17. Хомицька І.Ю. Моделювання статистичних структур функціональних стилів англійської мови / І.Ю. Хомицька // Наукові записки. – Сер.: Філологічні науки (мовознавство). – Кіровоград : РВВ КДПУ ім. В. Винниченка. – 2015. – Вип. 136. – С. 209-213.

18. Хомицька І.Ю. Статистичний аналіз англійських поетичних текстів / І.Ю. Хомицька, В.М. Теслюк // Науковий вісник НЛТУ України : зб. наук.-техн. праць. – Львів : РВВ НЛТУ України. – 2015. – Вип. 25.2. – С. 350-356.

19. Хомицька І.Ю. Модель статистичного аналізу процесу функціонування груп англійських приголосних фонем у системі функціональних стилів / І.Ю. Хомицька, В.М. Теслюк // Науковий вісник НЛТУ України : зб. наук.-техн. праць. – Львів : РВВ НЛТУ України. – 2015. – Вип. 25.3. – С. 364-369.

Хомицкая И.Ю., Теслюк В.Н. **Метод статистического анализа функциональных стилей английского языка на фонологическом уровне**

Проанализирована степень действия стилевого и языкового факторов при дифференциации художественного, разговорного, газетного и научного функциональных стилей английского языка на фонологическом уровне. Доказано, что критерием дифференциации стилей можно считать средние частоты групп согласных фонем. Предложена модель определения существенных и несущественных различий между сравниваемыми текстами, репрезентирующие исследуемые стили. Определена зависимость стилеразграничительной способности групп согласных фонем от позиции фонемы в слове (фонема в начале слова, фонема в конце слова).

Ключевые слова: средняя частота групп согласных фонем, статистические характеристики стилей, позиция фонемы в слове.

Khomytska I.Yu., Teslyuk V.M. The **Method of Statistical Analysis of English Functional Styles on the Phonological Level**

The degree of the effect of style and language factors while differentiating belles – lettres, colloquial, newspaper and scientific styles of the English language on the phonological level has been analyzed. It has been proved that style differentiation criterion can be considered to be mean frequencies of occurrence of groups of consonant phonemes. A model has been suggested to determine essential and unessential differences between the compared texts representing the styles under study. The dependence of style differentiating power of groups of consonant phonemes on the position of a phoneme in a word such as phoneme at the beginning of a word, phoneme at the end of a word is defined.

Keywords: mean frequency of occurrence of groups of consonant phonemes, statistical characteristics of styles, position of a phoneme in a word.

УДК 621.518

Проф. І.Г. Цмоць, д-р техн. наук; доц. Я.П. Кісь, канд. техн. наук; аспір. В.Я. Антонів[1] ***– НУ "Львівська політехніка"***

ЗАСТОСУВАННЯ ГРАФІЧНОГО ПРОЦЕСОРА ДЛЯ ПІДВИЩЕННЯ ШВИДКОДІЇ ПРОЦЕСУ СОРТУВАННЯ ВЕЛИКИХ МАСИВІВ ДАНИХ

Проаналізовано методи та алгоритми паралельного сортування масивів даних та особливості архітектури графічних процесорів GPU. Запропоновано розробку програмних засобів паралельного сортування масивів даних з використанням графічного процесора GPU та програмної моделі CUDA здійснювати на основі комплексного підходу, який охоплює: дослідження, розроблення методів та алгоритмів паралельного сортування великих масивів даних; графові моделі алгоритмів паралельного сортування масивів даних; архітектуру графічного процесора GPU та програмну модель CUDA. Розроблено конкретизований потоковий граф алгоритму сортування методом злиття, який забезпечує виявлення паралелізму та можливість управляти ним. Визначено складність паралельного алгоритму сортування злиттям та його швидкодію.

Ключові слова: паралельне сортування, графічний процесор, комплексний підхід, потоковий граф, злиття.

[1] Наук. керівник: проф. І.Г. Цмоць, д-р техн. наук

Постановка проблеми. Основним шляхом підвищення швидкодії процесу сортування великих масивів даних є розпаралелювання процесу сортування та використання масово-паралельних обчислюваних засобів із великим обсягом пам'яті. До таких засобів належать графічні процесори (GPU – Graphics Processing Unit), які є процесорами класу SIMD (Single Instruction Multiple Data). Особливістю SIMD процесорів є те, що в них одна операція використовується одночасно для опрацювання множини незалежних даних. Для розроблення програмного забезпечення сортування великих масивів даних, частина якого працює на CPU (центральний процесор), а частина на GPU доцільно використати кросплатформову систему компіляції та виконання програм CUDA.

Отже, розроблення паралельних алгоритмів і програмних засобів для сортування великих масивів даних на графічному процесорі GPU є актуальною задачею.

Аналіз останніх досліджень і публікацій. Задачі розроблення паралельних методів, алгоритмів і програмних засобів для сортування великих масивів даних з використанням графічних процесорів GPU розглянуто у працях **[**1-3**]**. Аналіз методів паралельного сортування масивів даних показує **[**3-9**]**, що для реалізації графічних процесорів GPU найбільше підходять методи сортування чисел підрахунком, витісненням, вставкою та злиттям. Порівняння цих методів показує, що алгоритми паралельного сортування даних злиттям є структурованими, однорідними та орієнтованими на паралельно-конвеєрну.

З публікацій **[**1-3, 10, 11**]** випливає, що для ефективного сортування великих масивів даних за допомогою графічних процесорів GPU потрібно розробити моделі паралельного сортування масивів даних у вигляді графів. Проте у цих публікаціях не висвітлено питання розроблення моделей паралельного сортування масивів даних у вигляді графів і їх оцінювання.

Невирішені частини проблеми. У публікаціях і дослідженнях недостатньо уваги приділено питанням комплексного підходу до розроблення програмних засобів сортування великих масивів даних на графічних процесорах GPU.

Метою дослідження є розроблення з використанням комплексного підходу програмних засобів для швидкого сортування великих масивів даних на базі графічного процесора GPU та програмної моделі CUDA.

Основні результати дослідження. Розробку високоефективних програмних засобів паралельного сортування масивів даних з використанням графічного процесора GPU та програмної моделі CUDA можна забезпечити комплексним підходом, який охоплює:

- дослідження, розроблення методів і алгоритмів паралельного сортування великих масивів даних;
- графові моделі алгоритмів паралельного сортування масивів даних;
- архітектуру графічного процесора GPU;

- програмну модель CUDA.

Переважна частина методів сортування масивів даних ґрунтується на однотипних базових елементарних операціях порівняння двох чисел та їх перестановки відповідно до формули:

$$y=\begin{cases}0, коли\ a_1 \le a_2\\ 1, коли\ a_1 > a_2\end{cases},\ b_1=\begin{cases}a_1, коли\ y=1\\ a_2, коли\ y=0\end{cases},\ b_2=\begin{cases}a_1, коли\ y=0\\ a_2, коли\ y=1\end{cases}, \quad (1)$$

де: y – результат порівняння двох чисел; a_1, a_2 – числа, які порівнюються; b_1, b_2 – виходи більшого та меншого відсортованих чисел. Обчислювальна складність методів сортування масивів чисел може бути в межах від $R\ (O)=N^2$ до $R\ (O)=N\log_2 N$ елементарних базових операцій, де N – кількість чисел у масиві. Сортування великих масивів даних вимагає виконання великої кількості елементарних базових операцій та значного часу сортування. Основним шляхом зменшення часу сортування великих масивів даних є розпаралелювання та конвеєризація процесу сортування даних.

Виявити паралелізм і використати його при реалізації на комп'ютерних засобах забезпечує графічне відображення алгоритму сортування даних у вигляді потокового графу $F=(\Phi, \Gamma)$, де $\Phi=\{\Phi_1, \Phi_2,\ldots, \Phi_i,\ldots, \Phi_n\}$ – множина функціональних операторів, Γ – закон відображення зв'язків між функціональними операторами. Для відображення алгоритмів сортування використовуються функціональні оператори Φ_i, які відповідають базовим елементарним операціям порівняння та перестановки згідно з формулою (1). При відображенні алгоритму сортування у вигляді потокового графу здійснюється розподіл на всіх функціональних Φ_i за ярусами отже, що в k-му ярусі розміщені функціональні оператори, які залежать від операторів попередніх ярусів і не залежать від операторів наступних ярусів. У середині кожного ярусу функціональні оператори між собою не мають з'єднань.

Для реалізації сортування великих масивів даних пропонують використати графічний процесор GPU, який має p конвеєризованих обчислювальних ядер [1, 3]. Ефективне використання графічного процесора GPU передбачає розбиття масиву даних на блоки розміром $2p$, які будуть опрацьовуватися обчислювальними ядрами паралельно.

Методи сортування, що ґрунтуються на базових елементарних операціях порівняння двох чисел та їх перестановки відрізняється один від одного способом вибору пар даних для виконання базових елементарних операцій. Особливістю графічного процесора GPU є те, що він орієнтований на виконання однієї команди над $2p$ даними, які одночасно надходять на p обчислювальних ядер. Для ефективного сортування масивів даних на базі графічного процесора GPU необхідно вибрати методи сортування, структура яких орієнтована на паралельно-конвеєрну реалізацію. Таким методом є сортування злиттям.

В основі алгоритмів сортування методом злиття лежить базова операція об'єднання двох упорядкованих масивів $\{a_{1i}\}_{i=1}^{2^{i-1}}$ та $\{a_{2i}\}_{i=1}^{2^{i-1}}$ в один упорядкований масив $\{b_{1i}\}_{i=1}^{2^i}$. На початку сортування вхідний масив чисел $\{a_j\}_{j=1}^{N}$ розбивається на $N/2$ упорядкованих масивів довжиною одиниця. Внаслідок виконання першої базової операції формуються $N/4$ впорядкованих масивів довжиною два. Кількість типів базових операцій, необхідних для сортування масиву із N чисел, визначають за формулою

$$k=\log_2 N. \quad (2)$$

Кожна (i+1)-та базова операція (i=1, …, k) паралельного алгоритму сортування злиттям реалізується на базі трьох i-х базових операцій. Кількість елементарних операцій порівняння двох чисел та їх перестановки для сортування масиву із N дорівнює

$$R(0)=\sum_{i=1}^{k} 3^{i-1}\frac{N}{2^i}. \quad (3)$$

Для ефективної реалізації алгоритму сортування великих масивів даних методом злиття на графічному процесорі GPU необхідно цей алгоритм подати у вигляді конкретизованого потокового графу. Особливістю конкретизованого потокового графу сортування великих масивів даних методом злиття є його орієнтація на архітектуру графічного процесора GPU.

Розроблення конкретизованого потокового графу сортування великих масивів даних методом злиття можна розбити на такі чотири етапи:

- декомпозиція алгоритму сортування на базові та елементарні операції (функціональні оператори);
- проектування комунікацій (обмінів даними) між функціональними операторами;
- укрупнення функціональних операторів;
- планування процесу сортування.

Результатом першого етапу є потоковий граф алгоритму сортування даних методом злиття. Внаслідок виконання трьох наступних етапів отримуємо конкретизований потоковий граф алгоритму сортування злиттям, який є лінійною проекцією потокового графу на горизонтальну вісь X. У цьому випадку укрупнення операцій здійснюється об'єднанням каналів передачі даних, визначаються затримки, перестановки та формується управління процесом сортування. Розроблений конкретизований потоковий граф для графічного процесора GPU наведено на рис. 1, де $\Phi_{МЗП}$ – макрооператор затримки та перестановки, $\Phi_{МУ}$ – макрооператор управління, Φ_1, Φ_p – функціональні оператори попарного порівняння та перестановки даних.

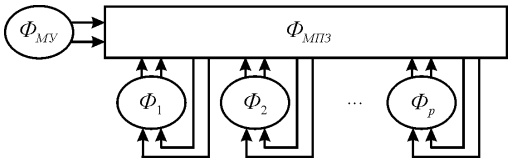

Рис. 1. Конкретизований потоковий граф сортування масиву даних

Конкретизований потоковий граф сортування орієнтований на архітектуру графічного процесора GPU. За допомогою макрооператора затримки та перестановки $\Phi_{МЗП}$ та макрооператора управління $\Phi_{МУ}$, визначаються величини затримок, перестановки даних і здійснюється планування процесу сортування.

Реалізація конкретизованого потокового графу сортування масиву даних методом злиття відбувається по тактам, кількість яких визначають за формулою

$$T = (N-1)\frac{N}{2p}, \quad (4)$$

де: N – кількість даних у масиві, p – кількість обчислювальних ядер у процесорі.

Для сортування даних методом злиття використано багатоядерний графічний процесор Nvidia GT 610 з підтримкою архітектури і програмної моделі CUDA. Графічний процесор Nvidia GT 610 має такі характеристики:

- кількість ядер – 48;
- тактова частота графічного процесора – 1620;
- тактова частота процесора – 810 МГц;
- швидкодія пам'яті – 1,8 Гбіт/с;
- об'єм пам'яті – 1024 Мб;
- інтерфейс пам'яті – 64-біт DDR3;
- максимальна смуга пропускання пам'яті – 14,4;
- програмне середовище – CUDA
- шина – PCI-E 2.0.

Для розроблення програмного забезпечення використовували програмні середовища розробки Nvidia CUDA 5.5, MS Visual Studio 2010 та MS SQL Server Management Studio 2012. Особливістю CUDA є те, що вона дає розробникові можливість на свій розсуд організовувати доступ до набору інструкцій графічного процесора GPU й управляти його пам'яттю, організовувати на ньому складні паралельні обчислення. Графічний процесор з підтримкою CUDA стає потужною програмованою відкритою архітектурою подібно до сьогоднішніх центральних процесорів.

Мовами реалізації сортування великих масивів даних на графічному процесорі Nvidia GT обрано C/C++, яка є мовою програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Особливостями мови C/C++ є: швидкодія; масштабованість; можливість роботи на низькому рівні з пам'яттю, адресами, портами; створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація та обчислення на етапі компіляції з використанням шаблонів; підтримка різних стилів та технології програмування.

Для розроблення програми сортування великих масивів даних на графічному процесорі Nvidia GT обрано модульне програмування, яке передбачає реалізацію складних програм за ієрархічним принципом на базі невеликих програмних блоків чи модулів. Програмні засоби складаються з таких модулів: попереднього оброблення даних transfortData; підготовки даних для сортування preparationData; сортування даних parallelSearchGPU і parallelSortGPU; збереження даних SaveData.

На рис. 2 наведено користувацький інтерфейс доступу до модулів, які реалізують технологію паралельного сортування даних.

Використання для сортування великих масивів даних методом злиття графічного процесора Nvidia GT 610 забезпечує зменшення часу сортування приблизно у 10 разів, порівняно з використанням тільки центрального процесора CPU.

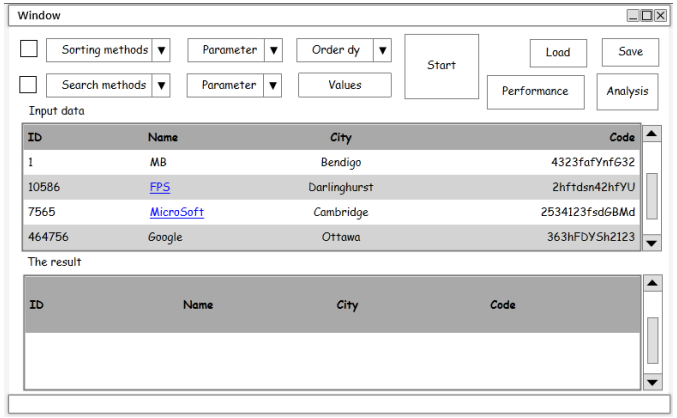

Рис. 2. Користувацький інтерфейс доступу до модулів паралельного сортування даних

Висновки:

1. Розробку програмних засобів паралельного сортування масивів даних з використанням графічного процесора GPU та програмної моделі CUDA можна забезпечити комплексним ом який охоплює: дослідження, розроблення методів та алгоритмів паралельного сортування великих масивів даних; графові моделі алгоритмів паралельного сортування масивів даних; архітектуру графічного процесора GPU та програмну модель CUDA.
2. Сортування масивів даних методом злиття ґрунтується на однотипних базових елементарних операціях порівняння двох чисел та їх перестановки, кількість яких дорівнює $R(0) = \sum_{i=1}^{k} 3^{i-1}\frac{N}{2^i}$.
3. Для розроблення програмних засобів паралельного сортування масивів даних з використанням графічного процесора GPU та програмної моделі CUDA алгоритм сортування необхідно подати у вигляді потокового графу, який забезпечить виявлення паралелізму та можливість управляти ним для знаходження оптимальних просторово-часових рішень.
4. Кількість тактів, необхідних для сортування масивів даних методом злиття в графічному процесорі GPU, дорівнює $T = (N-1)\frac{N}{2p}$.

Література

1. Боресков А.В. Параллельные вычисления на GPU / А.В. Боресков и др. // Архитектура и программная модель CUDA. – М. : Изд-во Московского ун-та, 2012. – 336 с.

2. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб. : Изд-во "БХВ-Петербург", 2002. – 608 с.

3. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем / В.П. Гергель. – М. : Изд-во Московского ун-та, 2010. – 544 с.

4. Кнут Д. Искусство программирования. – Т. 3: Сортировка и поиск / Д. Кнут. – Изд. 2-ое, [перераб. и доп.]. – М. : Изд-во "Наука". 2000. – 832 с.

5. Кормен Т. Алгоритмы: построение и анализ : пер. с англ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест; под ред. А. Шеня. – М. : МЦНМО БИНОМ. Лаборатория знаний, 2004. – 960 с.

6. Левитин А.П. Алгоритмы: введение в разработку и анализ : пер. с англ. / А.П. Левитин. – М. : Изд. дом "Вильямс", 2006. – 576 с.

7. Лорин Г. Сортировка и системы сортировки / Г. Лорин. – М. : Изд-во "Мир". 1983. – 384 с.
8. Макконелл Дж. Основы современных алгоритмов / Дж. Макконелл. – Изд. 2-ое, [перераб. и доп.]. – М. : Изд-во "Техносфера", 2004. – 368 с.
9. Мельничук А.С. Аналіз методів сортування масиву чисел / А.С. Мельничук, С.П. Луценко, Д.С. Громовий, К.В. Трофимова // Технологический аудит и резервы производства : сб. науч. тр. – 2013. – № 4/1(12). – С. 37-40.
10. Немнюгин С.А. Параллельное программирование для многопроцессорных систем / С.А. Немнюгин, О.Л. Стесик. – СПб. : Изд-во "БХВ-Петербург", 2002. – 400 с.
11. Prots'ko I. Algorithm of efficient computation DST using cyclic convolutions / I. Prots'ko, V. Teslyuk // Wseas transactions on signal processing. – 2014. – Vol. 10. – Pp. 278-288.

Цмоць И.Г., Кись Я.П., Антонив В.Я. **Применение графического процессора для повышения быстродействия сортировки больших массивов данных**

Проанализированы методы и алгоритмы параллельной сортировки массивов данных и особенности архитектуры графических процессоров GPU. Предложено разработку программных средств параллельной сортировки массивов данных с использованием графического процессора GPU и программной модели CUDA осуществлять на основе комплексного подхода, который включает: исследования, разработку методов и алгоритмов параллельной сортировки больших массивов данных; графовые модели алгоритмов параллельной сортировки массивов данных; архитектуру графического процессора GPU и программную модель CUDA. Разработан конкретизированный потоковый граф алгоритма сортировки методом слияния, который обеспечивает обнаружение параллелизма и возможность управлять им. Определены сложность параллельного алгоритма сортировки слиянием и его быстродействие.

Ключевые слова: параллельная сортировка, графический процессор, комплексный подход, потоковый граф, слияние.

Tsmots I.G, Kis Ya.P, Antoniv V.Ya. **Application of Graphic Processor to Improve Sorting Performance of Large Data Sets**

Some methods and algorithms for parallel sorting data sets and architectural features of graphics processor GPU are analysed. We suggest software development for parallel sorting data sets using graphics processor GPU and programming model CUDA based on a comprehensive approach, which includes the following: research and development of methods and algorithms for parallel sorting large data sets; graph models of algorithms for parallel sorting data sets; architecture of graphics processor GPU and programming model CUDA. We have also developed the concretized flow graph of algorithm of merge sort that provides parallelism detection and the ability to manage it. The complexity of algorithm of parallel merge sort and its performance are determined.

Keywords: parallel sorting, graphics processor, comprehensive approach, flow graph, merge.

УДК 681.3.05:004.056.5 ***Проф. Ю.І. Грицюк, д-р техн. наук – НУ "Львівська політехніка"; здобувач П.Ю. Грицюк, магістр – НЛТУ України, м. Львів***

МЕТОДИ І ЗАСОБИ ГЕНЕРУВАННЯ Q_P-МАТРИЦЬ ФІБОНАЧЧІ – КЛЮЧІВ ДЛЯ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕНЬ

Розглядаються особливості ефективного генерування Q_p-матриць Фібоначчі, які можуть використовуватися як ключі (де)шифрування для багатораундової матричної криптографічної системи перетворення інформації. З'ясовано, що основна проблема багатораундової матричної Афінної криптосистеми полягає у генеруванні множини звичайних і обернених матриць – ключів (де)шифрування інформації, елементами яких мають бути цілі числа. Розроблено процедуру генерування множини Q_p-матриць Фібоначчі, яка за відомими значеннями степені матриці (n) та p-чисел Фібоначчі дає змогу отримувати відповідну множину ключів (де)шифрування інформації, здійснювати їхнє розширення для кожного раунду, що забезпечує не тільки ефективний спосіб їх утворення та зберігання, але й створює зручність при передаванні каналами зв'язку.

Ключові слова: захист інформації, шифрування/дешифрування інформації, числа Фібоначчі, Q_p-матриць Фібоначчі, криптографічна система, матричні Афінні перетворення, багатораундова матрична криптографічна система.

Вступ. У роботі **[**3**]** було розглянуто особливості побудови надійної криптосистеми захисту інформації, яка поєднує матричні Афінні перетворення, багатораундові дії з різними ключами, а також перестановні алгоритми, що загалом значно підвищує її криптостійкість до брутальних атак. Математично описано алгоритм (де)шифрування інформації за допомогою багатораундової матричної Афінної перестановної криптосистеми з різними ключами шифрування на кожному раунді. Також у цій роботі було зазначено, що, порівняно з іншими методами захисту, класична криптографія гарантує захист інформації тільки за умов, якщо використано ефективний криптографічний алгоритм, а також дотримані умови секретності та цілісності ключів шифрування.

Однак, у матричних Афінних перетвореннях **[**3, розд. 1**]** основна проблема полягає у генеруванні множини матриць $\overline{\overline{A}} = [\overline{A}_i = [a_{ij},\ j = \overline{1,n}], i = \overline{1,n}]$ – ключів шифрування, елементами яких є спеціально підібрані цілі числами з діапазону $1 \le a_{ij} < m$ (де m – кількість символів алфавіту), а також НСД(a, m) = 1, де $a = \det(\overline{\overline{A}}) \bmod m$ – визначник матриці $\overline{\overline{A}}$ за модулем m. Є також деякі питання щодо генерування й стовпців $\overline{B} = [b_i, i = \overline{1,n}]$ – ключів додаткового коригування вже зашифрованого повідомлення, елементами яких є цілі числами з діапазону $1 \le b_i < m$. Водночас, для отримання зворотних матриць $\overline{\overline{A}}' = [\overline{A}_i' = [a_{ij}',\ j = \overline{1,n}], i = \overline{1,n}]$ – ключів дешифрування та зворотних стовпців $\overline{B}' = [b_i', i = \overline{1,n}]$ – ключів коригування потрібно виконати деяку послідовність дій, які детально описано в зазначеній вище роботі.

Якщо ж використовувати багатораундову матричну Афінну криптосистему **[**див. 3, розд. 3**]**, то на кожному раунді криптографічних перетворень (кількість яких може бути від 4 до 16 чи 24) виникає потреба у різних матричних ключах, тобто потрібно вирішувати питання розширення ключів для кожного раунду. Окрім цього, оскільки розмір цих матриць (n) може бути різним (мінімальний 32×32, нормальний 128×128 чи 256×256, надмірний 1048×1048 та більше), а кількість раундів шифрування – великою (32, 48, 64, …), то виникає проблема не тільки у їх зберіганні, але й передачі цих ключів каналами зв'язку з кожним повідомленням. А, як відомо з **[**4, 9**]**, розмір зашифрованого повідомлення практично немає відрізнятися від вхідного повідомлення. Водночас, передані з повідомленням ключі шифрування не мають викликати в криптоаналітиків підозри у цілісності зашифрованого повідомлення.

Отже, основна проблема багатораундової матричної Афінної криптосистеми полягає у генеруванні множини звичайних і обернених матриць – ключів шифрування/дешифрування інформації, елементами яких мають бути цілі числа, розширенні ключів для кожного раунду, а також у ефективній системі їх зберігання та передаванні каналами зв'язку. Для її вирішення пропонуємо використовувати Q_p-матриці, елементами яких є p-числа Фібоначчі [7, 8].

Об'єкт дослідження – матричні ключі (де)шифрування та їх розширення для багатораундової криптографічної системи перетворення інформації.