демонструють зростання виробництва [5], що викликане зовсім не поліпшенням економічного стану України, а військовими замовленнями, призначеними зупинити агресора.

Література

1. Укравтопром. *Статистика.* URL: https://ukrautoprom.com.ua/category/statistika
2. Средняя стоимость нового автомобиля в Украине выросла до 24 750 евро. URL: https://autonews.autoua.net/novosti/13805-srednyaya-stoimost-novogo-avtomobilya-v-ukraine-vyrosla-do-24-750-evro.html
3. Импорт легковых автомобилей в Украину обвалился. Цены подросли. URL: https://biz.liga.net/ekonomika/avto/novosti/import-legkovyh-avtomobiley-v-ukrainu-obvalilsya-tseny-podrosli
4. Запорізький автомобілебудівний завод. *Вікіпедія.* URL: https://uk.wikipedia.org/wiki/Запорізький_автомобілебудівний_завод
5. Кременчуцький автомобільний завод. *Вікіпедія.* URL: https://uk.wikipedia.org/wiki/Кременчуцький_автомобільний_завод

УДК 372.8:004.43

МЕТОДИЧНА ЦІННІСТЬ КОМБІНАТОРНИХ АЛГОРИТМІВ ПРИ ВИВЧЕННІ БАЗОВИХ ЗАСАД АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ ДЛЯ ПРОФІЛЬНИХ ТА ГУМАНІТАРНИХ СПЕЦІАЛЬНОСТЕЙ

О. С. Вєтров

Дисципліни, пов’язані із вивченням мов прикладного програмування, програмної інженерії та прикладних аспектів побудови та використання ефективних алгоритмів є невід’ємною частиною освітніх програм усіх спеціальностей галузі знань 12 «Інформаційні технології», спеціальності 113 «Прикладна математика» тощо. В останній час серйозну увагу до надання студентам навичок розробки прикладного програмного застосунку приділяють і спеціальності традиційного гуманітарного профілю, зокрема спеціальність 035.10 «Прикладна лінгвістика». Це зрозуміло, оскільки сучасні дослідження в області прикладної лінгвістики, як прикладного так і фундаментального характеру, не можуть бути ефективними без використання сучасних систем комп’ютерної обробки тексту, створення власних алгоритмів аналізу великих обсягів текстової інформації, і врешті-решт технологій машинного навчання, зокрема систем обробки природньої мови (NLP – Natural-language processing). Основою фахового використання зазначених технологій є перш за все чітке розуміння основ програмування та алгоритмізації.

Базовою мовою програмування для непрофільних спеціальностей (поза галузі «Інформаційні технології»), особливо коли справа стосується перш за все прикладного програмування, часто обирають Python. На це є декілька причин. Зазначимо дві основні. По-перше, низький поріг входження у порівнянні з мовами C++, C#, Java тощо. При цьому, не можна сказати, що Python є найпростішою мовою програмування. Серед затребуваних на ринку можна згадати потужну за своїм функціоналом мову JavaScript, вивчення основ якої, особливо на перших порах, є також значно більш ефективним, ніж, скажімо, мов вже заданої C-групи. При цьому застосування JavaScript зрештою зосереджене головним чином на веб-програмуванні, тому обирати JavaScript мовою загального використання є нераціональним. По-друге, мова Python має чи не найрозвиненішу мережу сторонніх бібліотек прикладного призначення, більшість з яких є відкритими для використання. Варто відмітити, що в мережі є доступними велика кількість відкритих ресурсів для вивчення програмування на Python. При цьому необхідно

також зазначити, що перелік сучасної україномовною літератури (включаючи переклади відомих іноземних видань), присвяченої проблемам алгоритмізації та програмування, зокрема мові Python, є досить скромним. На думку автора, заслуговують на увагу видання [1–4] останніх 5–7 років, написані колегами з київських та львівських ЗВО.

З досвіду роботи автора, при викладанні основ програмування для студентів перших курсів незалежно від спеціальності, основними труднощами для більшості студентів є не стільки вивчення основ синтаксису мову програмування, скільки більш фундаментальна проблема: вміння формалізувати задачу, розбити її на конкретні кроки, а вже потім реалізація програмного коду на основі структурної парадигми програмування. В підвалинах вивчення основ програмування лежить не стільки мова програмування, скільки алгоритмізація та вміння здійснити декомпозицію складної задачі. Тобто головна проблема на перших порах вивчення програмування для більшості полягає не в тому, що вони не можуть запам'ятати базові правила синтаксису мови програмування чи основні керівні структури (послідовне виконання, розгалуження, цикл) та правила створення власних користувацьких функцій. Проблема в том, що не маючи звички описувати задачу формально, розбиваючи її на прості кроки, важко побачити природній зв'язок між синтаксичними структурами if та while, та безпосереднім рішенням прикладної задачі. Людина в своєму повсякденному житті не мислить структурними конструкціями (принаймні, так для себе не формулює), тому формування навички формалізації задачі, на думку автора, є фундаментальною задачею на перших порах програмування, значно більш важливою, ніж вивчення синтаксису тієї чи іншої мови. Зрозуміло, що найкращих результатів при вивченні програмування за досить короткий строк студент досягне, якщо розуміння основ структурного та системного мислення були закладені ще у середній школі.

Після засвоєння основ програмування: синтаксис, керуючі структури, модульна парадигма, основи об'єктно-орієнтованого підходу – зазвичай переходять до етапу вивчення базових підходів до розробки ефективних алгоритмів розв'язку задачі. Зазвичай, якщо ця проблематика виділена в окрему дисципліну, академічно предмет має назву «Алгоритми та структури даних» (Algorithms and Data Structures). Найвідомішим підручником з цієї тематики є мабуть [5], а також фундаментальний багатотомник «Мистецтво програмування» Д. Кнута. Взагалі література з предмету є досить обширною.

Основними темами вивчення сучасних підходів для розробки власних ефективних алгоритмів є аналіз просторової та часової ефективності (складності) алгоритмів, «жадібні» алгоритми, парадигма декомпозиції «розділяй та володарюй», динамічне програмування тощо. Безумовно, зазначені теми є основою ефективної алгоритмізації, і більшість прикладних проблем можуть бути розв'язані за допомогою зазначених підходів. Але за досвідом автора, більш раціональним є підхід до викладання, коли проблематика складності (ефективності) алгоритмів починається не з згаданих спеціальних розділів алгоритмізації, а з ґрунтовної проробки алгоритмів перебору варіантів, як базового (повного перебору), так і покращених підходів для організації пошуку з поверненням тощо.

З методичної точки зору перебірні задачі є корисними в тому смислі, що більшість складних прикладних задач принаймні на першому етапі – етапі моделювання – намагаються вирішити саме методом перебору. Інколи, задача полягає у генерації (візуалізації) всіх варіантів, а ухвалює рішення про розв'язок безпосередньо користувач. Це важливо для спеціалістів гуманітарної сфери, коли формально описати критерії оптимального розв'язку задачі буває досить складно.

Перебірні задачі також можна назвати комбінаторними, тому що в основі дуже часто лежить необхідність генерації певних комбінаторних об'єктів – перестановок, сполучень, розміщень тощо. Засвоєння алгоритмів формування зазначених об'єктів з методичної точки зору дозволяє ще раз зосередитись на практичному застосуванні рекурсивних функцій, теми, що є незвичною і деякий час досить складною навіть для студентів, для яких програмування є профілюючим предметом.

Після засвоєння базових комбінаторних алгоритмів, логічним є перехід до задачі організації пошуку з поверненням. Тут є можливість розвернутись, маючи в запасі демонстрацію побудови алгоритму розв'язку відомих задач: задача про N ферзів, задача про обхід конем шахової дошки, задача про обхід лабіринту, задача про рішення судоку, генерація греко-латинського квадрату тощо. Підказку для вибору демонстраційних прикладів з тематики програмної організації перебору варіантів та інших комбінаторних алгоритмів можна знайти [6–7].

Початок розв'язку задачі з програмної реалізації перебору варіантів має додатково методичну цінність: демонстрація підходів до підвищення ефективності алгоритмів стає наочною, і проблема складності алгоритму перестає бути суто теоретичною. Оцінюючи час виконання програми повним перебором і за допомогою побудови більш ефективних алгоритмів дає змогу студенту інтуїтивно переконатись у доречності використання більш складних концепцій алгоритмізації: парадигми «розділяй та володарюй», динамічного програмування тощо.

Література

1. Анісімов А. В., Дорошенко А. Ю., Погорілий С. Д., Дорогий Я. Ю. Програмування числових методів мовою Python. К.: Видавничо-поліграфічний центр «Київський університет», 2014. 640 с.
2. Коротєєва Т. О. Алгоритми та структури даних. Львів: Видавництво Львівської політехніки, 2014. 280 с.
3. Креневич А. Алгоритми та структури даних. К.: ВПЦ «Київський Університет», 2018. 172 с.
4. Креневич А.П. Python у прикладах і задачах. Частина 1. Структурне програмування. К.: ВПЦ «Київський Університет», 2017. 206 с.
5. Кормен Т. Г., Лейзерсон Ч. Е., Рівест Р. Л., Стайн К. Вступ до алгоритмів. К.: К.І.С., 2019. С. 1288.
6. Skiena S.S. The Algorithm Design Manual. Springer, 2020. 800 p.
7. Knuth D.E. The Art of Computer Programming. Volume 4A: Combinatorial Algorithms, Part 1. Upper Saddle River, New Jersey: Addison-Wesley, 2011. 883 p.

УДК 517.9

HARNACK'S INEQUALITY FOR QUASILINEAR ELLIPTIC EQUATIONS WITH GENERALIZED ORLICZ GROWTH

M. O. Savchenko (Shan)

In this paper we consider quasilinear elliptic equations of the form

$$div\left(g(x,\nabla u),\frac{\nabla u}{|\nabla u|}\right)=0 \tag{1}$$

where Ω is a bounded domain in в R^n, $n\geq 2.$

We suppose that the function $g(\cdot\,,\mathrm{v}):\Omega\times \mathrm{R}_+\to \mathrm{R}_+$, $\mathrm{R}_+=[0,+\infty)$ satisfies the following assumptions:

(A1) for all $v\in \mathrm{R}_+$, $g(\mathrm{x}\,,\cdot)$ is continuous and non-decreasing for almost all $x\in\Omega$, $\lim\limits_{v\to+0} g(x,v)=0$ and $\lim\limits_{v\to+\infty} g(x,v)=+\infty$;

(A2) there exist $c_1>0,\ q>1$ and $b_0\geq 0$ such that

$$\frac{g(x,\mathrm{w})}{g(x,v)}\leq c_1\left(\frac{w}{v}\right)^{q-1}$$

for all $x\in\Omega$ and for all $w\geq v>b_0$;