УДК 004.94(045)

Кондес Ю. В.
Національний авіаційний університет, Київ

ЗБІЛЬШЕННЯ ШВИДКОДІЇ ОПЕРАЦІЙ МНОЖЕННЯ ТА ДІЛЕННЯ НАД ЧИСЛАМИ У РОЗРЯДНО-ЛОГАРИФМІЧНОМУ ПРЕДСТАВЛЕННІ

В даній статті розглянуто проблеми що виникають при обчисленнях з використанням розрядно-логарифмічної системи числення на звичайних IBM PC сумісних ПК. Розглядаються методи збільшення швидкодії, засновані на підборі оптимального алгоритму сортування та зміні структури РЛ числа, яка дозволяє зменшити кількість дій у операціях множення та ділення. Досліджується ефект від застосування запропонованих методів збільшення швидкодії.

Моделювання засноване на використанні обчислювальної техніки є основним засобом вирішення багатьох задач, що виникають в процесах проектування та дослідження. Розвиток методів моделювання здійснюється одночасно з урахуванням появи нових обчислювальних методів, алгоритмів а також інших досягнень в різних галузях науки та техніки [1].

Одним з засобів представлення даних в комп'ютерному середовищі є розрядно-логарифмічне кодування, яке дозволяє значно розширити числовий діапазон оброблюваних величин. Це дає можливість підвищити точність у обчисленнях чутливих до похибок за рахунок зменшення кількості та величини округлень у проміжних результатах.

Відомо, що будь-яке додатне число A, з будь-якої позиційної системи числення, може бути представлене у вигляді:

$$A = \sum_{i=-\infty}^{m} a_i \cdot p^i = a_m \cdot p^m + a_{m-1} \cdot p^{m-1} + \\ + a_{m-2} \cdot p^{m-2} + a_0 \cdot p^0 + a_{-1} \cdot p^{-1}, \quad (1)$$

де a_i – значення певної цифри числа A, p – основа системи числення, a_m – старший значущий розряд числа A [1].

Для двійкової системи числення p=2, а значення кожної цифри належить множині {0, 1}. Тому в даному випадку вираз (1) можна представити у вигляді:

$$A = p^{n_i} + p^{n_{i-1}} + p^{n_{i-2}} + \ldots + p^{n_2} + p^{n_1} \quad (2)$$

де p – основа системи числення (p = 2); n_1, n_2, … , n_i – ступені двійки, які відповідають вазі тих розрядів двійкового числа, які дорівнюють 1. Вираз (2) є математичною формою представлення числа в розрядно-логарифмічній арифметиці.

У пам'яті ЕОМ РЛ число зберігається у вигляді [1]:

Sign	Q	n_i	n_{i-1}	n_{i-2}	..	n_2	n_1

де Sign знак числа (1 або -1), Q – довжина РЛ числа (кількість елементів), n – окремі елементи РЛ числа (ступені двійки). Всі елементи числа розташовані за спаданням від i-го до 1-го.

1. Проблеми швидкодії при використанні розрядно-логарифмічної системи числення

При виконанні додавання двох РЛ чисел (A та B) однакової довжини елементи кожного з доданків необхідно об'єднати у третє число-суму, яке складатиметься з $2Q$ елементів і матиме наступний вигляд [1]:

Sign	$2Q$	$n_{B(i)}$	$n_{B(i-1)}$	$n_{B(i-2)}$	…	$n_{B(2)}$	$n_{B(1)}$	$n_{C(i)}$	$n_{C(i-1)}$	$n_{C(i-2)}$	…	$n_{C(2)}$	$n_{C(1)}$

Даний набір елементів необхідно 1 раз відсортувати та звести подібні. Операція віднімання подібна до додавання з тією різницею, що додатково необхідно виконати розгортання найбільшого елемента в ланцюжок. Кількість сортувань та зведень подібних не змінюється.

При виконанні операції множення двох РЛ чисел (A та B) необхідно кожний елемент одного множника скласти з окремим набором усіх елементів іншого множника. В результаті отримаємо наступний результат [1]:

Sign	$Q*Q$	$n_{B(i)}+n_{C(i)}$	$n_{B(i)}+n_{C(i-1)}$	$n_{B(i)}+n_{C(i-2)}$	…	$n_{B(i)}+n_{C(2)}$	$n_{B(i)}+n_{C(1)}$	…

У комп'ютерному представленні для випадку множників однакової довжини, виникає необхідність у виконані Q-1 кількості сортувань та зведень подібних або одне сортування та зведення подібних для числа довжиною Q^2 в залежності від обраного методу реалізації. Ділення виконується з використанням операцій множення та віднімання, кількість кожної з яких наближено дорівнює Q^2.

2. Збільшення швидкодії шляхом вибору оптимального алгоритму сортування

За приблизними підрахунками (для комп'ютерів 60-х років) на сортування в типових задачах витрачається більше чверті машинного часу. В багатьох обчислювальних системах цей показник доходить до 50 %. Такий стан справ виникає через високий рівень практичних потреб у операціях сортування даних та розповсюдженість неоптимальних алгоритмів сортування [2]. При обчисленнях на основі розрядно-логарифмічної арифметики потреба у сортуванні масивів є особливо гострою. Як показує практичний досвід, на виконання сортувань у РЛ арифметиці витрачається більша частина машинного часу, яка в декілька разів перевищує час потрібний для зведення подібних. Найбільше проблем, сортування викликає при виконанні операцій множення та ділення.

Швидкість роботи будь-якого алгоритму сортування безпосередньо пов'язана з кількістю порівнянь та кількістю обмінів яку необхідно виконати, при чому обміни займають більше часу [3].

Найпростіший алгоритм сортування, відомий як «бульбашка» відноситься до класу обмінних алгоритмів. Кількість порівнянь в процесі його роботи є постійною і не залежить від попередньої відсортованості масиву[3]. Така властивість даного алгоритму робить його найбільш неефективним для використання в РЛ арифметиці. Це пов'язано з тим, що у більшості випадків потреба у сортуванні виникає при злитті двох РЛ чисел, тобто при об'єднанні повністю впорядкованих масивів. При сортуванні бульбашкою у середньому кількість порівнянь, яку необхідно виконати є величиною, пропорційною до:

$$\frac{n^2 - n}{2}, \qquad (3)$$

де n – кількість елементів у масиві.

На відміну від сортування бульбашкою, у сортуванні вставками кількість порівнянь залежить від початкової відсортованості масиву. У випадку, коли масив є повністю впорядкованим, кількість порівнянь буде дорівнювати n-1. У середньому кількість порівнянь можна описати як величину, що приблизно дорівнює:

$$\frac{n^2}{4}, \qquad (4)$$

де n – кількість елементів у масиві.

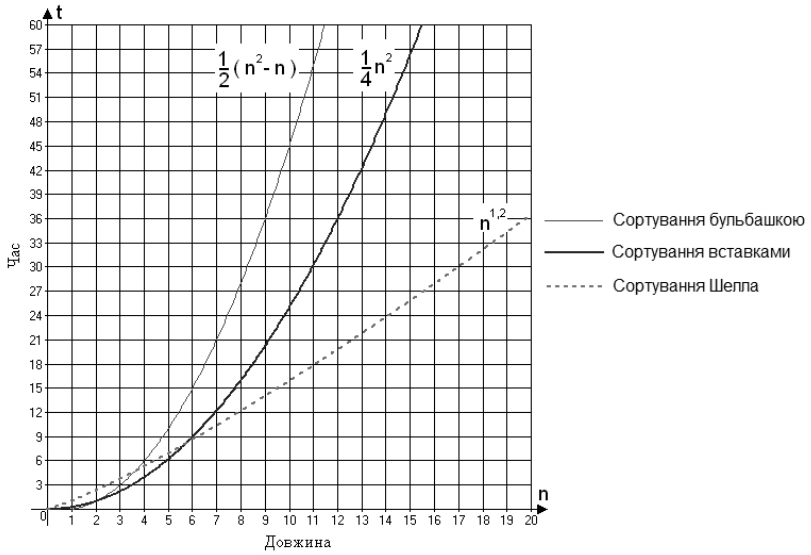

Рис. 1. Порівняння залежностей швидкодії від довжини масиву для різних алгоритмів сортування

Очевидно, що у обох вищезгаданих алгоритмів час виконання сортування є величиною порядку n^2 і тому їх швидкодія при використанні в розрядно-логарифмічній арифметиці буде сильно зростати із збільшенням довжини РЛ числа. Однак алгоритм сортування вставками має перевагу – він працює найменше у випадку повністю відсортованого масиву і найбільше, коли масив відсортований у зворотному порядку. Ця особливість робить його практично ідеальним

алгоритмом для майже повністю впорядкованих масивів [3].

Розвитком ідеї сортування вставками є алгоритм Шелла, який виконує декілька проходів сортування, використовуючи різні за довжиною кроки у спадному порядку. У середньому, час сортування є величиною, пропорційною до $n^{1,2}$, де n – кількість елементів у масиві.

Як видно з графіка, для дуже коротких масивів, сортування бульбашкою може виявитися найбільш швидкодіючим. При збільшенні довжини масиву найбільш ефективним стає сортування вставками. У випадку дуже довгих масивів самим швидкодіючим алгоритмом стає алгоритм Шелла. Наведений графік демонструє ефективність алгоритмів сортування для найбільш поширених випадків їх застосування.

Перевіримо ефективність наведених алгоритмів при їх використанні у специфічній області – при виконанні обчислень у розрядно-логарифмічній арифметиці. В якості тестового завдання обрано розрахунок оберненої матриці Гілберта 11-го порядку методом Крамера при довжині РЛ числа Q = 81. Даний алгоритм обчислень має приблизно: 40 % операцій додавання-віднімання, 40 % множення та 20 % ділення.

Як видно з табл.1, перехід від сортування алгоритмом бульбашки до алгоритму вставками збільшує швидкість обчислень у 5,8 разів. Алгоритм Шелла здебільшого програє сортуванню вставками, і лише у одному випадку зрівнюється з ним за ефективністю роботи. Швидкодія алгоритму Шелла сильно залежить від набору кроків і стає максимальною при значенні першого кроку ≈0,5*Q.(при загальній кількості кроків – 2). Збільшення кількості кроків в даному випадку не призводить до збільшення швидкодії.

Таблиця 1

Час обчислення зворотної матриці Гілберта для різних алгоритмів сортування

Алгоритм	Час обрахунку (сек)								
Бульбашка	335								
Вставками	57								
Шелла	68	72	66	62	60	57	59	61	70
	{1}	{5,1 }	{10,1}	{19,1}	{34,1}	{39,1}	{44,1}	{49,1}	{39,10,1}
	Набір кроків								

Таким чином, найбільш оптимальним серед розглянутих алгоритмів сортування для використання у розрядно-логарифмічній арифметиці є сортування вставками. Алгоритм Шелла, навіть за умови вдалого підбору кроків буде значно програвати йому в універсальності, потребуючи постійного переналагодження при кожній зміні довжини Q РЛ числа.

3. Збільшення швидкодії шляхом зміни структури розрядно-логарифмічного числа

Прискорити операції множення та ділення можна за рахунок такої зміни структури РЛ числа, яка призведе до меншої кількості дій над його компонентами при виконанні обчислень. Один з можливих варіантів полягає у винесенні в формулі (2) старшого розряду за дужки наступний чином:

$$A = p^{n_i}(p^0 + p^{n_{i-1}-n_i} + p^{n_{i-2}-n_i} + \\ +...+ p^{n_2-n_i} + p^{n_1-n_i}). \quad (5)$$

Ми отримали ситуацію коли перший і найстарший елемент РЛ числа є множником усіх інших. В РЛ числі подібної структури елемент p^0 завжди буде присутній, тому цей елемент можна не зберігати в пам'яті, а враховувати його існування в алгоритмах виконання операцій. Інші параметри РЛ числа не змінюється. Подібна зміна структури дозволяє підвищити швидкодію операцій множення та ділення розрядно-логарифмічних чисел у випадку коли множник (дільник) є цілою ступінню двійки. Це пов'язано з тим, що операцію додавання (віднімання) окремих елементів РЛ числа необхідно виконувати не з кожним з них окремо, а лише з першим елементом.

У пам'яті ЕОМ таке число буде представлено у вигляді:

Sign	Q	n_i	$n_{i-1} - n_i$	$n_{i-2} - n_i$	…	$n_3 - n_i$	$n_2 - n_i$	$n_1 - n_i$

Задля перевірки ефекту від запропонованої зміни структури РЛ числа, було створено тестову програму, яка виконує розрахунок 1024 членів геометричної прогресії із знаменником 2, фіксуючи при цьому тривалість виконання всього комплексу обчислень.

Після виконання досліджень, проведених на тестовій програмі отримано наступні результати:

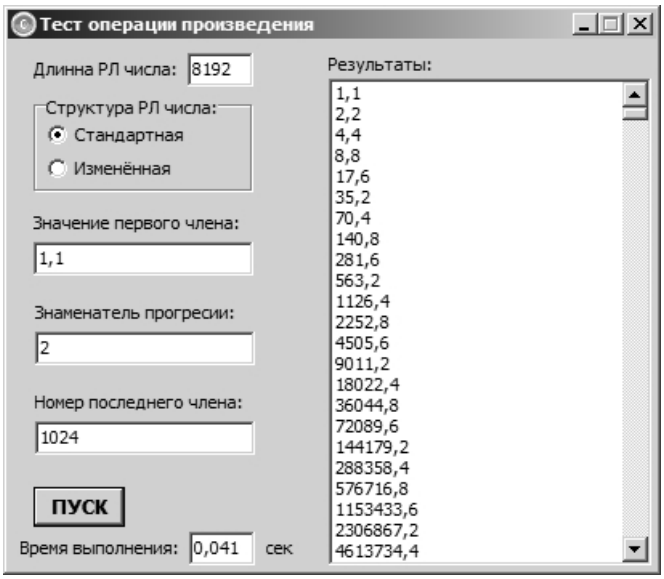

Рис. 2. Вигляд інтерфейсу тестової програми

Як видно з рис. 3, даний метод збільшення швидкодії призводить до прискорення виконання операцій множення приблизно на 30 %. Більш того, певною мірою зменшується залежність швидкості від довжини чисел. Крім того цей метод не призводить до зростання довжини РЛ числа і зберігає зручність перетворення в РЛ число нормального вигляду. Подібну зміну структури РЛ числа доцільно використовувати в задачах по обчисленню геометричних прогресій, факторіалів, тощо.

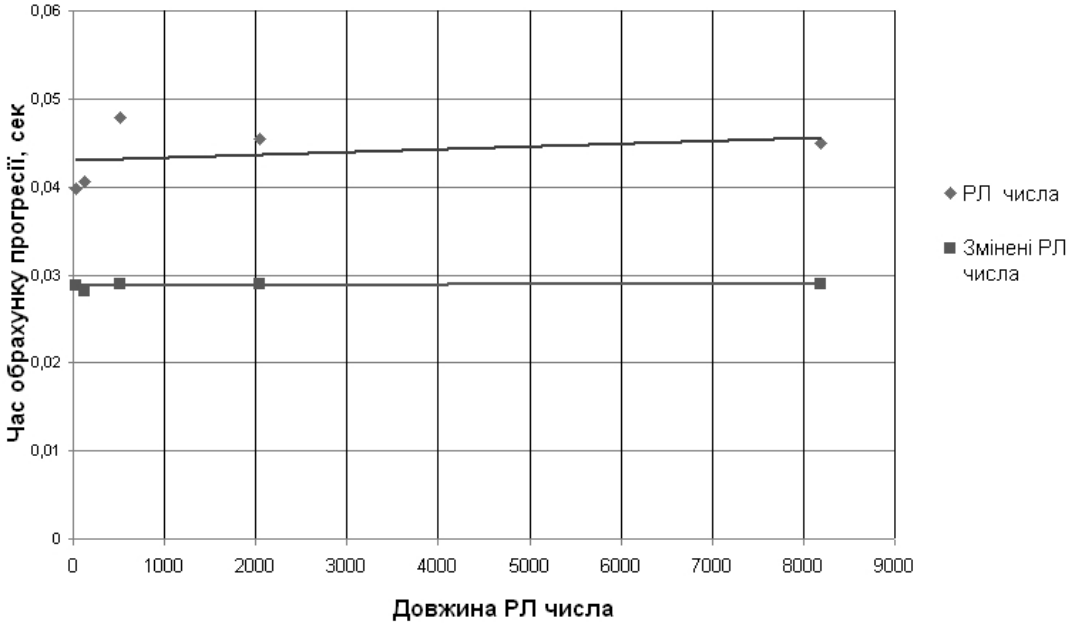

Рис. 3. Час обрахунку геометричної прогресії різними методами в залежності від довжини РЛ числа

Список літератури

1. *Гамаюн В. П.* Моделювання багаторозряднних комп'ютерних систем: Навч. посібник / В. П. Гамаюн – К.: Книжкове видавництво НАУ, 2007. – 112 с.

2. *Кнут Д.* Искусство программирования, том 3. Сортировка и поиск. – 2-е изд. / Д. Кнут – М.: Издательский дом «Вильямс», 2007. – 824 с.

3. *Шилдт Г.* Полный справочник по С. – 4-е изд. / Шилдт. Г. – М.: Издательский дом «Вильямс», 2002. – 704 с.

Науковий керівник – Гамаюн В. П., д-р техн. наук, проф.