УДК 004.89

К.П. Осадча, О.В. Хромишев

Мелітопольський державний педагогічний університет імені Богдана Хмельницького, Мелітополь

АНАЛІЗ МЕТОДІВ РОЗРОБКИ АЛГОРИТМІВ РОЗВ'ЯЗАННЯ МАТЕМАТИЧНИХ ЗАДАЧ ЗАСОБАМИ МОВИ PYTHON

У статті на основі аналізу літературних джерел зроблено висновок про те, що кожний із загальних методів розробки алгоритмів (перебору, грубої сили, зменшення розміру задачі, декомпозиція, перетворення, жадібні алгоритми, динамічне програмування, ітеративне поліпшення, пошук з поверненням, метод гілок і меж, локальний пошук) придатний до вирішення конкретного кола математичних задач. Автори запропонували методи розробки алгоритмів для розв'язання задачі на знаходження найбільшого спільного дільника та задачі на обчислення факторіалу числа. На основі виділених математичних можливостей мови програмування Python у статті здійснено аналіз ефективності запропонованих методів на основі використання модулів profile itime та методу timeit мови Python.Зроблено висновок про ефективність методу декомпозиції.

***Ключові слова**: методи розробки алгоритмів, аналіз ефективності, Python.*

Вступ

Постановка проблеми. У всіх сферах діяльності людина стикається з різноманітними методиками рішення різноманітних задач. Рішення задач на комп'ютері засноване на понятті алгоритму. Алгоритми мають важливе значення для наукової і технічної сфери, адже їх використання є етапом проектування, дає загальну методику пошуку загального рішення задач в інформатиці. Методи розробки алгоритмів є стратегічним планом рішення, який знадобиться не залежно від області вирішуваних задач. Найчастіше – цематематичні задачі, з вирішення яких починається вивчення будь-якої мови програмування.

Серед давно відомих сучасних мов програмування останнім часом набула популярності мова Python, що засвідчує рейтинг мов програмування, здійснюваний українським IT-порталом DOU.UA [11] та зарубіжна аналітична компанія RedMonk [22]. Являючись інтерпретованою об'єктно-орієнтованою мовою програмування високого рівня з динамічною семантикою, Python підтримує декілька парадигм програмування, зокрема: об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану. Тому викладачі програмування і практикуючі програмісти вважають Python придатною для вирішення великого кола задач, зокрема математичних.

Аналіз останніх досліджень. Теорії алгоритмів у вітчизняній і зарубіжній літературі приділено багато уваги, зокрема роботи у цьому напрямку представлено Галлієвим Ш.І., Гудрічем М.Т. (Goodrich M.), Єршовим А.П., Кнутом Д. (Knuth D.), Левітіним А.В., Макконнеллом Дж. (McConnell J.), Марковим А.А., Приймою С.М., Татассія Р. (Tamassia R.), Царевим Р.Ю. та ін.

Особливості мови програмування Python розглядали Бізлі Д., Гифт Н., Джонс Д., Круглик В.С., Лутці М., Саммерфілд М., Сваруп С.Х. та ін. Практичними розробками для мови програмування Python займалися Пилєва О.О., Хахаєв І.А., зокрема засоби математичних обчислень Python вивчали Репін А.Ю., Соловьйов І.А., Червяков А.В., Шібзухов З.М.

Мета роботи. Такі аспекти як ефективність методів розробки алгоритмів для розв'язання математичних задач за допомогою мови програмування Python висвітлюються не системно і не достатньо. Тому за мету нами було поставлено здійснити аналіз ефективності методів розробки алгоритмів розв'язання математичних задач засобами мови Python.

Основна частина

До теперішнього часу фахівці з інформаційних технологій розробили ряд ефективних методів, які нерідко дозволяють отримувати ефективні алгоритми вирішення великих класів задач.Метод проектування алгоритму – це універсальний підхід, застосовуваний для алгоритмічного вирішення широкого кола завдань, що відносятьсядо різних областей обчислювальної техніки [3, с. 36]. Ці методи забезпечують набір універсальних принципів, керуючись якими можна розробити алгоритми вирішення завдань.

Загальні методи розробки алгоритмів – це найбільш загальні підходи до розробки алгоритмів, які можна застосувати до найрізноманітніших задач. До них відносять: метод перебору, метод грубої сили, зменшення розміру задачі, декомпозицію, перетворення, жадібні алгоритми, динамічне програмування, ітеративне поліпшення, пошук з поверненням, метод гілок і меж, локальний пошук [3].

З аналізу літературних джерел [3 – 7] ми зробили висновок, що кожний з наведених загальних

методів придатний до вирішення конкретного кола завдань, зокрема і математичних задач. Метод грубої сили доцільно використовувати для багатьох елементарних, але важливих алгоритмічних задач, таких як обчислення суми, пошук найбільшого елемента в списку і т.д. Метод зменшення розміру задачі є природнім підходом для розробки алгоритмів для генерації елементарних комбінаторних об'єктів. Метод декомпозиції ідеально підходить для паралельних обчислень, успішно застосовується для розв'язання таких важких задач обчислювальної геометрії, як задача про пару найближчих точок і задача про випуклу оболонку. Метод перетворення, являючи собою групу методів, застосований для розв'язання системи лінійних рівнянь, обчислення поліномів, зведення у ступінь, пірамідального сортування. Жадні методи розглядаються як загальні методи проектування алгоритмів, застосованих для задач оптимізації. Метод динамічного програмування використовується для рішення задач з перекриваючими підзадачами, а саме: обчислення біноміального коефіцієнта, оптимальної тріангуляції трикутника, складних інженерних задач, задач пошуку найкоротших шляхів між всіма парами вершин, складних задач комбінаторної оптимізації тощо. На методі ітеративного поліпшення засновані сортування підрахунком розподілення і алгоритми пошуку підрядків. Метод пошуку з поверненням і метод гілок та меж роблять можливим обчислення великих примірників складних комбінаторних задач. Методи локального пошуку потрібні для розв'язання задач, точні рішення яких вимагають експоненціальних витрат часу: задачі комівояжера, задачі одномірного розміщення блоків, задачі про «ханойські вежі» та ін.

В окремих джерелах [8, 9, 10, 11] можна знайти згадування про інші методи розробки алгоритмів, які можна назвати специфічними через ті задачі, для рішення яких вони пропонуються. До них ми віднесли: чисельні методи (методи систем лінійних алгебраїчних рівнянь, методи диференційних рівнянь, методи Монте-Карло та ін.), метод покрокової розробки, методи локального спуску, методи паралельних обчислень та ін.

Аналіз книг і навчальних посібників з Python [12, 13, 14, 15] дозволив нам зробити висновки щодо її особливостей та можливостей розв'язання математичних задач. Це вільно розповсюджувана мова програмування з великою кількістю вільно розповсюджуваних модулів і бібліотек модулів для створення програм майже будь-якого типу (веб-додатки, робота з базами даних, наукові обчислення та інженерні розрахунки, візуалізація даних,розробка користувацького інтерфейсу, програмування нейронних мереж і т.д.). Її доцільно використовувати для розв'язання математичних задач, тому що вона близька з MATLAB йіснує вільний і добротний інструментарій для наукових обчислень, інженерних розрахунків, візуалізації моделей і даних, доступний на будь-якій платформі (Windows, Linux, OS X і т.д.).

Ми виділити такі математичні можливості цієї мови програмування: 1) існування чотирьох вбудованих типів числових символів: булеві значення, цілі числа, числа з плаваючою точкою, комплексні числа; 2) підтримка всіх основних математичних операцій і дій; 3) наявність математичних модулів: decimal, fractions, math, numbers, random; 4) наявність бібліотек для складних обчислень: SymPy, Numeric, NumPy, Pandas, Matplotlib, Dislin та ін.

Для досягнення мети дослідження нами було обрано двітипові задачі, а саме: задача на знаходження найбільшого спільного дільника ізадача на обчислення факторіалу числа.

Задача на знаходження найбільшого спільного дільника (НСД) розв'язувалася за допомогою алгоритму Евкліда з використанням двох методів його розробки: методу грубої сили (на основі оператора умови) – структурний алгоритм, методу грубої сили (з використанням функції) – ітеративний алгоритм, методу декомпозиції – рекурсивний алгоритм. Для оцінки ефективності запропонованих методів розробки алгоритмів для розв'язання задачі на знаходження НСД засобами Python ми використали елементи модуля profile та time. З допомогою профайлера (модуль profile) ми дізналися, скільки часу займає виконання різних функцій і методів у трьох розроблених програмах.

Для отримання результатів нами було введено такі вхідні дані:125 і 75.У підсумку роботи програм було отримано такі результати. Виконання різних функцій і методів у алгоритмі знаходження НСД, який було розроблено на основі методу грубої сили з використанням оператора умови зайняло 0,016 с. Виконання різних функцій і методів у алгоритмі знаходження НСД, який було розроблено на основі методу грубої сили з використанням функції, зайняло 0,062 с. Виконання різних функцій і методів у алгоритмі знаходження НСД, який було розроблено на основі методу декомпозиції зайняло 0,000 с. Для отримання більш показових даних при виконанні рекурсивного алгоритму ми збільшили вхідні дані задачі (123456789123456789 і 102345678102345678) і отримали такі дані – 0,016 с.Наведемо приклад програми мовою Python 3.5, реалізованої на основі методу декомпозиції:

```
1. m = int(input("Введіть перше ціле число: "))#введення 1-го числа
2. n = int(input("Введіть друге ціле число: ")) "))#введення 2-го числа
3. defgcd(m, n): #оголошення функції
```

```
    4. return a ifn == 0 elsegcd(n, m % n) #Якщо n =0 - повертаємо a, інакше повертаємо результат роботи цієї функції з іншими аргументами
    5. gcd = gcd(m, n) #запит на виконання функції
    6. print ("Найбільший спільний дільник чисел %s і %s = %s" % (m, n, gcd)) #виведення результату
```

Отже, рекурсивний алгоритм розв'язання задачі знаходження НСД розроблений на основі методу декомпозиції дозволяє суттєво зменшити час виконання функцій, ніж структурний та ітеративний алгоритми на основі методу грубої сили. Рекурсивний алгоритм виконується швидше: у ньому немає циклів, що спричиняють збільшення часу роботи функцій. Слід зазначити, що результати роботи модулю profile можуть відрізнятися залежно від того, скільки процесів запущено на комп'ютері під час роботи програми. Також показники залежать від величини аргументів, що задаються. Тому ми скористалися ще одним способом з'ясування ефективності алгоритму.

Для того, щоб дізнатися, як довго виконується програма на мові Python, найпростіший спосіб визначити це полягає в тому, щоб запустить її під керуванням утиліти time.

У результаті виконання програм із використанням можливостей модулю time ми отримали такі дані (табл. 1).

Таблиця 1

Результати аналізу ефективності
методів розробки алгоритму розв'язання задачі
на знаходження НСД із використанням можливостей модулю time

№ п/п	Назва методу	Дійсний час виконання програми (сек)	Процесорний час виконання програми (сек)
1	Метод грубої сили (з використанням оператора умови)	3,763215	3,762456
2	Методу грубої сили (з використанням функції)	2,502143	2,502061
3	Метод декомпозиції	2,559146	2,559125

З аналізу цих результатів виконання 3-х програм можна зробити висновок, що на роботу ітеративного алгоритму витрачається менше часу ніж на роботу структурного і рекурсивного алгоритмів. Отже, на цьому прикладі розв'язання задачі знаходження НСД можна зробити висновок про те, що ефективність алгоритму, розробленого на основі методу грубої сили, залежить від алгоритмічних структур, використовуваних у розробленій програмі. Метод грубої сили з використанням функції, в якій основою є цикл, виявляється ефективніше методу декомпозиції, проте не на достатньо значний показник. Тому можна зробити остаточний висновок, що ефективним методом для розв'язання задачі на знаходження НСД можна вважати метод декомпозиції.

До розв'язку задачі на обчислення факторіалу числа можна застосувати кілька математичних методів. Ми обрали стандартний метод знаходження добутку всіх натуральних чисел від 1 до n включно; метод дерева та метод факторизації (розкладання факторіала на прості множники).Для розробки алгоритму і написання програми мовою Python ми обрали такі методи розробки алгоритмів: грубої сили, декомпозиції і перетворення.

Для оцінки ефективності запропонованих методів розробки алгоритмів для розв'язання задачі обчислення факторіалу числа засобами Python ми додали до коду розроблених програм елементи модуля profile itime, а також до двох останніх програм метод timeit, що перевіряє час роботи конкретної функції. Для всіх програм було введено однаковий аргумент – 3000. У результаті роботи програм було отримано такі результати. Виконання різних функцій і методів у алгоритмі, який було розроблено на основі методу грубої сили зайняло 0,0 секунд, дійсний час виконання всієї програми –4,204 с, процесорний час – 4,203.

Виконання різних функцій і методів у алгоритмі, який було розроблено на основі методу декомпозиції зайняло 0,006 с, дійсний час виконання всієї програми –0,64003 с, процесорний час – 0,64087. Виконання різних функцій і методів у алгоритмі, який було розроблено на основі методу перетворення зайняло 0,004 секунд, дійсний час виконання всієї програми –0,66603 с, процесорний час –0,6615 с.

З аналізу цих результатів виконання 3-х програм можна зробити висновок, що на роботу функцій алгоритму, розробленого методом грубої сили витрачається менше часу ніж на роботу алгоритму, розробленого методом декомпозиції та методом перетворення. Це можна пояснити відсутністю складних функцій і алгоритмічних структур. А час виконання програм показує перевагу методу декомпозиції з використанням рекурсії.

Отже, на цьому прикладі розв'язання задачі обчислення факторіалу числа можна зробити одноз-

начний висновок про те, що алгоритм розроблений на основі методу декомпозиції виявився більш ефективним, хоча не значно, порівняно із алгоритмом, що розроблений методом перетворення. Проте різниця у часі виконання цього алгоритму, порівняно з алгоритмом розробленим методом грубої сили, дуже важлива.

Висновки

Таким чином, на основі розробки і реалізації мовою Python алгоритмів для розвязання двох математичних задач (на знаходження найбільшого спільного дільника та обчислення факторіалу числа) ми здійснили аналіз ефективності таких методів розробки алгоритмів як метод грубої сили, метод декомпозиції і метод перетворення. Цей аналіз здійснювався на основі з'ясування швидкості виконання функцій і програм з використанням модулів profile itime та методу timeit мови Python.

Його результати засвідчили ефективність методу декомпозиції, який полягає у розділенні задачі на декілька підзадач, рекурсивному їх розв'язку і комбінуванні розв'язку початкової задачі з розв'язків допоміжних підзадач.

Список літератури

1. Рейтинг языков программирования № 6: новые лошадки на коммерческом рынке [Електронний ресурс]. – Режим доступу до ресурсу: URL: http://dou.ua/lenta/articles/language-rating-jan-2015.

2. The Red Monk Programming Language Rankings: June 2015 [Електронний ресурс]. – Режим доступу до ресурсу: URL: http://redmonk.com/sogrady/ 2015/07/ 01/language-rankings-6-15.

3. Левитин А.В. Алгоритмы: введение в разработку и анализ: пер. с англ. / А.В. Левитин. – М.: Издательский дом «Вильямс», 2006. – 576 с.

4. Культин Н.Б. Программирование в Turbo Pascal 7.0 и Delphi / Н.Б. Культин. – СПб: БХВ-Петербургб 2007. – 400 с.

5. Ахо А.В. Структуры данных и алгоритмы: пер. с англ.: уч. пос. / А.В. Ахо, Дж. Хопкрофт, Дж.Д. Ульман. – М.: Издательский дом «Вильямс», 2007. – 400 с.

6. Молчановський О. Конспект лекцій «Теорія алгоритмів: Тема 3. Метод декомпозиції» [Електронний ресурс]. – Режим доступу до ресурсу: URL: http://oim.asu.kpi.ua/files/TA/03_Divide_and_conquer.pdf.

7. Кормен Т. Алгоритмы: построение и анализ = IntroductiontoAlgorithms / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К.Штайн. – 2-е. – М.: Вильямс, 2005. – 1296 с.

8. Колдаев В.Д. Численные методы и программирование: учебное пособие / В.Д. Колдаев; под ред. проф. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2009. – 336 с.

9. Касьянов В.Н.. Практикум по программированию / В.Н. Касьянов, Е.В. Касьянова. – Новосибирск, 2004. – 200 с.

10. Кочетов Ю.А. Методы локального поиска для дискретных задач размещения: дисс. ... докт. физико-математических наук:05.13.18 / Кочетов Юрий Андреевич. – Новосибирск, 2009. – 267 с.

11. Баркалов К.А. Методы параллельных вычислений: метод. пособ. / К.А. Баркалов. – Нижний Новгород, 2011. – 124 с.

12. Соловьёв И.А. Вычислительная математика на смартфонах, коммуникаторах и ноутбуках с использованием программных сред Python: учеб. пособ. / И.А. Соловьёв, А.В. Червяков, А.Ю. Репин. – СПб.: Издательство «Лань», 2011. – 272 с.

13. Бизли Д. Python. Подробный справочник / Д. Бизли; пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с.

14. Лутц М. Изучаем Python, 4-е издание / М. Лутц; пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.

15. Саммерфилд М. Программирование на Python 3. Подробное руководство /М. Саммерфилд; пер. с англ. – СПб.: Символ-Плюс, 2009. – 608 с.

Надійшла до редколегії 15.01.2016

Рецензент: д-р техн. наук, проф. В.С. Єремєєв, Мелітопольський державний педагогічний університет імені Богдана Хмельницького, Мелітополь.

АНАЛИЗ МЕТОДОВ РАЗРАБОТКИ АЛГОРИТМОВ РЕШЕНИЯ МАТЕМАТИЧЕСКИХ ЗАДАЧ СРЕДСТВАМИ ЯЗЫКА PYTHON

Е.П. Осадчая, А.В. Хромышев

В статье на основе анализа литературных источников сделан вывод о том, что каждый из общих методов разработки алгоритмов (перебора, грубой силы, уменьшения размера задачи, декомпозиция, преобразование, жадные алгоритмы, динамическое программирование, итеративного улучшения, поиск с возвратом, метод ветвей и границ, локальный поиск) пригоден к решению конкретного круга математических задач. Авторы предложили методы разработки алгоритмов для решения задачи на нахождение наибольшего общего делителя и задачи на вычисление факториала числа. На основе выделенных математических возможностей языка программирования Python в статье осуществлен анализ эффективности предложенных методов на основе использования модулей profile и time, метода timeit языка Python. Сделан вывод об эффективности метода декомпозиции.

Ключевые слова: *методы разработки алгоритмов, анализ эффективности, Python.*

ANALYSIS OF METHODS FOR ALGORITHMS DEVELOPMENT TO COMPLETE MATHEMATICAL TASKS BY LANGUAGE PYTHON

K.P. Osadcha, O.V. Hromyshev

On the basis of literary sources analysis the article concludes that each of the common methods for algorithms development (brute force, reduction of the task size, decomposition, transformation, greedy algorithms, dynamic programming, an iterative improvement, backtracking, branch-and-bound method, local search) is suitable to complete a range of mathematical tasks. The authors have proposed methods for algorithms development to complete the tasks of finding the greatest common divisor and the factorial number. Based on the selected Python mathematical possibilities the article analyzes the effectiveness of the proposed methods through the use of modules profile and time and method timeit of language Python.

Keywords: *methods for algorithms development, efficiency analysis, Python.*