УДК 621.3

Микола Черкаський, Саїд Садек Абдалла
Національний університет "Львівська політехніка",
кафедра комп'ютерної інженерії

РІВНІ СКЛАДНОСТІ МОДЕЛЕЙ АЛГОРИТМІВ

Розглянуто можливості визначення характеристик складності моделей алгоритму різних ієрархічних рівнів. Показано, що найбільш перспективними моделями для визначення часової, програмної і структурної складностей є мікропрограмний рівень і рівень блок-схеми програми.

The ways of determining program complexity of algorithm models of different hierarchical algorithm levels are examined. It is shown, that the most promising models for determining program and structural complexity are microprogram and program's block-diagram levels.

Вступ

Алгоритми Евкліда і аль-Хорезмі є прикладами ефективного розв'язання арифметичних задач. Вони ілюструють використання алгоритмів, але не дають тлумачення поняття "алгоритм", яке існує як самостійний, незалежний від конкретних прикладів, математичний об'єкт. Пройшло декілька століть перед тим, як поступово почалось викристалізовуватися це важливе поняття. Але й тепер не можна сказати, що цей процес завершився. Проблема полягає у наближенні поняття "алгоритм" до здобутків прикладної теорії обчислень, перетворення і зберігання даних. Усе те, що було зроблено в рамках класичної теорії алгоритмів в докомп'ютерну епоху, потрібно пов'язувати з досягненнями сучасних комп'ютерних наук. Саме ця проблема розглядається у цій роботі.

Перші формулювання

Перший період формулювання поняття алгоритму відноситься до часу від XIII до кінця XIX ст. Кнут Д. посилається на одне з ранніх джерел, що безпосередньо пов'язане з працями аль-Хорезмі, в якому наведено таке тлумачення алгоритму: "Під цією назвою об'єднані поняття про чотири арифметичні дії, а саме: про сумування, множення, віднімання і ділення" [1]. Ця назва не несе в собі посилань на властивості, характеристики, параметри, пізніше вона була замінена на значно ширше поняття – "арифметика". Перше тлумачення сутності терміна "алгоритм" відноситься до періоду XVI–XVII ст. і пов'язано з працями Хр. Рудольфа (1525) і Лейбніца (1684). Потреба в тлумаченні алгоритму була викликана, зокрема, пошуком єдиного універсального способу розв'язання будь-якої задачі. В адаптованій до сучасної термінології формі це тлумачення виглядає так: "*Алгоритм позначає будь-який регулярний обчислювальний процес, який за кінцеву кількість кроків розв'язує задачі визначеного класу*" [2]. Тут чітко простежуються три властивості алгоритму: детермінованість (регулярність), дискретність (кінцева кількість кроків) і масовість, що випливає зі слів "...розв'язує задачі визначеного класу". Крім того, фіксується параметр алгоритму – правило закінчення.

Отже, завершальне тлумачення першого етапу ґрунтувалося на відображенні властивостей обчислювального процесу – детермінованості, дискретності і масовості. Властивість "елементарність" не згадувалась. По замовчуванні крок алгоритму приймався таким, що дорівнював одній арифметичній операції.

Уточнення поняття “алгоритм”

Наприкінці XIX і на початку XX ст. в математиці виникло ряд проблем, вирішення яких вимагало уточнення – розуміння, що таке алгоритм. Проблеми виявилися складними. Їх вирішення звичайними математичними методами були ненадійними з точки зору можливості виникнення помилок і неефективними за витратами часу. Виникла необхідність побудови нового математичного апарата і засобів досліджень. Були прийняті дві принципові умови, загальні для будь-яких конкретних математичних побудов:

1) кількість операцій, необхідна для вирішення конкретної проблеми, повинна бути кінцевою;

2) операції повинні бути елементарними, щоб уникнути можливих помилок у складному ланцюзі інтуїтивних переходів в процесі вирішення проблеми.

Але вимога елементарності кроку алгоритму необхідна також для прикладних досліджень. Прикладна теорія складності алгоритмів, до тематики якої відноситься ця робота, ґрунтується на понятті елементарності кроку.

Слово “елементарність” математично не визначалось. Саме ця обставина сприяла появі багатьох різних напрямків подальшого уточнення поняття “алгоритм” і використання його для побудов систем дослідження проблем антимоній і розв’язності. Першою такою системою був апарат рекурсивних функцій. Значно більшого поширення набули такі алгоритмічні системи, як машина Тюрінга, нормальні алгоритми Маркова тощо. Вони дали змогу описати поняття “елементарність” на точно заданому переліку операцій. Прикладом тлумачення поняття алгоритм на основі алгоритмічних систем, що враховує повний набір властивостей, у тому числі властивості “елементарність”, є:

а) алгоритм – це процес послідовної побудови величин, який проходить в дискретному часі таким чином, що в початковий момент задається початкова скінченна система величин, а в кожний наступний момент система величин втримується за певним законом (програмою) із системи величин, які були в попередній момент часу (дискретність алгоритму);

б) система величин, яка отримана в якийсь (не початковий) момент часу, однозначно визначається системою величин, отриманих в попередні моменти часу (детермінованість алгоритму);

в) закон отримання наступної системи величин із попередньої повинен бути простим і локальним (елементарність кроків алгоритму);

г) якщо спосіб отримання наступної величини із якої-небудь заданої величини не дає результату, то повинно бути вказано, що потрібно вважати результатом алгоритму (спрямованість алгоритму);

д) початкова система величин може вибиратися із деякої потенційно нескінченної множини (масовість алгоритму)” [3].

Бурхливий розвиток комп’ютеризації в останню чверть XX ст. дав потужний поштовх утворенню нового напрямку інженерної діяльності – програмуванню. В цих умовах програма як одна з форм алгоритму не могла сприйматися як процес. Більш природно сприймається термін “припис”. В результаті тлумачення алгоритму набуло такого вигляду:

”*Алгоритм – точний припис, який задає обчислювальний процес (що називається в цьому випадку алгоритмічним), що починається з довільного початкового даного (з деякої сукупності можливих для даного алгоритму початкових даних) і спрямований на отримання результату, який повністю визначається цим початковим даним*“ [4].

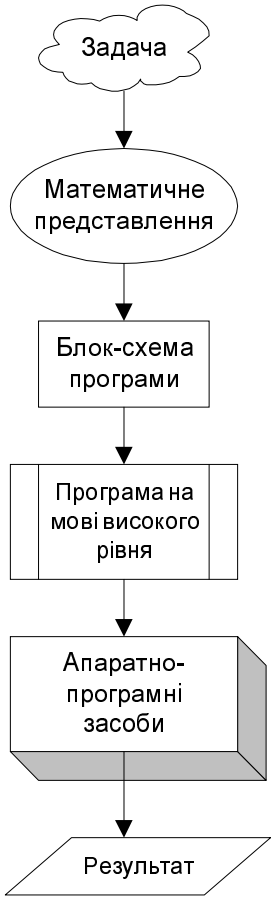
У цьому тлумаченні алгоритм розглядається не як процес, а як “...припис, що задає …процес…”, як засіб розв’язання задачі. Слова “точний припис” не мають однозначного тлумачення. Для дослідження алгоритму, що визначається як точний припис, використовуються

моделі. Наприклад, це може бути алфавітний оператор, математична функція, граф, блок-схема програми, програма на мові високого рівня. Істотною особливістю такого представлення алгоритмів є відсутність засобів його реалізації. Назвемо клас таких алгоритмів приписними алгоритмами. Другий клас представлення алгоритмів – алгоритмічні системи або формальні алгоритмічні системи, які мають такі засоби (машина Тюрінга, нормальні алгоритми Маркова), SH-модель [5] тощо. Поряд з програмою в цих моделях передбачені уявні або технічні засоби розв'язання задач.

Моделі приписних алгоритмів та деяка формальна алгоритмічна система пов'язані між собою. В процесі розв'язання задачі вони утворюють декілька рівнів – ланцюг, який дає змогу поетапно наближатись до розв'язку і отримувати кінцевий результат. На останньому етапі використовується формальна алгоритмічна система, а на попередніх – послідовність моделей приписних алгоритмів. Етапність підготовки, наявність декількох рівнів моделей алгоритмів і розв'язку задачі зумовлена намаганням мінімізувати можливі помилки і оптимізувати значення характеристик складності задачі в процесі комп'ютерних перетворень.

Моделі алгоритмів

Термін "складність" в комп'ютерних науках використовується як характеристика кількості операцій, що потрібна для розв'язання деякої задачі. Ця характеристика – часова складність – має найбільшу вартісну вагу. Крім часової складності, існують інші характеристики. Серед них програмна складність, яка поступається вартісному показнику лише часової складності. Програмна складність характеризує логіку побудови алгоритмів, ступінь інтелектуальних зусиль, необхідних для синтезу ефективних алгоритмів, вона впливає на витрати часу проектування алгоритму. Комп'ютерне розв'язання задачі охоплює декілька рівнів (див. рисунок). Для кожного рівня розробляється своя модель алгоритму і оптимізуються її характеристики. Усі характеристики складності є взаємозалежними. Практично мета оптимізації полягає у досягненні у процесі проектування мінімальних значень часової (в першу чергу) та програмної складності за рахунок збільшення інших характеристик, наприклад, апаратної, структурної, місткісної. Але модель кожного рівня має свій набір характеристик, тому способи оптимізації характеристик різних моделей є різні. Розглянемо особливості дослідження характеристик складності кожного рівня.



Рівень математичних моделей алгоритму

До моделей цього рівня відносяться аналітичні співвідношення, графічні, табличні залежності, вербальні представлення алгоритмів. Усі вони належать до класу приписних алгоритмів з абстрактним обчислювачем. Ознакою цих алгоритмів є інтегрованість кроку, властивість "елементарність" є відсутньою. Дослідження цих алгоритмів здебільшого проводиться з метою побудови ефективних за часовою складністю алгоритмів. Наприклад, алгоритм ШПФ, що виводиться з ДПФ, має істотно меншу часову складність порівняно з ДПФ. Програмна складність у таких випадках зростає, але розглядається лише у якісному плані. Поки що немає кількісних засобів її оцінки на цьому рівні.

Блок-схема програми

Ця модель є проміжною ланкою між математичними залежностями і програмами на мовах високого рівня. Вона теж відноситься до класу приписних алгоритмів і є моделлю алгоритму з абстрактним обчислювачем [6]. Кожний блок моделі є інтегрованим і відрізняється від інших блоків кількістю внутрішніх операцій. Тому часова складність по блок-схемі не підраховується. Модель має властивість "ієрархічність". На кожному ієрархічному рівні: системному, проміжних, детальному – можна оцінити програмну складність, що визначається кількістю операційних вершин блок-схеми. Це дає змогу якісно оцінити інформаційну місткість моделі алгоритму. Блок-схема програми уможливлює досліджувати іншу інформаційну характеристику – структурну складність [6]. При визначенні структурної складності користуватимемося матрицею інциденцій, яка в якнайповнішому обсязі містить дані про міжзв'язки системи.

Визначення. Структурна складність алгоритмічного пристрою є ступенем нерівномірності матриці інциденцій:

$$S = -F \log_2 \frac{F}{q \cdot r},$$

де F – кількість додатних і від'ємних елементів матриці інциденцій системи; $q \cdot r$ – розмір матриці.

У матриці можуть існувати її фрагменти, які повторюються. За визначенням вони не повинні враховуватися в структурній складності. У такому разі $F = \sum_l f_l$, де f_l-тий фрагмент матриці. Узагальнену умову визначення фрагментів, що повторюються, можна записати у такому вигляді:

$$\forall_{i,j}; i \neq j, \left\{ \left[\left(\varphi_i \cap \varphi_j \neq \varphi_i \vee \varphi_j \right) \vee \left(\varphi_i \cap \varphi_j = \varnothing \right) \Rightarrow f_l = \left| \varphi_i \right| + \left| \varphi_j \right| \right] \vee \left[\left(\varphi_i \cap \varphi_j = \varphi_i \vee \varphi_j \right) \right] \Rightarrow f_l = \left| \varphi_j \right| \right\}, \quad (2)$$

де φ_i, φ_j – і-тий і j-тий фрагменти матриці інциденцій; f_l – l-та потужна множина з'єднань у фрагменті.

Властивість "ієрархічність" і можливість оцінювати структурну складність є істотними перевагами блок-схеми програми перед математичними моделями та програмами на мовах високого рівня.

Програми на мовах високого рівня

Програмну складність програм на мовах високого рівня оцінюють кількістю інструкцій. Інструкція – крок цієї моделі алгоритму, також є інтегрованою. Кількість інструкцій лише наближено свідчить про складність логіки побудови програми. Через значну розбіжність в часі виконання інструкцій дослідження часової складності заміняють на оптимізацію часу розв'язання задачі [5]. Значну увагу приділено оптимізації використання основної пам'яті і кеш. Це свідчить про необхідність пошуку нових моделей програмування, які б враховували вимоги комп'ютерних засобів.

Мікропрограмний рівень

Організація обчислювального процесу на рівні операційних пристроїв або вузлів процесора проводиться під керуванням мікропрограм. Вона реалізується апаратними засобами і розміщається разом з функціональними вузлами на кристалі. Через те, що пристрій керування безпосередньо участі у перетворенні даних не бере, в загальному випадку необхідно мінімізувати площу, яку він займає на кристалі. Звідси випливає вимога мінімізувати величину програмної складності.

При синтезі, аналізі і оптимізації процесорів доцільно користуватися SH-моделлю алгоритму [5]. Вона відповідає всім вимогам формальної алгоритмічної системи і дає змогу розробляти апаратно-програмні засоби за п'ятьма характеристиками складності. Для однієї мікропрограми

програмна складність визначається як ступінь нерівномірності розташування мікронаказів в границях часової діаграми.

Існують дві групи способів мінімізації величини програмної складності. До першої відносяться ті, що зумовлені вимогами безпосередньо програмування, наприклад, обмеження кількості команд і їх типів. Друга група включає способи побудови функціональних вузлів, які вимагають зменшених витрат на керування. Прикладами таких способів є: апаратне виконання обчислювальних операцій, елементарних і спеціальних функцій; розділення трактів оброблення, зберігання, перетворення даних.

Висновки

1. Комп'ютерне розв'язання задач поділяється на декілька рівнів. Для кожного рівня розробляється і оптимізується власна модель.

2. Програмна складність алгоритму будь-якого рівня, яка описує його логічну побудову, поряд з часовою складністю є важливою його характеристикою.

3. На рівні математичних залежностей досліджують часову складність і обирають модель з мінімальною часовою складністю. На цьому рівні інші моделі не мають формального опису.

4. Блок-схема програми дає графічне двовимірне уявлення про алгоритм. Вона в якнайповнішому обсязі серед приписних алгоритмів дає можливість оцінити структурну складність.

5. Оцінка характеристик складності програм на мові високого рівня обмежується лише кількістю інструкцій.

6. Якнайповнішу оцінку характеристик складності отримують за допомогою SH-моделі на мікропрограмному рівні.

1. Кнут Д. Искусство программирования для ЭВМ. Т1. Основные алгоритмы / Пер. с англ. – М., 1976. 2. История математики в средние века / Под ред. А.П. Юшкевича – М., 1976. 3. Марков А.А. Теория алгорифмов. –М.–Л., 1954. (Труды МИАН. Т. 42). 4. Математическая энциклопедия / Гл. ред. И.М. Виноградов. – М., 1977. –Т.1. 5. Черкаський М.В. SH-модель алгоритму // Комп'ютерна інженерія та інформаційні технології / Вісник НУ "Львівська політехніка". – Львів, 2001. – №433. – С.127–134. 6. Черкаський М. Структурна складність // Комп'ютерна інженерія та інформаційні технології / Вісник НУ "Львівська політехніка". – Львів, 2002. – №450. – С.121–126.