УДК 004.4 http://doi.org/10.15407/pp2023.04.003

В.І. Шинкаренко, О.*В. Макаров*

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ДЕЯКИХ ДЕТЕРМІНОВАНИХ МЕТОДІВ ПЕРЕДОБРОБКИ СОРТУВАННЯ ДАНИХ

Для перевірки гіпотези щодо зменшення часу сортування алгоритмами різної обчислювальної складності здійснено експерименти. Опрацьовано декілька ідей із детермінованої передобробки масивів даних для алгоритмів сортування. Запропоновані відповідні алгоритми: швидка передобробка – передбачення індексу елемента у відсортованому масиві і перестановка, передобробка з пам'яттю – передбачення і перестановка із запам'ятовуванням раніше встановлених елементів, передобробка із розворотом – розвертання послідовностей елементів, відсортованих у зворотному порядку. Також запропоновані блочні варіації швидкої та передобробки з пам'яттю, які виконуються для частин масиву заданої довжини. Встановлено, що більшу ефективність передобробки вони показують разом із алгоритмами сортування, які значно прискорюються на відсортованих (або майже відсортованих) масивах даних. Блочні передобробки можуть виконуватися швидше через можливість уникнення промахів кешу (cache miss), однак показують нижчий відсоток впорядкованості масиву. Виконані експерименти з оцінки ефективності різних алгоритмів сортування після і разом із запропонованими передобробками.

Ключові слова: алгоритм сортування, сортування, часова ефективність, комбінований алгоритм, передобробка.

Вступ

Ера великих даних (Big Data) відкриває нові можливості, але має й більше проблем для алгоритмів сортування. Класичні алгоритми сортування не можуть пристосуватися до вибухового зростання даних. Існує велика кількість алгоритмів сортування за різноманіттям сценаріїв застосування, пристроїв для зберігання даних і стратегій удосконалення.

Алгоритми сортування класифікуються за різними показниками часової ефективності, стабільності і вимог до виділення додаткової пам'яті. Прийнято оцінювати алгоритми за показником асимптотичної обчислювальної складності O [1]. Найбільш популярні алгоритми сортування, такі як швидке сортування (Quick sort) чи сортування злиттям (Merge sort), мають обчислювальну складність у середньому $O(n * log(n))$. Однак на невеликих наборах даних сортування вставками (Insertion sort) з обчислювальною складністю $O(n^2)$ працює швидше.

Ефективні реалізації зазвичай використовують гібридний алгоритм, що поєднує асимптотично ефективний алгоритм для загального сортування із сортуванням вставками для невеликих масивів у нижній частині рекурсії. Використовуються і складніші варіанти, такі як, Timsort [2] – попередній аналіз даних та пошук відсортованих послідовностей, розбиття масиву на відсортовані ділянки за допомогою сортування вставками і модифікований алгоритм злиття в один відсортований масив. Або Introsort [3], що починає із швидкого сортування, потім переходить на Heapsort, коли глибина рекурсії перевищує заданий рівень, і закінчує сортуванням невеликих послідовностей за допомогою сортування вставками.

Досліджувалися можливі модифікації існуючих алгоритмів для покращення їхньої часової ефективності. Наприклад, у [4] використовувалось сортування вставками із розбиттям масиву на дві відсортовані частини і одну невідсортовану, з якої вибиралися елементи. У [5] досліджувалось швидке сортування з різною кількістю pivot елементів.

Попередню обробку можна представити як аналіз і деяку перестановку елементів масиву. Залежно від основного алгоритму сортування попередня обробка може робити перестановки, які зменшують час сортування. Або запобігають виник-

ненню найгірших ситуацій, що призводять до погіршання часової ефективності алгоритму. В даній праці запропоновано декілька алгоритмів попередньої обробки масивів сортованих даних.

У [6] досліджувалися методи стохастичної передобробки. Випадкові елементи, вибрані з усього масиву або із його половин, певну кількість разів (кратну розміру масиву) порівнюються і, якщо вони йдуть у зворотньому порядку, міняються місцями.

У [7] розглядається метод стохастичної передобробки, в якому випадкові елементи міняються місцями спочатку в межах половин вхідного масиву, а потім ще раз у межах цілого масиву. Також представлений детермінований метод передобробки, в якому порівнюються елементи з кінця та початку масиву і, якщо вони стоять у зворотному порядку, виконується перестановка, а індекси порівнюваних елементів зміщуються до середини масиву. Процес повторюється для усіх елементів масиву, а потім рекурсивно для підмасивів.

Показники ефективності передобробки

У цьому дослідженні ефективність передобробки оцінювалась за декількома показниками.

Поліпшення впорядкованості масиву визначається за показником невпорядкованості масиву [6] після закінчення одинарних або серійних перестановок:

$$U = \frac{1}{N}\left(\sum_{i=0}^{N}\frac{|j-i|}{max(\,i, N-i)}\right) * 100\%, \quad (1)$$

де j – індекс елемента у відсортованому масиві, i – індекс того ж елемента у невпорядкованому масиві, N – загальна кількість елементів масиву. Тут і далі індексація масивів починається з нуля. Якщо у масиві є декілька однакових елементів під індексами i_1 та i_2, для них будуть використані відповідні індекси у відсортованому масиві j_1 та j_2. Зазначимо, що у разі $i_1 < i_2$, тоді $j_1 < j_2$, як за умови стабільного сортування.

Для визначення найбільш ефективного алгоритму попередньої обробки масиву використовується формула:

$$Ef = \left(\frac{U_0 - U_p}{U_0}\right) * 100\% \quad (2)$$

де U_0 – показник невпорядкованості вхідного масиву, U_p – показник невпорядкованості масиву після виконання передобробки.

Часова ефективність алгоритмів оцінюється за S показником [8]. Якщо V^* – множина можливих значень обсягу даних, T^* – множина можливих типів даних, X^* – множина можливих значень даних, Ψ^* – множина можливих програмних середовищ, $\Re^*$ – множина архітектур ЕОМ, на яких можлива реалізація алгоритмів, тоді для порівняння часової ефективності альтернативних обчислювальних алгоритмів визначається ступінь переваги одного (і-го) алгоритму над іншим (j-им) на обмежених множинах, що є підмножинами зазначених вище множин $\bar{V} \subseteq V^*$, $\bar{T} \subseteq T^*$, $\bar{X} \subseteq X^*$, $\bar{\Psi} \subseteq \Psi^*$, $\bar{\Re} \subseteq \Re^*$:

$$SUP_{ij}\left|\bar{V}, \bar{T}, \bar{X}, \bar{\Psi}, \bar{\Re} = \right. \quad (3)$$

$$= \frac{1}{N}\sum_{\forall v \in \bar{V}}\sum_{\forall t \in \bar{T}}\sum_{\forall x \in \bar{X}}\sum_{\forall \psi \in \bar{\Psi}}\sum_{\forall r \in \bar{\Re}} g_{ij}(v,t,x,\psi,r),$$

де N – загальна кількість виконань, g_{ij} – ступінь переваги і-го алгоритму над j-им в точці, визначається як:

$$g_{ij}(v,t,x,\psi,r) = \frac{\tau_j(v,t,x,\psi,r) - \tau_i(v,t,x,\psi,r)}{\max\left(\tau_i(v,t,x,\psi,r), \tau_j(v,t,x,\psi,r)\right)}, \quad (4)$$

де $\tau_j(v,t,x,\psi,r)$ – час виконання j-го алгоритму, $\tau_i(v,t,x,\psi,r)$ – час виконання і-го алгоритму.

В експериментах змінюються розміри масивів даних ($\bar{V} \subseteq V^*$) і значення елементів ($\bar{X} \subseteq X^*$). Тип даних, програмна середа та архітектура ЕОМ залишаються постійними і можуть бути видалені з рівняння. Кінцева формула для визначення ступеня переваги алгоритму має вигляд:

$$g_{ij}(v,t) = \frac{\tau_j(v,x) - \tau_i(v,x)}{\max\left(\tau_i(v,x), \tau_j(v,x)\right)}, \quad (5)$$

Експериментально статистична оцінка S показника, як обґрунтовано у [8], обчислюється так:

$$S_{ij} = \frac{1}{N}\sum_{k=1}^{N} g_{ij}(v, x_k) \qquad (6)$$

де N – кількість експериментальних випробувань алгоритму, v – фіксований у конкретному експерименті розмір масиву, x_k – випадковим чином сформований масив у k-тому випробуванні. В експериментах використовується N рівне 51.

Детерміновані методи передобробки

Розглянемо запропоновані авторами методи передобробок, що використовуються для покращення часової ефективності алгоритмів сортування.

Швидка передобробка (QP). Основна ідея швидкої передобробки полягає у спробі «передбачити» позицію елемента в масиві. Знаходиться мінімальне та максимальне значення елементів масиву. Для кожного елемента розраховується передбачуване місце (індекс) у відсортованому масиві

$$i = \frac{(x_i - \min\limits_i x_i)}{(\max\limits_i x_i - \min\limits_i x_i)} * (N-1), \qquad (7)$$

де x_i– поточний елемент масиву, N – кількість елементів у масиві. Виконується перестановка поточного елемента з елементом за передбаченим індексом. Операція повторюється аж доки для поточного елемента не буде передбачений його ж індекс, тобто, коли елемент уже стоїть на передбаченому місці, або елемент за передбаченим індексом дорівнює поточному. Кількість передбачень для поточного місця елемента не може перевищувати наперед заданого M (це дозволяє уникнути зациклювання).

Після передобробки деяка кількість елементів опиниться на місцях, відповідних їхнім місцям у відсортованому масиві. Таким чином зменшується кількість перестановок елементів, що буде виконана безпосередньо алгоритмом сортування.

Покажемо на масиві випадкових цілих чисел у діапазоні від 0 до 9 роботу передобробки. Визначаємо мінімальний і максимальний елементи масиву. Вони дорівнюють відповідно 0 і 9. Максимальний індекс дорівнює N-1 = 9. Починаємо передобробку з першого елемента масиву. Жирним шрифтом виділено поточний елемент. Затіненням жовтого кольору показано передбачене місце для поточного елемента. Праворуч від масиву на кожному кроці наведені значення показника невпорядкованості масиву (U_i).

6	0	4	4	1	3	8	9	2	5	U_0=39.5
8	0	4	4	1	3	6	9	2	5	U_1=38.8
2	0	4	4	1	3	6	9	8	5	U_2=25.3
4	0	2	4	1	3	6	9	8	5	U_3=24.8
1	0	2	4	4	3	6	9	8	5	U_4=17.1
0	1	2	4	4	3	6	9	8	5	U_5=14.9
0	**1**	2	4	4	3	6	9	8	5	U_6=14.9
0	1	**2**	4	4	3	6	9	8	5	U_7=14.9
0	1	2	**4**	4	3	6	9	8	5	U_8=14.9
0	1	2	4	**4**	3	6	9	8	5	U_9=14.9
0	1	2	4	4	**3**	6	9	8	5	U_{10}=14.9
0	1	2	3	4	**4**	6	9	8	5	U_{11}=7.9
0	1	2	3	4	4	**6**	9	8	5	U_{12}=7.9
0	1	2	3	4	4	6	**9**	8	5	U_{13}=7.9
0	1	2	3	4	4	6	**5**	8	9	U_{14}=3.1
0	1	2	3	4	5	6	**4**	8	9	U_{15}=6.5
0	1	2	3	4	5	6	4	**8**	9	U_{16}=6.5
0	1	2	3	4	5	6	4	8	**9**	U_{17}=6.5

Рис.1. Приклад швидкої передобробки (QP)

Передобробка з пам'яттю (PM). Відмінність від швидкої передобробки полягає у запам'ятовуванні передбачених індексів, у яких була виконана перестановка. Алгоритм не намагається передбачати ін-

декс для того ж самого елемента декілька разів і для поточного елемента буде знайдено наступний ще непередбачений індекс.

У передобробці з пам'яттю додатково використовується окремий масив булевих змінних. Після того, як у передбачений індекс p встановлюється елемент масиву, булевий флаг за індексом p встановлюється в значення true. Булеві значення, що відповідають кожному елементу масиву, відобразимо під кожним елементом. Напочатку усі булеві флаги дорівнюють false, що відображається знаком "-". Після встановлення елемента булеве значення зміниться на true ("+"). Жирним виділимо поточний елемент, затіненням жовтого кольору – елемент під передбаченим індексом, зеленого – елемент переставлений за передбаченим на попередньому кроці індексом. Праворуч від масиву на кожному кроці записані значення невпорядкованості масиву (U_i).

6	0	4	4	1	3	8	9	2	5
-	-	-	-	-	-	-	-	-	-

U_0=39.49

8	0	4	4	1	3	6	9	2	5
-	-	-	-	-	-	+	-	-	-

U_1=38.82

2	0	4	4	1	3	6	9	8	5
-	-	-	-	-	-	+	-	+	-

U_2=25.32

4	0	2	4	1	3	6	9	8	5
-	-	+	-	-	-	+	-	+	-

U_3=24.82

1	0	2	4	4	3	6	9	8	5
-	-	+	-	+	-	+	-	+	-

U_4=17.06

0	1	2	4	4	3	6	9	8	5
-	+	+	-	+	-	+	-	+	-

U_5=14.95

0	1	2	**4**	4	3	6	9	8	5
+	+	+	-	+	-	+	-	+	-

U_6=14.95

0	1	2	**3**	4	4	6	9	8	5
+	+	+	-	+	+	+	-	+	-

U_7=7.85

0	1	2	3	4	4	6	**9**	8	5
+	+	+	+	+	+	+	-	+	-

U_8=7.85

0	1	2	3	4	4	6	**5**	8	9
+	+	+	+	+	+	+	-	+	+

U_9=3.09

0	1	2	3	4	4	6	5	8	9
+	+	+	+	+	+	+	+	+	+

U_{10}=3.1

Рис. 2. Приклад передобробки з пам'яттю

Потребує пояснення вибір місця і перестановка на сьомому кроці, тому що для елемента «4» передбачене місце під індексом 4, де стоїть елемент «4», але у масиві флагів стоїть «+». Тому вибирається наступне після передбаченого місце, в якому флаг дорівнює «-». Таке місце знаходиться під індексом «5», за яким стоїть елемент «3».

Блочна локальна передобробка. Ідея блочної локальної передобробки полягає в розбитті масиву на блоки певної довжини і застосуванні алгоритмів, описаних раніше на кожному підмасиві. Маємо дві комбінації алгоритмів: блочна локальна швидка передобробка (BLQP) і блочна локальна передобробка з пам'яттю (BLPM). У результаті початковий масив буде мати N частково (або повністю) відсортованих підмасивів. Масив після локальної передобробки матиме «форму пилки». За умови правильного вибору довжини блоку, меншої за довжину кешу процесора, поділеної на розмір елементів у сортованому масиві, можна значно зменшити кількість (або взагалі уникнути) промахів кешу (cache miss). У разі, якщо розмір блока буде дуже великим, отримаємо багато промахів кешу і зменшення часової ефективності роботи алгоритму.

Блочна глобальна передобробка. Глобальна передобробка також включає розбиття масиву на блоки певної довжини і застосування швидкої (BGQP) та передобробки з пам'яттю (BGPM) до підмасивів. Відмінність від локального алгоритму полягає у тому, що для кожного елемента масиву передбачається глобальний індекс, тобто приблизне місце, на якому стояв би даний елемент у відсортованому масиві. Після передбачення елемент може бути переставлений, тільки якщо передбачений індекс не виходить за межі поточного блока. Такий підхід також дозволяє уник-

нути промахів кешу (cache miss). Але приблизна вірогідність того, що елемент буде переставлений, дорівнює $1/K$ (де K – кількість блоків), для масиву рівномірно розподілених випадкових чисел.

Попередня обробка із розворотом (SR). В останньому варіанті попередньої обробки виконується прохід масивом, й усі послідовності, відсортовані у зворотному напрямку, розвертаються. Тож максимальна довжина послідовностей, відсортованих у зворотному напрямку, буде дорівнювати 2 (кінець попередньої і початок наступної відсортованої послідовностей). Найбільший вплив даний вид попередньої обробки буде мати для швидкого сортування (Quick sort), для якого вхідний масив, відсортований у зворотному порядку, є спеціальним випадком, де обчислювальна складність зростає до $O(n^2)$.

Наведемо приклад роботи попередньої обробки із розворотом на масиві випадкових цілих чисел від 0 до 9. Послідовності, відсортовані у зворотному порядку, перед розворотом виділені затіненням жовтого кольору, після – зеленого кольору.

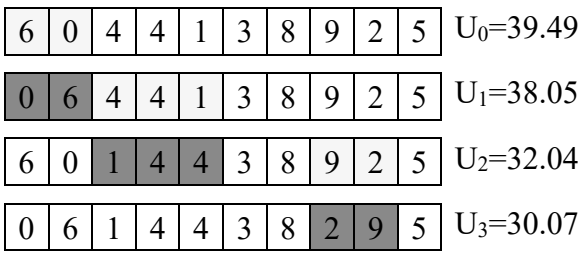

Рис.3. Приклад передобробки із розворотом

Отож, досліджено сім детермінованих алгоритмів передобробки:
– швидка передобробка (QP);
– передобробка з пам'яттю (PM);
– передобробка із розворотом (SR);
– блочна локальна швидка передобробка (BLQP);
– блочна локальна передобробка з пам'яттю (BLPM);
– блочна глобальна швидка передобробка (BGQP);
– блочна глобальна передобробка з пам'яттю (BGPM).

Організація експериментів із дослідження ефективності передобробки

Структури даних. Дослідження відбувалося на масивах цілих 4-байтових чисел. Пам'ять під масиви виділялася безпосередньо засобами C++, створенням std::vector<int> та викликом std::vector<int>::reserve. Сортований масив заповнювався рівномірно розподіленими псевдовипадковими числами від 0 до N-1, де N – кількість елементів масиву x_i. Псевдовипадкові числа були отримані генератором псевдовипадкових величин std::mt19937 з початковим станом, заданим генератором std::random_device. Згенеровані числа приведені до значення із заданого інтервалу за допомогою std::uniform_int_distribution.

Особливості вимірювання часу виконання алгоритму. У зв'язку з нестабільністю роботи потоків операційної системи Windows та технічних засобів, час сортування одного й того ж масиву буде випадковим із деякою дисперсією.

Час сортування визначався як медіанний час із *M*-кратним виконанням сортування одного й того ж масиву даних (дані копіювалися з оригінального невідсортованого масиву перед кожним вимірюванням).

Перед виконанням окремого сортування (паралельного вимірювання) виконувалося очищення кешу. Для цього застосовувався такий спосіб: створювався масив розміром $N = 8 * 2^{20}$ 4-байтових псевдовипадкових чисел, 2 випадкових елементи масиву мінялись місцями за допомогою std::swap N разів.

Технічні засоби. Експерименти виконані на ноутбуці з технічними характеристиками, наведеними у табл. 1.

Таблиця 1

Характеристики технічних засобів

Характеристики центрального процесору	
Поле	Значення
Тип	Intel Core i7-6700HQ, 2600 МГц

Кількість ядер	4
Кеш L1	256 KB per core
Кеш L2	1 MB per core (On-Die, ECC, Full-Speed)
Кеш L3	6 MB (On-Die, ECC, Full-Speed)
Набір інструкцій	64-Bit, Intel® SSE4.1, Intel® SSE4.2, Intel® AVX2, Intel AES
Характеристики оперативної пам'яті	
Поле	Значення
Формфактор	FB-DIMM
Розмір модулю	8 ГБ
Кількість модулів	2 x 8 ГБ

Результати експериментів

Ефективність передобробки будемо оцінювати за показниками (1)-(3). Для дослідження обирались загальновідомі алгоритми сортування, які зазвичай використовують для вирішення задач із різними вимогами. Швидке сортування [9] (Quick sort) є одним із найшвидших універсальних алгоритмів сортування. Використовується для великих списків, коли важлива ефективність. Сортування вставками [10] (Insertion sort) ефективне для малих або частково відсортованих списків. Може використовуватись як базовий варіант для інших алгоритмів. Шейкерне сортування [11] (Cocktail sort) подібне до сортування бульбашкою [12], але виконується в обидва напрямки. Може бути ефективно використане для невеликих або майже відсортованих масивів.

Дослідження невпорядкованості масиву до та після передобробок. Для визначення найбільш ефективного алгоритму попередньої обробки було проведено експеримент, де для масиву випадкових цілих чисел у діапазоні [0, N-1] (де N – розмір масива) вираховувався показник ефективності за формулою (2). Максимально можливе значення ефективності 100% означало б, що результатом передобробки є повністю відсортований масив. На рисунку .4 видно, що найбільшу ефективність має передобробка з пам'яттю. Далі із значенням близько 62% йде швидка передобробка. Передобробка з розворотом показала значення ефективності близьке до 0% для усіх масивів даних з експерименту.

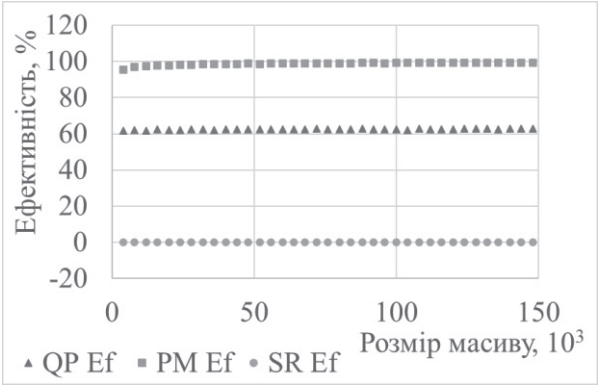

Рис.4. Ефективність передобробок

Дослідження часової ефективності швидкого сортування з використанням різних типів передобробок.

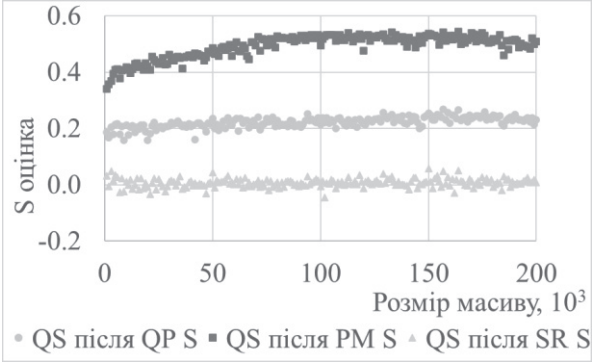

Рис.5. S оцінка швидкого сортування після швидкої, з пам'яттю і передобробки з розворотом

Найбільше зростання часової ефективності швидкого сортування дає попередня обробка з пам'яттю. Значення S оцінки стартує з 0.35 на 1000 елементів і стрімко зростає у разі збільшення розміру масиву. На 50 – 60 тисячах елементів S оцінка досягає 0.5 і лишається на тому ж рівні за подальшого збільшення кількості елементів.

Швидка передобробка має стабільні показники S оцінки у межах 0.18 – 0.25 для усіх розмірів масиву. Передобробка із розворотом має показник близький до 0, плюс мінус 0.05.

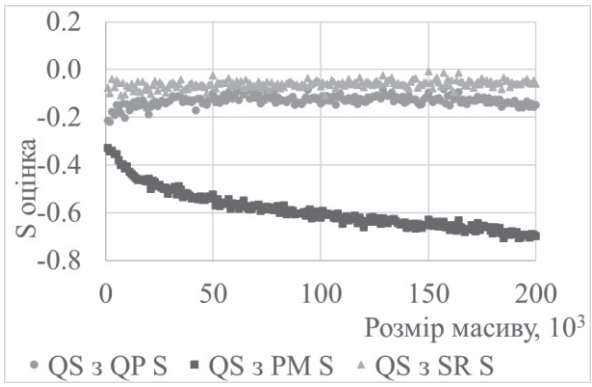

Рис.6. S оцінка швидкого сортування разом із швидкою, з пам'яттю і передобробкою з розворотом

Однак усі досліджені типи передобробок разом зі швидким сортуванням у порівнянні із чистим сортуванням мають негативні значення S оцінки. Для передобробки із розворотом ці значення знаходяться у діапазоні від - 0.02 до - 0.12. Швидка показує значення у діапазоні - 0.1..- 0.2. А для передобробки з пам'яттю S оцінка різко знижується від -0.32, для масиву 1000 елементів, до -0.5 – на 30 тисячах. Далі зі збільшенням розміру масиву продовжується падіння показника до -0.7 на 200 тисячах елементів.

Дослідження впливу блочних передобробок. Блочні локальні передобробки мають позитивний вплив на часову ефективність швидкого сортування. S оцінка сортування після локальних передобробок знаходиться на рівні 0.1 на малих розмірах масиву. На масивах близько ста тисяч і більше елементів показник коливається у межах 0.1 – 0.2.

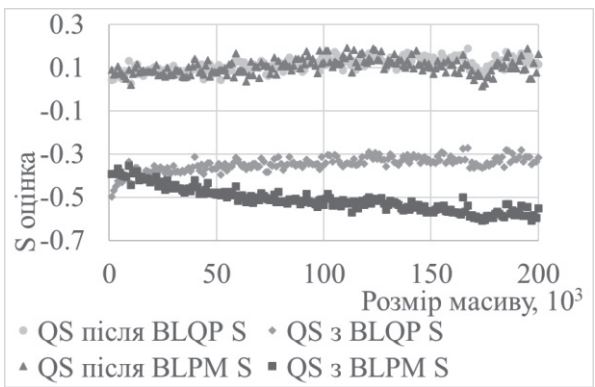

Рис.7. S оцінка швидкого сортування після блочної локальної швидкої та передобробки з пам'яттю

Локальні передобробки разом із швидким сортуванням мають негативні значення S оцінки. Блочна локальна швидка показує значення близько -0.4 – -0.5 на дуже малих розмірах масиву, а після десяти тисяч елементів і до кінці експерименту показує значення у діапазоні -0.3..-0.4. Блочна локальна з пам'яттю починає зі значення близько -0.39 і рівномірно погіршується впродовж усього експерименту, досягаючи значення -0.6.

Блочні глобальні передобробки також мають позитивний вплив на швидке сортування, але дещо менший, ніж локальні. Значення S оцінки швидкої передобробки коливається у межах -0.02 – 0.1 впродовж усього експерименту. Передобробка з пам'яттю показує значення від -0.03 до 0.17.

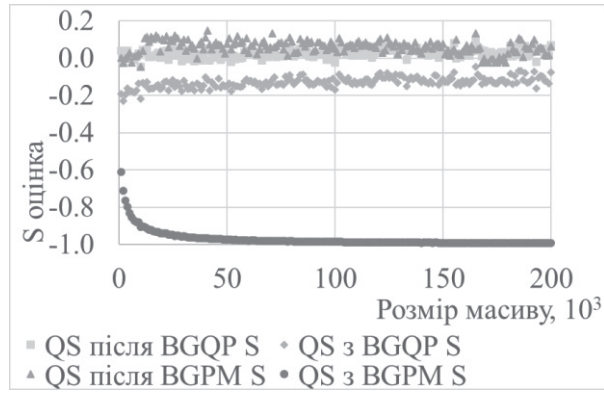

Рис.8. S оцінка швидкого сортування після блочної глобальної швидкої та передобробки з пам'яттю

Значення S оцінки для блочних глобальних передобробок разом із сортуванням є негативним. Швидка передобробка має значення у діапазоні -0.22..-0.07. А з пам'яттю починає з -0.6, сягає -0.9 на десяти тисячах елементів, а після сімдесяти тисяч лишається близьким до -1.

Підсумки. Швидке сортування має складність O(n*log(n)) у кращому випадку, і відсортованість даних у масиві перед початком сортування майже не впливає на час виконання. Усі запропоновані типи передобробок разом зі швидким сортуванням показали довший час виконання, ніж окреме сортування. Однак деякі передобробки показали позитивний вплив на часову ефективність сортування. Передобробка з пам'яттю досягла показника S оцінки 0.5, а швидка передобробка – 0.25. Найкращий показник серед блочних передобробок показала глобальна передобробка з пам'яттю – до 0.17. Решта показала значення на рівні 0.1 – 0.2.

Дослідження часової ефективності сортування вставками з використанням різних типів передобробок. Значення S оцінки для сортування вставками після швидкої передобробки має позитивне значення на малих розмірах масиву. Для масиву довжиною 100 елементів оцінка дорівнює 0.3. Далі значення швидко зростає до 0.5 для масиву довжиною 1000. Зі збільшенням довжини масиву до 100000 елементів, значення S оцінки лишається на рівні 0.55.

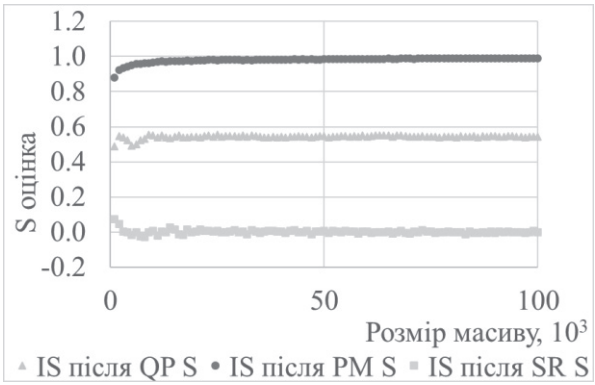

Рис.9. S оцінка сортування вставками після швидкої, з пам'яттю і передобробки з розворотом

Сортування після передобробки з пам'яттю має вищі показники S оцінки. Для масива довжиною 100 елементів оцінка дорівнює 0.55 і стрімко зростає. На 2000 елементів значення S оцінки сягає 0.92, на 10000 – 0.97 й із подальшим збільшенням кількості елементів наближається до 1.

Передобробка із розворотом має показник S оцінки дещо вище 0 на малих розмірах масиву, досягає 0 після 1000 і залишається таким протягом усього експерименту.

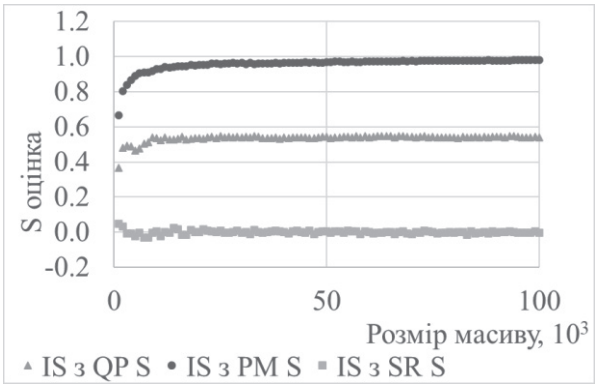

Рис.10. S оцінка сортування вставками зі швидкою, з пам'яттю і передобробкою з розворотом

Швидка передобробка показує негативне значення S оцінки на малих розмірах масиву. Для розміру масиву в діапазоні 200..300 елементів значення близьке до 0. Далі значення рівномірно зростає до 0.53 для масиву довжиною 10000. Подальше збільшення довжини масиву не змінює значення S оцінки.

Передобробка з пам'яттю також має негативне значення S оцінки на малих розмірах масивів. Оцінка для масиву з 300 елементів досягає 0 і продовжує стрімко зростати. На 2000 елементів значення S оцінки сягає 0.8, на 6000 – 0.9 і, за подальшого збільшення кількості елементів, наближається до 1.

Передобробка із розворотом має показник S оцінки дещо нижче 0 на малих розмірах масиву, досягає 0 після 1000 і залишається таким протягом усього експерименту.

Дослідження впливу блочних передобробок. Сортування вставками після обох блочних глобальних передобробок має показник S оцінки у діапазоні від -0.04 до 0.1 на малих розмірах масивів. Із збільшенням довжини масиву діапазон звужується до значень -0.009..0.03. Водночас швидка передобробка має незначну перевагу над передобробкою з пам'яттю.

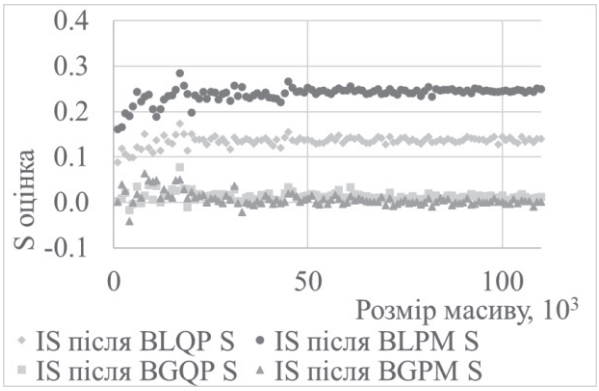

Рис.11. S оцінка сортування вставками після блочних передобробок

Блочні локальні передобробки мають більшу ефективність порівняно з глобальними. Сортування після швидкої передобробки має показник S оцінки 0.029..0.17. Із збільшенням довжини масиву діапазон значень звужується до величин 0.127..0.145.

Блочна локальна передобробка з пам'яттю на малих розмірах масиву показує більший діапазон значень S оцінки 0.128..0.285. За умови збільшення довжини масиву діапазон звужується до значень 0.224..0.271.

Найгірші показники S оцінки показує сортування вставками після блочної глобальної передобробки з пам'яттю. На малих розмірах масивів значення коливаються у діапазоні -0.31..-0.24, для масивів понад 50000 елементів і до кінця експерименту – - 0.29..- 0.28.

Блочна глобальна швидка передобробка дає негативні значення S оцінки на масивах малої довжини. Досягає 0 на масивах довжиною 5000 елементів і лишається на тому ж рівні до кінця експерименту.

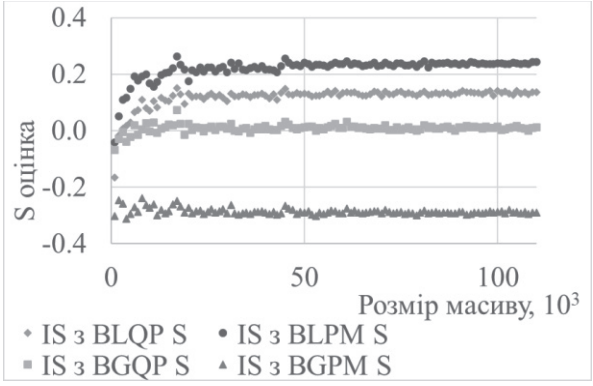

Рис.12. S оцінка сортування вставками разом із блочними передобробками

Блочні локальні передобробки показують вищий рівень значень S оцінки ніж глобальні. Швидка передобробка має негативне значення на малих розмірах масивів. Для масивів довжиною 15000 і до кінця експерименту S оцінка лишається на рівні 0.11..0.13.

Передобробка з пам'яттю також має негативні значення S оцінки на малих розмірах масиву. На масивах довжиною понад 15000 показує значення у діапазоні 0.2..0.26.

Підсумки. Сортування вставками має складність O(n) у кращому випадку – на повністю відсортованому масиві. Тому передобробки разом із сортуванням мають переважно позитивні значення S оцінки. А через те, що час передобробки дуже малий порівняно із часом сортування, результати S-оцінок для алгоритму сортування після і разом з передобробкою відрізняються на 0.01 – 0.02. Передобробка з розворотом має показники на рівні 0, швидка передобробка – 0.53, передобробка з пам'яттю – наближається до 1 зі збільшенням розміру масиву. Серед блочних найбільші показники у локальної передобробки з пам'яттю – 0.26, локальна швидка має – 0.13, глобальна швидка – 0. Найнижчі показники у глобальної передобробки з пам'яттю – -0.

Дослідження часової ефективності шейкерного сортування з використанням різних типів передобробок. Шейкерне сортування після швидкої передобробки має позитивне значення S оцінки на всіх довжинах масивів з експерименту. На малому розмірі масиву S оцінка дорівнює 0.54 і швидко зростає зі збільшенням розміру масиву. На масивах близько 1000 елементів сягає 0.58, а вже після 7000 елементів виходить на свій постійний рівень – 0.7.

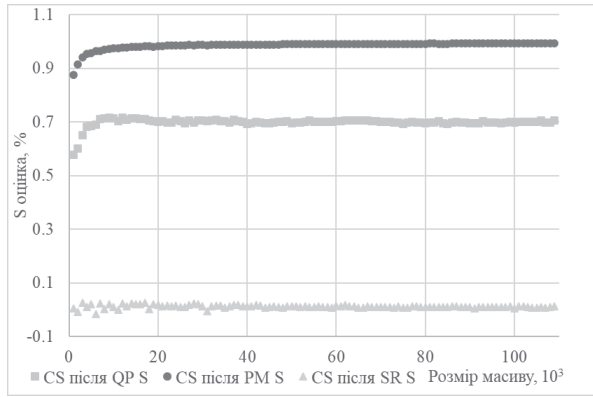

Рис.13. S оцінка шейкерного сортування після швидкої, з пам'яттю і передобробки з розворотом

Передобробка з пам'яттю має досить велике значення (0.7) S оцінки на малих розмірах масиву. Масив з 1000 елементів досягає 0.87, а після 10000 – 0.98. Далі на всіх довжинах масивів, використаних в експерименті, значення S оцінки сягає і перевищує 0.99.

Шейкерне сортування після передобробки з розворотом на масивах довжиною до 1000 показує значення S оцінки у діапазоні -0.02..0.06. На довших масивах діапазон значень звужується і лишається рівним 0.0..0.02.

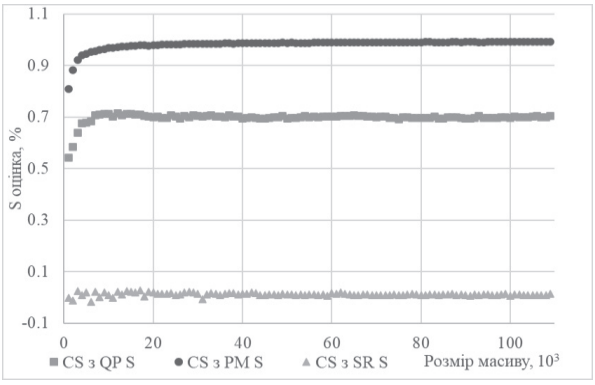

Рис.14. S оцінка шейкерного сортування разом із швидкою, з пам'яттю і передобробкою з розворотом

Усі вищезгадані типи передобробок займають мало часу порівняно з шейкерним сортуванням. Тому відмінність у показниках S оцінки можна побачити на малих розмірах масивів. Початкове значення для швидкої передобробки для масиву довжиною 100 дорівнює 0.3, а вже для 1000 досягає 0.54, після 7000 елементів і до кінця експерименту – 0.7.

Передобробка з пам'яттю має досить мале значення 0.15 для масиву зі 100 елементів. А при довжині рівній 1000 – 0.81. На великих розмірах сягає 0.99 і наближається до 1.

Значення для передобробки з розворотом знаходяться у діапазоні -0.034..0.008 для масивів довжиною до 1000 елементів. Далі усі значення S оцінок сортування з передобробками співпадають із результатами сортування після передобробок.

Дослідження впливу блочних передобробок. Найкращі показники S оцінки на великих розмірах масивів показує блочна локальна передобробка з пам'яттю. На масивах до 1000 елементів значення збільшуються від 0.16 до 0.29. На 10000 дорівнює 0.54, а на 20000 досягає 0.6 і лишається на тому ж рівні до кінця експерименту.

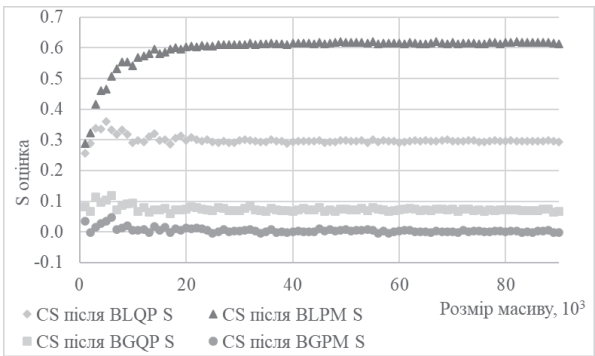

Рис.15. S оцінка шейкерного сортування після блочних передобробок

Блочна локальна швидка передобробка має показники S оцінки у діапазоні 0.12..0.26 на масивах довжиною до 1000 елементів. Для масивів 1000–10000 елементів діапазон розширюється і має значення 0.26..0.34. Для більших масивів показник лишається на рівні 0.3 протягом усього експерименту.

Блочні глобальні передобробки показують значно нижні результати S оцінки порівняно з локальними. Швидка передобробка має значення у діапазоні 0.01..0.13 на масивах довжиною до 10000. На довших масивах показники варіюються від 0.07 до 0.08.

Блочна глобальна передобробка з пам'яттю має найнижчі показники S оцінки. На масивах до 10000 елементів – -0.04..0.06, на довших масивах – -0.06..0.04. Зі збільшенням довжини масиву наближається до нуля.

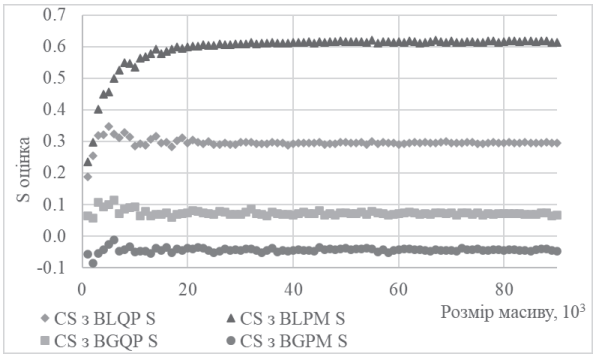

Рис.16. S оцінка шейкерного сортування разом із блочними передобробками

Шейкерне сортування разом із блочними локальними передобробками має нижчі показники S оцінок, порівняно із сортуванням після передобробок, на малих розмірах масивів. Для швидкої передобробки на масивах довжиною менше 10000 елементів значення оцінки стрімко зростають від -0.26 до 0.19. А для передобробки з пам'яттю – від -0.29 до 0.24. Далі протягом усього експерименту значення S оцінок співпадають із результатами сортування після блочних локальних передобробок.

Блочна глобальна швидка передобробка на масивах довжиною до 10000 елементів має показник S оцінки, що зростає з -0.07 до 0.06. Для довших масивів значення знаходяться у діапазоні 0.065..0.076.

Блочна глобальна передобробка з пам'яттю має негативні показники S-оцінки протягом усього експерименту. На масивах довжиною до 10000 елементів значення лежать у діапазоні -0.29..-0.05, а більше 10000 – -0.06..-0.01.

Підсумки. Шейкерне сортування, як і сортування вставками, має складність O(n) у кращому випадку, що підтверджують високі експериментально отримані показники S оцінки для сортування разом із передобробками. Для швидкої передобробки – 0.7, передобробки з пам'яттю – 0.99. Ефективність передобробки із розворотом близька до 0. Серед блочних алгоритмів найбільш ефективна локальна з пам'яттю – 0.6, локальна швидка – 0.3, глобальна швидка – 0.08, глобальна з пам'яттю – близька до 0.

Висновки

Встановлено, що із розмежуванням процесів обробки та сортування можливо отримати значне скорочення часу виконання алгоритмів сортування. Більш ефективна передобробка для алгоритмів сортування, що прискорюються з майже відсортованими даними.

Передобробка з пам'яттю дозволяє отримати краще впорядковані масиви даних, але потребує більше часу та пам'яті порівняно зі швидкою передобробкою.

Передобробка із розворотом не погіршує, але і не поліпшує в межах похибки час сортування для усіх досліджених алгоритмів.

Для великих обсягів даних, більших, ніж довжина кешу, ефективнішим може стати використання блочних локальних або глобальних передобробок. Через їхню особливість – не виконувати перестановки, якщо передбачений елемент знаходиться за межами блоку, - можна мінімізувати або зовсім уникнути промахів кешу (cache miss).

У випадках, коли сумарний час передобробки і сортування перевищує час сортування, використання передобробки видається недоцільним. Однак залежно від архітектури і вимог до даних у програмі передобробка може викликатись один або багато разів протягом роботи програми, за умови надходження певної кількості нових даних. Таким чином дані можна тримати частково відсортованими. А за потреби у повністю відсортованих даних, викликати безпосередньо алгоритм сортування, який виконається швидше через те, що дані знаходяться у «передобробленому» стані.

Найбільший ефект застосування запропонованих алгоритмів передобробки виявлено під час застосування передобробки з пам'яттю разом із сортуванням вставками і шейкерним сортуванням. Значення S оцінки у цих випадках наближається до 1. Також досить високу ефективність показує швидка передобробка для вищезгаданих алгоритмів сортування. Серед блочних передобробок локальні передобробки показали більшу ефективність, ніж глобальні.

Жодний тип передобробки не мав позитивного впливу у використанні разом зі швидким сортуванням. Час виконання передобробки перевищував виграш від скорочення часу сортування переднеоброблених даних.

Література

1. Knuth, Donald. The Art Of Computer Programming, vol. 3: Sorting And Searching. : Addison-Wesley, 1973.
2. Timsort [Online] – Available from: https://svn.python.org/projects/python/trunk/Objects/listsort.txt, last accessed 2023/08/07.
3. Musser, David R. (1997). "Introspective Sorting and Selection Algorithms". Software: Practice and Experience. 27 (8): 983–993.
4. Adnan Saher Mohammed, Şahin Emrah Amrahov, Fatih V. Çelebi. Bidirectional Conditional Insertion Sort algorithm; An efficient progress on the classical insertion sort. Future Generation Computer Systems, Volume 71, June 2017, 102-112.
5. Shrinu Kushagra, Alejandro López-Ortiz, Aurick Qiao, J. Ian Munro. Multi-Pivot Quicksort: Theory and Experiments. 2014 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX), 47-60
6. Шинкаренко В.І., Дорошенко А.Ю., Яценко О.А., Разносілін В.В., Галанін К.К. Двокомпонентні алгоритми сортування Проблеми програмування. – 2022. – № 3-4. – С. 32-41. – Бібліогр.: 18 назв. – укр.

7. Abbas Mubarak, Sajid Iqbal, Qaisar Rasool, Nabeel Asghar, Neetu Faujdar, & Abdul Rauf. (2022). Preprocessing: A method For Reducing Time Complexity. Journal of Computing & Biomedical Informatics, 4(01), 104–117.
8. Шинкаренко В.І Сравнительный анализ временной эффективности функционально эквивалентных алгоритмов / В. И. Шинкаренко // Проблемы программирования. – 2001. – № 3-4. – С. 31-39
9. Hoare, C. A. R. (1962). "Quicksort". Comput. J. 5 (1): 10–16.
10. Insertion sort [Online] – Available from: https://en.wikipedia.org/wiki/Insertion_sort, last accessed 2023/08/07.
11. Knuth, Donald E. (1973). "Sorting by Exchanging". Art of Computer Programming. Vol. 3. Sorting and Searching (1st ed.). Addison-Wesley. pp. 110–111.
12. Bubble sort [Online] – Available from: https://en.wikipedia.org/wiki/Bubble_sort, last accessed 2023/08/07.

Одержано: 18.10.2023

Про авторів:

Шинкаренко Віктор Іванович,
доктор технічних наук, професор,
Кількість публікацій в українських виданнях – більше 200
Кількість зарубіжних публікацій – більше 30
індекс Хірша – 6
https://orcid.org/0000-0001-8738-7225
E-mail: shinkarenko_vi@ua.fm

Макаров Олексій Вікторович,
аспірант,
Кількість публікацій в українських виданнях – 1
https://orcid.org/0009-0003-0921-155X
E-mail: makarovov@hotmail.com

Місце роботи авторів:

Український державний університет науки і технологій,
49010, Україна, Дніпро,
вул. академіка Лазаряна, 2.
E-mail: office@ust.edu.ua