УДК 621.3

М. Черкаський, Саід Садек Абдалла
Національний університет "Львівська політехніка",
кафедра електронно-обчислювальних машин

СКЛАДНІСТЬ БЛОК-СХЕМ ПРОГРАМ СОРТУВАННЯ

Розглядаються характеристики складності блок-схем програм сортування, синтезованих з використанням декількох відомих алгоритмів. Досліджується залежність часової та об'єктної складностей від структурної. Показано, що зменшення структурної складності супроводжується збільшенням часової складності.

It is examined descriptions of complexity of block-diagram of the routine sorting, synthesized with the use of a few known algorithms. Explore dependence of time complexity from structural complexity. It is shown, that the diminished structural complexity is accompanied by gain timecomplexity.

Вступ

Оцінка програмного продукту є чи не актуальнішою проблемою сучасного стану теорії алгоритмів. Підкреслимо у зв'язку з цим такі моменти.

- програма є складовою будь-якої моделі алгоритму. У моделях абстрактних алгоритмів, наприклад, в машині Тюрінга, нормальних алгоритмах Маркова та інших, крім програми для її реалізації використовують також технічні засоби. В машині Тюрінга – це стрічка, головка, механізм пересування головки. В нормальних алгоритмах Маркова – це розпізнавачі і перетворювачі. Але ці технічні (апаратні) засоби не декларуються у визначеннях моделей;
- моделі комп'ютерних алгоритмів у своєму складі передбачають наявність апаратних і програмних засобів [8];
- будь-який продукт, зокрема і програми, оцінюють за деяким набором характеристик. Для алгоритмів, програм та архітектурних комп'ютерних побудов таким набором є характеристики складності;
- сьогодні найбільшу увагу під час розроблення й дослідження програм приділено мінімізації її часової складності. Це сприяє, у загальному випадку, зростанню продуктивності комп'ютерної реалізації задач;
- крім продуктивності виконання задачі, істотне значення для оцінки програми мають також характеристики складності, які впливають на час розробки програми, – продуктивність проектування. До таких характеристик належить об'єм програми (кількість команд), структура програми;
- отже, оцінка програмного продукту зводиться до визначення та підрахунку значень його характеристик складності.

Постановка проблеми

Програмний продукт можна подати у декількох формах. Найпоширенішими є блок-схема програми та програма мовою високого рівня. Ці дві форми програмного продукту є абстрактними неформальними алгоритмами з інтегрованим кроком. Вони тотожні за змістом виконуваних операцій.

З достатньо великим ступенем достовірності можна припустити, що ці дві форми подання програми надають розробнику найбільші можливості для її удосконалення за характеристиками складності. Вони порівняно з алгоритмом у вигляді функціональної залежності володіють всіма

параметрами алгоритму, зокрема правилами початку і закінчення, структурою зв'язків, що задають послідовність операцій перетворення та пересування даних.

З іншого боку, на відміну від асемблеру, блок-схема та програма мовою високого рівня значно менше (а блок-схема зовсім) не залежать від типу комп'ютера, на якому передбачено виконання задачі.

Отже, блок-схема програми та програма мовою високого рівня найбільшою мірою відповідають сформульованим умовам дослідження та оцінювання характеристик складності.

Тепер відповімо на запитання, яка з двох форм подання програми краща для аналізу складності. На користь блок-схеми свідчать:

- незалежність від мови програмування;
- краще сприйняття і розуміння графічних об'єктів порівняно з текстовою послідовністю команд;
- наявність видимої множини зв'язків між вершинами блок-схеми;
- можливість переходу від блок-схеми до математичних об'єктів – графів і матриць.

На користь програми, записаної мовою високого рівня, свідчить лише те, що вона є кінцевим результатом розроблення, тоді як блок-схема є проміжним результатом і часто в процесі синтезу взагалі не використовується. Але для аналізу складності вона є необхідною.

Отже, об'єктом досліджень для аналізу й оцінювання складності програмного продукту доцільно обрати блок-схему програми.

Формулювання задачі

Блок-схема програми еквівалентна абстрактному неформальному алгоритму з інтегрованим кроком. Такі алгоритми аналізують за трьома характеристиками складності: часовою, ємнісною та кількістю команд, що містяться у програмі. Часову складність можна визначати кількістю виконуваних вершин блок-схеми, рахуючи їхнє повторення у циклах. Кожна вершина відповідає деякій інтегрованій операції, яка може складатися з декількох простіших операцій. У загальному випадку інтегровані операції не рівноцінні за часовою складністю. Тому за підрахунком часової складності за блок-схемою можна отримати лише наближене уявлення про сумарне значення елементарних операцій програми. Краще перенести на блок-схему значення часової складності, отримане під час аналізу алгоритму у вигляді функціональної або графічної залежності.

Ємнісну складність визначають за кількістю комірок пам'яті, потрібних для розв'язання задачі. Найчастіше ємнісну складність приймають такою, що дорівнює розміру входу.

Кількість команд, що містяться у програмі, іноді математично не строго називають програмною складністю. Ця оцінка дає уявлення про об'єм програми і лише опосередковано зв'язана із інформаційним вмістом програмного продукту, збільшенням продуктивності комп'ютерного розв'язання задач і продуктивністю проектування. Отже, для дослідження інформаційного вмісту програмного продукту потрібні інші моделі.

Псевдо SH-модель

Таку модель блок-схеми програми можна побудувати подібно до SH-моделі – моделі комп'ютерного алгоритму [9, 10]. Вона імітує роботу апаратно-програмних засобів. Апаратно-програмні засоби наведено обраною для заданого алгоритму множиною елементарних перетворювачів та множиною з'єднань між перетворювачами. Функціонування цієї моделі розглядають за часовою діаграмою її роботи.

Блок-схема програми має багато спільного з SH-моделлю. Вона частково моделює алгоритм, має всі вісім параметрів алгоритму, наведена обраною для розв'язання деякого завдання множиною вершин та множиною з'єднань між вершинами. Операційні та умовні вершини і зв'язки між ними можна ототожнити з елементарними перетворювачами та з'єднаннями SH-моделі. Отже, модель блок-схеми програми можна розглядати як подібну до SH-моделі. Назвемо її псевдо SH-моделлю.

Крім спільних рис, SH-модель та псевдо SH-модель мають істотні розбіжності. По-перше, псевдо SH-модель не має у своєму складі програми, вона сама є моделлю програми. По-друге, операційні та умовні вершини не відображають апаратного змісту елементарних перетворювачів.

Тепер за аналогією з SH-моделлю дамо визначення псевдо SH-моделі. Псевдо SH-модель – це шістка:

$$< D, Q, q_0, q_f, G, M >,$$

де $D-$ кінцева множина символів зовнішнього алфавіту; Q – кінцева множина станів псевдо SH-моделі; q_0 *і* q_f – початковий і кінцевий стани, $q_0, q_f \in Q$; M – множина комірок пам'яті, необхідних для реалізації програми; G – конфігурація операційних та умовних вершин і зв'язків між ними

$$G = (H, L),$$

де H – множина операційних та умовних вершин блок-схеми; L – множина зв'язків між вершинами блок-схеми.

Порівняємо SH-моделі та псевдо SH-моделі. Нагадаємо, що SH-модель – це сімка:

$$< D, Q, q_0, q_f, G, P, M >$$

У визначенні псевдо SH-моделі відсутня P-програма. Граф G псевдо SH-моделі подає дві множини – вершин і зв'язків, на відміну від множин елементарних перетворювачів і з'єднань між ними у SH-моделі. Різниця тут у тому, що елементарний перетворювач безпосередньо здійснює операцію перетворення даних. Він є неподільною одиницею часової складності алгоритму, тоді як вершина блок-схеми програми є тільки інструкцією про те, що треба робити над даними. Вона не є одиницею часової складності програми.

Властивості псевдо SH-моделі

Псевдо SH-модель має чотири властивості: дискретність, детермінованість, масовість, ієрархічність. Від неформальних алгоритмів її відрізняє наявність властивості "ієрархічність". Від SH-моделі – відсутність властивості "елементарність". Крок блок-схеми програми є інтегрованим.

Характеристики псевдо SH-моделі

Псевдо SH-моделі має чотири характеристики складності: часову, об'єктну, ємнісну і структурну. Перші три є технічними, остання – інформативна.

Для псевдо SH-моделі характеристику "часова складність" безпосередньо не можна визначити через те, що крок програми не є елементарним. Але значимість цієї характеристики складності є великою, її вплив на продуктивність розв'язання задач у архітектурному плані є вирішальним. Тому значення часової складності для псевдо SH-моделі набуває підрахунок обчислювальних операцій неформальних алгоритмів, які є основою для побудови блок-схеми програми. Підрахунок передбачає операції у циклі.

Об'єктна складність еквівалентна апаратній складності SH-моделі. Вона є сумою всіх виконуваних вершин блок-схеми програми, але не є інформативною характеристикою. Назву "об'єктна складність" введено для того, щоб відрізнити її від інформативної характеристики SH-моделі "програмна складність ".

Ємнісна складність дорівнює потужності множини комірок пам'яті, необхідних для реалізації програми.

Структурну складність визначають за ступенем нерегулярності матриці суміжності, що відображає блок-схему програми. Вона є єдиною інформаційною характеристикою псевдо SH-моделі. Намагання збільшити структурну складність програми істотно впливає на збільшення трудомісткості розроблення програмного продукту. Структурна складність програми безпосередньо визначає час проектування.

Структурну складність визначають за такою послідовністю:

1. Перетворення блок-схеми алгоритму в еквівалентний орієнтований граф.
2. Стиснення графу алгоритму.
3. Побудова матриці суміжності.
4. Визначення структурної складності.

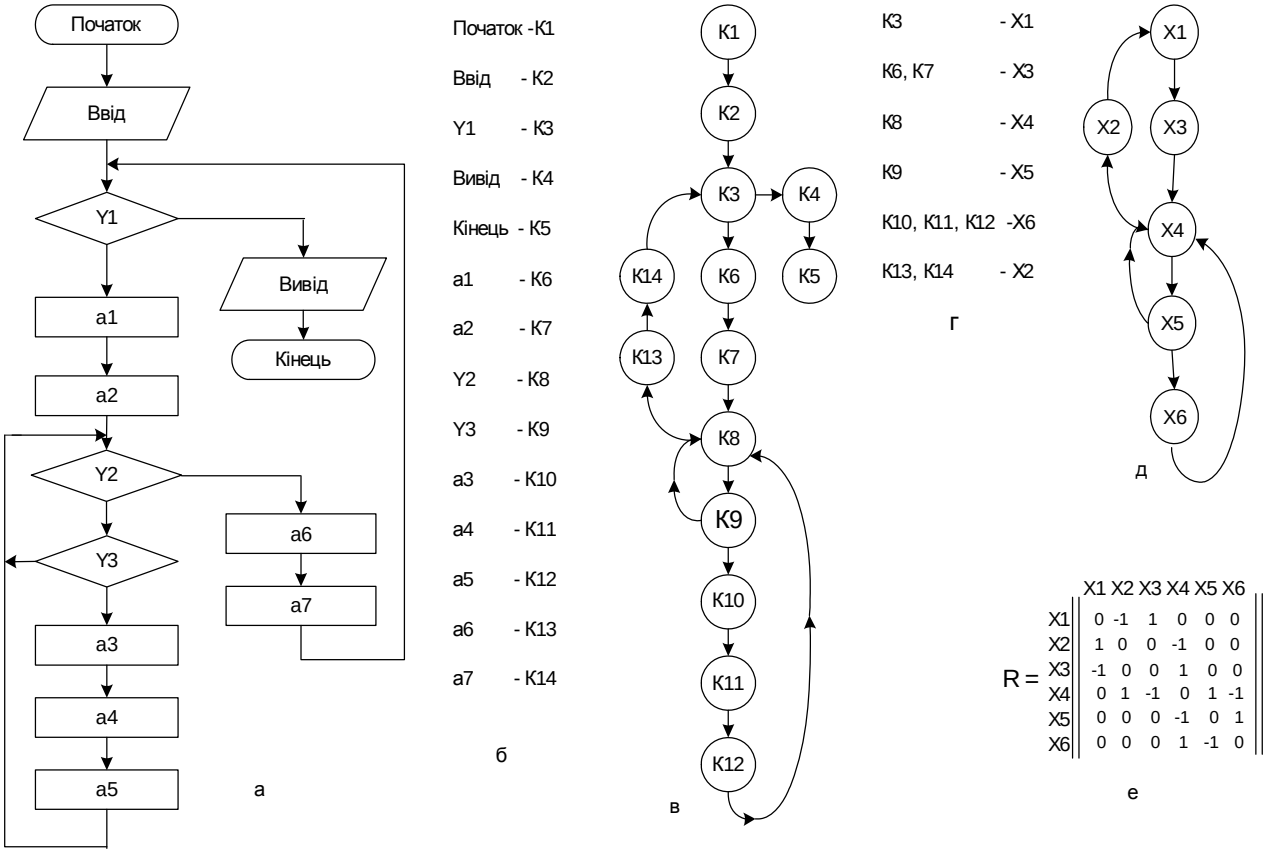

Рис. 1. Приклад розрахунку структурної складності псевдо SH-моделі

Блок-схема програми (рис. 1, *а*) перетворюється в граф (рис. 1, *в*). Конфігурація вершин К13 і К14; К6 і К7; К10, К11, К12. повторюється, тому їх потрібно вилучити з розрахунків та кожну групу вершин замінити одною. Така заміна не веде до зменшення об'єму операцій, не вносить помилки в алгоритм обчислень. Об'єднання вершин блок-схема програми не заперечує тези про інтегрованість кроку. Під час визначення нерегулярності програми доцільно також обмежитися лише вершинами, що утворюють правило безпосереднього перероблення і не беручи до уваги стандартні вершини початку і закінчення та вводу–виводу.

Після об'єднання і вилучення вершин отримуємо стиснутий граф (рис. 1, *д*), для якого будуємо матрицю суміжності (рис. 1, *е*). Враховуючи, що матриця суміжності квадратна, симетрична, має нульову головну діагональ, визначення структурної складності можна обмежити підрахунком ступеня нерегулярності лише однієї трикутної частини матриці без головної діагоналі. Тому формулу для розрахунку подамо у вигляді:

$$S = -FLog_2 \frac{F}{\frac{n^2}{2} - n} = -FLog_2 \frac{F * 2}{n(n-2)}.$$

Для розглянутого прикладу отримаємо значення структурної складності:

$$S = -7Log_2 \frac{7 * 2}{6 * 4} = \text{-7*(-0,79)} = 5{,}53$$

Складність сортування

З практики розробки алгоритмів відомо, що характеристики складності взаємозалежні. Якщо потрібно зменшити часову складність, то треба очікувати зростання іншої, наприклад, структурної, чи декількох інших характеристик складності. Дослідимо цю ситуацію, обравши з цією метою відповідну задачу. Серед багатьох практично важливих задач, які б у максимальному ступені задовольняли потреби дослідження алгоритмічних проблем, є задача сортування. Процедури сортування створюють унікальне поле для дослідження завдяки таким ознакам:

- кількість алгоритмів сортування становить декілька десятків;
- вони оперують масивами цілих додатних чисел, що є ознакою точних класичних алгоритмів;
- фундаментальні дослідження їхньої часової складності проведено відомими фахівцями з теорії алгоритмів [1–7]
- часова складність змінюється в достатньо широких границях від $O(N^2)$ до $O(N)$, де N розмір входу (кількість вхідних даних).

Наша задача – визначити структурну складність і показати її залежність від інших характеристик. Проаналізуємо дві блок-схеми програм сортування, приділяючи основну увагу взаємозалежності технічної характеристики – часової та інформативної – структурної складності. Оберемо алгоритми, які мають такі наближені значення часової складності: $O(N^2)$, $O(NLog_2N)$. Це алгоритми: сортування простими включеннями та сортування з розділенням (рекурсивний спосіб). Для побудови блок-схем скористуємося описом програм і даними досліджень часової складності, наведеними Н. Віртом [4, 5].

Алгоритм сортування простими включеннями. Це один з простіших алгоритмів. Його опис, приклад, програма, аналіз кількості операцій порівняння та пересилок наведено на с. 78–81 [4]. На цій основі нами створено блок-схему програми, її граф, стиснений граф та матрицю суміжності (рис. 2).

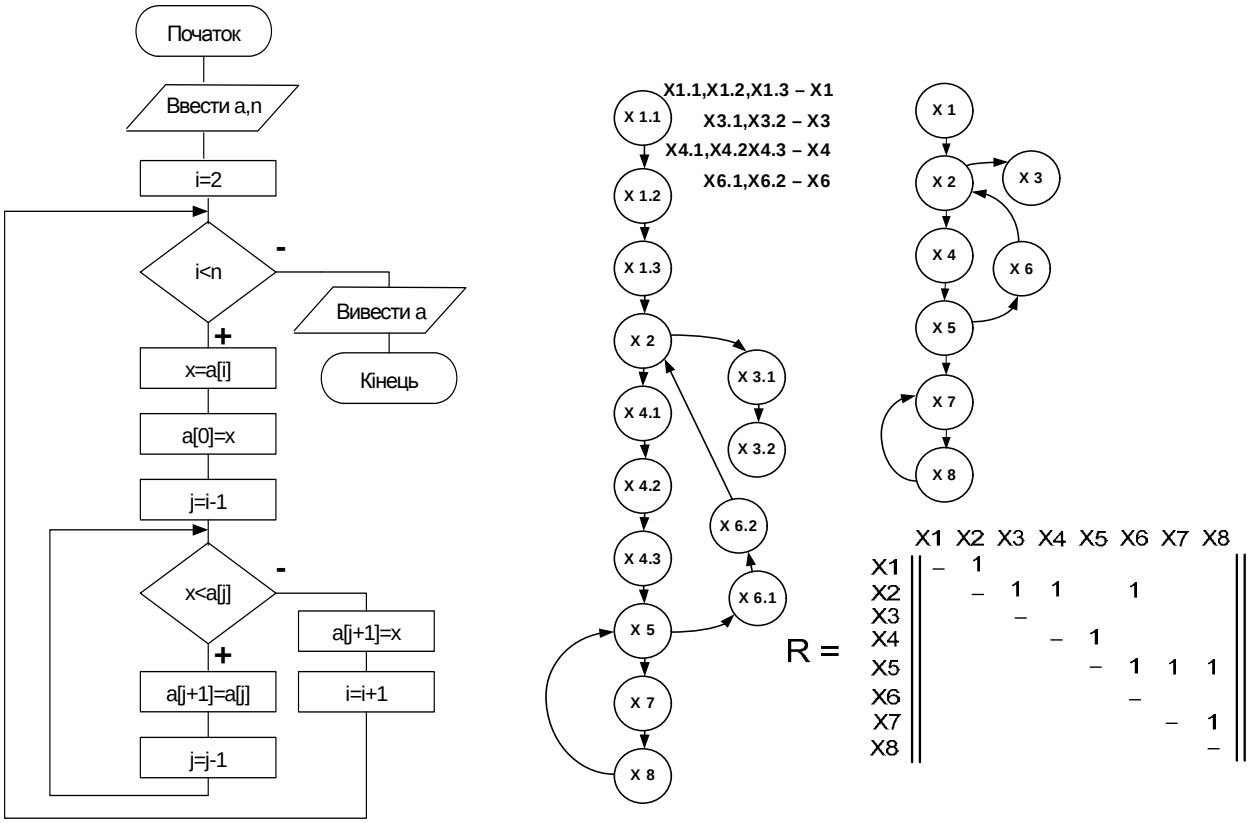

Рис. 2. Блок-схема програми її граф, стиснений граф та матриця сумісності

Структурна складність стисненого графу за матрицею суміжності дорівнює:

$$S = -FLog_2 \frac{F*2}{n(n-2)} = -9\text{Log}_2 \frac{9*2}{9*7} = 12{,}7.$$

Об’єктна складність за блок-схемою програми дорівнює 14, за стисненим графом – 8. Нагадаємо, що стиснення графу у нашому випадку означає неврахування стандартних операцій (початку, закінчення, операцій вводу–виводу) та об’єднання операційних вершин, з’єднаних послідовно одна за однією (вершина блок-схема програми є інтегрованою, вона може містити деяку кількість елементарних обчислювальних операцій). Умовні вершини не можуть бути об’єднані між собою і з операційними вершинами.

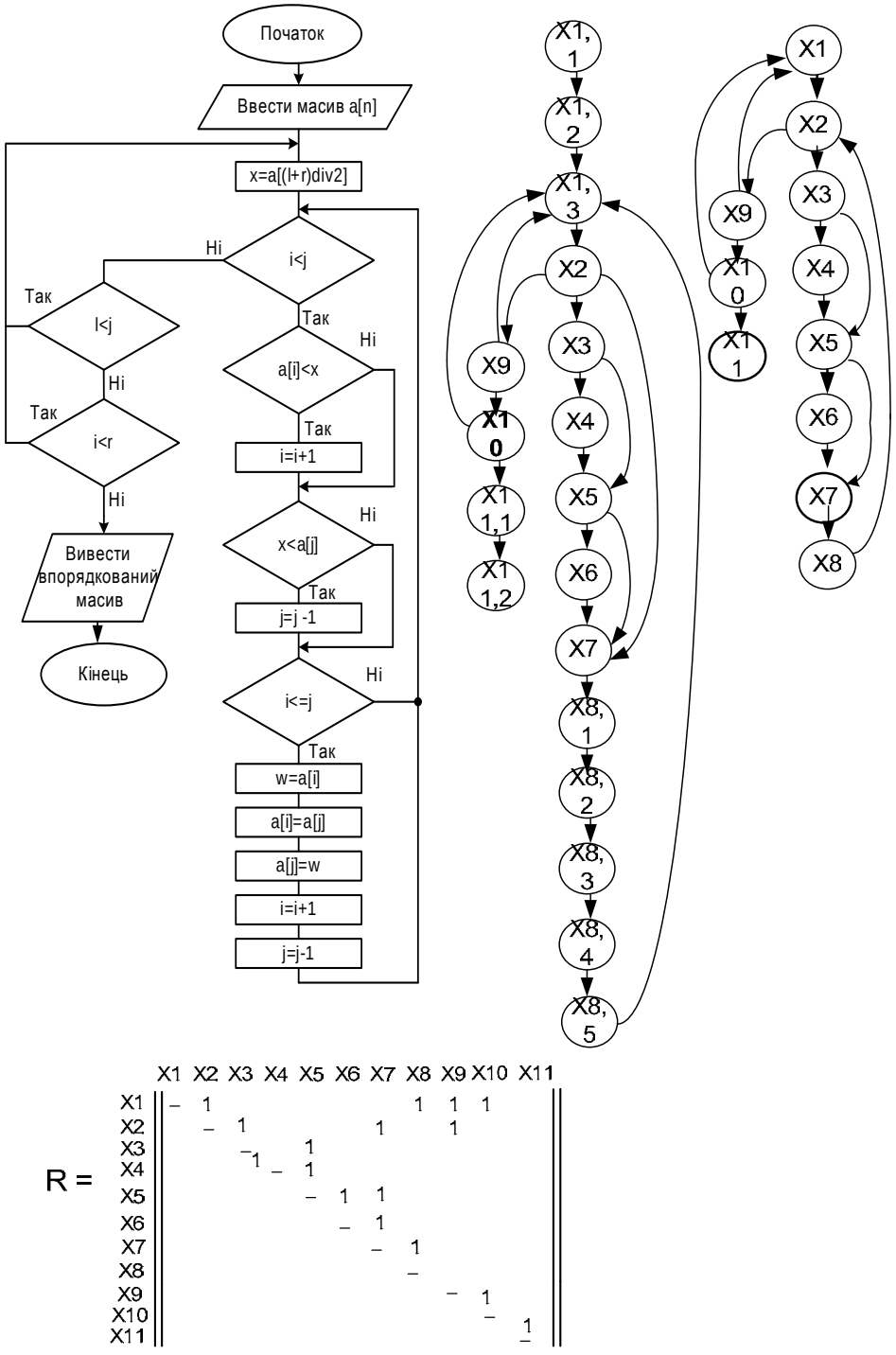

Рис. 3. Блок-схема програми її граф, стиснений граф та матриця сумісності

Ємнісна складність приблизно дорівнює розміру входу N (кількості чисел вхідного масиву).

Часова складність, визначена Н.Віртом, складається з операцій порівняння L_1 та пересилки L_2.

$L_{1серд} = (N^2 + N - 2)/4$; $L_{2серд} = (N^2 - 9N - 10\ 4)/4$. В обох випадках $L_{серд} = O(N^2)$

Швидке сортування (рекурсивний спосіб). Його опис, приклад, програма, аналіз кількості операцій порівняння та пересилок також наведені Віртом [4]. Нами здійснено подальший аналіз структурної складності та її взаємозалежності з іншими характеристиками алгоритму. На рис. 3 зображено блок-схему програми, її граф, стиснений граф та матрицю суміжності.

Структурна складність стиснення графу за матрицею суміжності дорівнює:

$$S = -FLog_2 \frac{F*2}{n(n-2)} = -16\mathrm{Log}_2 \frac{16*2}{11*9} = 26.$$

Об'єктна складність за блок-схемою програми дорівнює 18, за стиснутим графом 11.

Ємнісна складність приблизно дорівнює розміру входу N.

Часова складність, визначена Н. Віртом, складається з операцій порівняння L_1 та пересилання L_2.

$L_{1серд} = (2Ln2)N*Log_2N$; $L_{2\ серед} = \frac{N}{3}(Ln2)*Log_2N$. В обох випадках $L_{\ серед} = O(NLog_2N)$.

У таблиці наведено значення характеристик складності двох розглянутих блок-схем програм сортування. Простежується їх взаємозалежність. Збільшенням її структурної і об'єктної складності можна зменшити кількість операцій сортування. Збільшення об'єктної складності відбувається за рахунок зростання умовних операцій, тоді як кількість операційних вершин залишається незмінною.

Значення характеристик складності двох розглянутих блок-схем програм сортування

Характеристики складності	Сортування простим вибором	Сортування з розділенням
Часова (у середньому)	$O(N^2)$	$O(NLog_2N)$
Ємнісна	~N	~N
Об'єктна	14; 8	18; 11
Структурна	12,7	26

Висновки

1. Псевдо SH-модель призначена для аналізу блок-схеми програми. Формально вона відрізняється від SH-моделі алгоритму тим, що в її визначенні відсутня складова "програма".
2. Кількість характеристик складності дорівнює чотирьом: трьом технічним і одній інформаційній – структурній складності.
3. У процесі розрахунку структурної складності використовують матрицю суміжності, яку будують на основі стиснутого графу вершин блок-схема програми.
4. Синтез блок-схеми програми із зменшеною часовою складністю супроводжується збільшенням її структурної та об'єктної складності. Об'єктна складність є аргументом для визначення структурної складності.

1. Стивенс Р. Visual Basic. Готовые алгоритмы / Пер. с англ. – М., 2000. 2. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ / Пер. с англ. – М.: МЦНМО, 2000. – 995 с. 3. Кнут, Дональд Е. Искусство программирования для ЭВМ. Т1. Основные алгоритмы / Пер. с англ. – М.: Мир, 1976. – 736 с. 4. Вирт Н. Алгоритмы + структуры данных = программы / Пер. с англ. – М.: Мир, 1985. – 406 с. 5. Вирт Н. Алгоритмы и структуры данных / Пер. с англ. – 2-е изд., испр. – СПб.: Невский диалект, 2001. – 352 с. 6. Седжвик, Роберт. Фундаментальные алгоритмы на С++. Анализ/Структура данных/Сортировка/Поиск / Пер. с англ. – К.: Изд-во "ДиаСофт", 2001. – 688 с. 7. Седжвик, Роберт. Фундаментальные алгоритмы на С++. Алгоритмы на графах / Пер. с англ. – К.: Изд-во "ДиаСофтЮП", 2002. – 496 с. 8. Черкаський М. Складність апаратно-програмних

комп'ютерних засобів // Сучасні проблеми в комп'ютерних науках: Зб. наук. пр. – Львів: Держ ун-ту "Львівська політехніка", наукове товариство ім. Шевченка, 2000. – С. 58–67. 9. Черкаський М.В. SH-модель алгоритму // Вісн. Держ. ун-ту "Львівська політехніка". – Львів, 2001. 10. Черкаський М., Абдалла Саїд Садек. Псевдо SH-модель // Вісн. Нац. ун-ту "Львівська політехніка". – 2004. – № 523. – С. 145–150.

УДК 681.324

Б. Демида, А. Ратз*
Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління,
*Львівська залізниця

ОПТИМІЗАЦІЯ АЛГОРИТМУ ПОШУКУ КЛЮЧІВ ІЄРАРХІЧНОГО ДЕРЕВА СИСТЕМНОГО РЕЄСТРУ WIN32 ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ ПОТОКОВОЇ БАГАТОЗАДАЧНОСТІ

Зроблено спробу розроблення модифікованого алгоритму пошуку ключів ієрархічного дерева системного реєстру Win32 із використанням методу часового розпаралелювання процесу виконання операцій між множиною автономних потоків. Опис розробленого алгоритму містить також і порівняльну характеристику алгоритмів послідовного та паралельного виконання операцій пошуку з точки зору ефективності їхнього застосування. Ефективність описаного алгоритму визначають на основі розрахунку показника критичної тривалості виконання операцій пошуку ключів та процентної величини сумарного виграшу в часі завдяки використанню розробленого алгоритму.

The topic is being discussed in this technical article concerns the development of modified Win32 System Registry hierarchical tree key-node retrieval algorithm. According to the multithreading, the key-node retrieval operations are synchronously performed by the group of independent threads. The article's detailed review also includes the algorithm effectiveness evaluation which is based on determining the differences between either asynchronous or synchronous data retrieval process. The conclusion to the given material provides estimation of the overall productivity increasing percent ratio.

Вступ

Одними з найважливіших критеріїв оцінки ефективності впровадження сучасних програмних засобів оброблення великих масивів ієрархічних даних є час та швидкість виконання операцій [2]. До згаданих вище засобів насамперед належать і програмні засоби, орієнтовані на оброблення інформації, що зберігається в ієрархічних сховищах даних, зокрема в базі даних системного реєстру [2]. Головним недоліком існуючих сьогодні алгоритмічних розробок цих програмних засобів є невисокі показники продуктивності виконання операцій пошуку за достатньо великих обсягів інформації [2].

Серед існуючих методів підвищення продуктивності процесу оброблення даних найефективнішим було визнано метод часового розпаралелювання. Цей метод базується на розподіленні процесу оброблення даних між множиною автономних потоків, що забезпечують виконання однотипних операцій пошуку синхронно, за одиницю часу [4].

Встановлено, що загальний час виконання операцій над даними певного об'єму є обернено пропорційним кількості потоків, що виконують операції над цими даними. Застосування методу часового розпаралелювання стало можливим лише завдяки використанню багатозадачних програм-