ±0.1
		R: 100	R: 100	R: 100
brute_force	S: -84.9 ±0.1	X	S: -61.9 ±0.9	S: 14.1 ±0.4
	R: 0		R: 0	R: 99
kmp	S: -60.0 ±0.7	S: 61.9 ±0.9	X	S: 67.3 ±0.8
	R: 0	R: 100		R: 100
rabin_karp	S: -87.0 ±0.1	S: -14.1 ±0.4	S: -67.3 ±0.8	X
	R: 0	R: 1	R: 0	

У табл. 3 наведено значення S- та R-показників часової ефективності алгоритмів пошуку підрядка в рядку (Бойера-Мура, наївного пошуку, Кнута-Морріса-Пратта, Рабіна-Карпа). Експеримент проводився у середовищі GCP VM (c2d-highcpu-4, AMD) на маленькому наборі даних, в режимі прогрітого кешу, серія випробувань складала 500 ітерацій. В даному сценарії найкращим є алгоритм КМП, а алгоритм Бойера-Мура посідає тільки друге місце. Згідно з

розрахунками найгіршим алгоритмом у цьому сценарії також є алгоритм Рабіна-Карпа, який програє навіть алгоритму наївного пошуку (брут-форс).

Таблиця 3

S- та R-показники ефективності (малий набір даних)

Алгоритм	boyer_moore	brute_force	kmp	rabin_karp
boyer_moore	X	S: 46.7 ±2.0	S: -22.6 ±0.6	S: 54.9 ±1.9
		R: 100	R: 1	R: 100
brute_force	S: -46.7 ±2.0	X	S: -58.7 ±2.1	S: 15.4 ±0.7
	R: 0		R: 0	R: 99
kmp	S: 22.6 ±0.6	S: 58.7 ±2.1	X	S: 65.0 ±1.9
	R: 99	R: 100		R: 100
rabin_karp	S: -54.9 ±1.9	S: -15.4 ±0.7	S: -65.0 ±1.9	X
	R: 0	R: 1	R: 0	

В результаті експериментального дослідження встановлено, алгоритм Боєра-Мура впевнено демонструє найкращу часову ефективність у більшості сценаріїв, особливо на великих та середніх наборах даних. Водночас алгоритм Кнута-Морріса-Пратта (КМП) у деяких випадках демонстрував порівняну або навіть кращу (особливо на малому обсязі даних) часову ефективність. В той же час, алгоритм Рабіна-Карпа проявляє нестабільність і високу дисперсію часу, особливо на маленьких об’ємах даних.

Що стосується впливу кешу («холодного»/«прогрітого») на результати роботи, то для «прогрітого» кешу спостерігається зменшення середнього часу виконання алгоритмів до 25%. При цьому, у режимі прогрітого кешу розбіжності між алгоритмами зменшуються, хоча алгоритм Боєра-Мура все одно зберігає лідерство. У режимі холодного кешу час виконання дуже коливається, особливо для алгоритму Рабіна-Карпа.

В подальших дослідженнях розглядається можливість використання вище зазначеного підходу для вибору алгоритму пошуку заданого набору ключових слів (навичок та компетенцій) у певній множині текстових файлів (резюме, анкетах та профілях кандидатів) для подальшого їх ранжування та побудови рекомендацій щодо формування команд для виконання ІТ проектів .

ЛІТЕРАТУРА

1. Шинкаренко В. І. Порівняльний аналіз часової ефективності функціонально еквівалентних алгоритмів. Проблеми програмування. – 2001. № 3–4. С. 31–39.

2. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. MIT Press. – 2009. 1292 p.
3. Knuth D.E., Morris J.H., Pratt V.R. Fast Pattern Matching in Strings. SIAM Journal on Computing. – 1977. Vol. 6(2). P. 323–350.
4. Boyer R. S., Moore J. S. A Fast String Searching Algorithm. Communications of the ACM. – 1977. Vol. 20(10). P. 762–772.
5. Karp R. M., Rabin M. O. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development. – 1987. Vol. 31(2). P. 249–260.

COMPARATIVE ANALYSIS OF TIME EFFICIENCY OF SUBSTRING SEARCH ALGORITHMS

Klymenko I.V., Lebid Y.A.

Abstract. *This paper presents a comparative analysis of the time efficiency of four classical string searching algorithms that attempt to identify the position where one or more pattern strings occur within a longer string or text. The evaluated algorithms include the naive (brute-force) method, Knuth–Morris–Pratt (KMP), Boyer–Moore, and Rabin–Karp. The experiments were conducted on three different processor architectures using three datasets of varying sizes. Each algorithm was tested under both cold and warm cache conditions. To reduce external influence, the benchmarking was limited to a single CPU core, and memory was forcibly cleaned after each run. The experimental results were analyzed using S- and R-indicators of efficiency, and practical recommendations are provided regarding the applicability of each algorithm under different operating conditions.*

Keywords: *information technology, machine-based benchmarking, S- and R-metrics, substring search algorithm, time efficiency of algorithms.*

REFERENCES

1. Shynkarenko V. I. Comparative Analysis of Time Efficiency of Functionally Equivalent Algorithms. Problems of Programming, 2001, Vol. 3–4, pp. 31–39.
2. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. MIT Press. – 2009. 1292 p.
3. Knuth D.E., Morris J.H., Pratt V.R. Fast Pattern Matching in Strings. SIAM Journal on Computing. – 1977. Vol. 6(2). P. 323–350.
4. Boyer R. S., Moore J. S. A Fast String Searching Algorithm. Communications of the ACM. – 1977. Vol. 20(10). P. 762–772.
5. Karp R. M., Rabin M. O. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development. – 1987. Vol. 31(2). P. 249–260.