МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

В.А. Висоцька, О.В. Оборська

PYTHON: АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

Навчальний посібник

«Новий Світ-2000»

Львів – 2021

УДК 81:004.93

Гриф надано Науково-методичною радою
Національного університету «Львівська політехніка»
(Протокол № 37 від «04» вересня 2018 р.)

Рецензенти:
Бісікало О.В. – доктор технічних наук, професор, декан факультету комп'ютерних систем автоматики Вінницького національного технічного університету;
Винокурова О.А. – доктор технічних наук, професор, головний науковий співробітник Проблемної науково-дослідної лабораторії АСУ Харківського національного університету радіоелектроніки;
Гожий О.П. – доктор технічних наук, професор кафедри комп'ютерної інженерії Чорноморського національного університету імені Петра Могили;
Литвиненко В.І. – доктор технічних наук, професор, завідувач кафедри інформатики та комп'ютерних наук Херсонського національного технічного університету;
Пелешко Д.Д. – доктор технічних наук, професор, проректор з науково- технічної роботи «IT Step University»;
Шаронова Н.В. – доктор технічних наук, професор, завідувач кафедри інтелектуальних комп'ютерних систем Національного технічного університету «Харківський політехнічний інститут»;
Шаховська Н.Б. – доктор технічних наук, професор, завідувач кафедри систем штучного інтелекту Національного університету «Львівська політехніка».

Висоцька В.А., Оборська О.В.

Python: алгоритмізація та програмування: навчальний посібник – Львів: Видавництво «Новий Світ – 2000», 2021. – 514 с.

ISBN 978-617-7519-74-3

Навчальний посібник містить матеріал для вивчення основних теоретичних засад, функціональних можливостей та практичного застосування теорії алгоритмів та основ програмування, розроблення прикладних засобів та інформаційних систем аналізу та опрацювання інформації за допомогою алгоритмів на мові Python. Теоретичний та практичний матеріал викладено у доступній формі. Викладення матеріалу супроводжується значною кількістю прикладів, що полегшує його сприйняття та засвоєння. Подається перелік питань та тестів для самоконтролю, а також завдань для самостійного виконання трьох рівнів складності. Навчальний посібник призначається для студентів, що навчаються за спеціальностями 122 «Комп'ютерні науки», 124 «Системний аналіз», 126 «Інформаційні системи та технології» та споріднених спеціальностей, які пов'язані з інформатикою та інформаційними технологіями. Він може бути використаний аспірантами як підґрунтя для наукових досліджень та викладачами як дидактичний матеріал, а також для самостійного вивчення. Книга призначена для спеціалістів із проектування, розроблення та впровадження інтелектуальних систем опрацювання інформаційних ресурсів, науковців у галузі глобальних інформаційних системи, систем штучного інтелекту, Інтернет-технологій, фахівців з електронної комерції, Інтернет-маркетингу та Інтернет-реклами, менеджерів комплексних Web-проектів, а також для здобувачів 3-ого (освітньо-наукового) рівня вищої освіти в галузі знань 12 «Інформаційні технології». Кожний розділ закінчується переліком питань для самоконтролю, прикладом тестових питань з відповідями та переліком індивідуальних завдань для виконання лабораторних робіт.

ISBN 978-617-7519-74-3

ЗМІСТ

Вступ

Термін “інформація” походить від латинського слова “*informatio*”– виклад, роз’яснення фактів, подій. Тому, в найширшому розумінні інформація трактується як відомості, пояснення, знання про об’єкти, явища та процеси реального світу. Основна маса інформації збирається, передається та опрацьовується у знаковій формі – числовій, текстовій, табличній, графічній і ін. Знакове подання інформації інформатика пов’язує з поняттям “дані”. Дані – це відомості, подані у певній знаковій формі, придатній для передавання, інтерпретації та опрацювання людиною або автоматичним пристроєм.

Взаємозв’язок між інформацією, даними і методами оперування ними характеризується низкою властивостей. Інформація має динамічний характер. Вона не є статичним об’єктом, а динамічно змінюється й існує тільки у момент взаємодії даних і методів, тобто в момент протікання інформаційного процесу. Весь інший час вона перебуває у стані даних. Зв’язок даних і методів носить діалектичний характер. Дані є об’єктивними, коли вони виникають у результаті реєстрації об’єктивно існуючих сигналів, викликаних змінами в матеріальних тілах або полях. У той же час, методи оперування даними в інформаційних процесах є суб’єктивними. Дійсно, інформаційний процес здійснюється за допомогою штучних або природних методів, в основі штучних методів лежать алгоритми, складені і підготовлені людьми (суб’єктами), в основі природних – біологічні властивості суб’єктів інформаційного процесу. Дані надають інформацію лише в момент їх використання. Це означає, що дані, які зберігаються, залишаються лише даними, поки їх не використано в тих чи інших методах деяким суб’єктом інформаційного процесу (людиною або автоматичним пристроєм). Якщо дані зовсім не використовуються, то кажуть, що вони мають нульову інформативність, і вважають такі дані інформаційним шумом. Дані, що використовуються, називають інформативними. Рівень інформативності даних залежить від ступеня адекватності методів, що застосовуються в інформаційних процесах. Дані сприймаються їх отримувачем як певна змістовна інформація лише в тому випадку, коли в його “пам’яті” закладені поняття і моделі, що дають змогу зрозуміти зміст отриманих відомостей. Коли дані опрацьовуються певними методами або евристиками, то на їх основі можна встановити залежності, послідовності чи висновки, які дають змогу здійснити підтримку процесу прийняття рішення. Основними в інформатиці є три поняття: задача, алгоритм, програма. Відповідно, маємо три етапи в розв’язуванні задач (зазначимо, що, з точки зору інформатики, розв’язати задачу – це отримати програму, тобто, забезпечити можливість отримати рішення за допомогою комп’ютера): постановка задачі, побудова і обґрунтування алгоритму, складання і налагодження програми. Оскільки програма – об’єкт гранично формальний, а тому точний (можливо не завжди прозорий, навантажений неістотними із змістовної точки зору деталями, але недвозначний) то, пов’язані з нею об’єкти також мають бути точними. Поняття алгоритму інтуїтивно зрозуміле і часто використовується в математиці та інформаційних технологіях, зокрема в

комп'ютерних науках, системному аналізі та розробленні інформаційних систем.

Термін "*алгоритм*" походить із давніх-давен. Понад тисячу років тому у Багдаді жив Абу Джафар Мухамад ібн Муса аль-Хорезмі, який у своїй праці "Трактат аль-Хорезмі про арифметичне мистецтво індусів" з арифметики та алгебри сформулював правила і окреслив послідовність дій при додаванні та множенні чисел. При переведенні латиною ім'я автора трансформувалося в Algorithmi, а з часом методи розв'язування задач стали називати алгоритмами.

Алгоритм – це система формальних правил, які чітко та однозначно визначають послідовність дій обчислювального процесу від початкових даних до шуканого результату. Алгоритм визначає певні правила перетворювання інформації, тобто визначає певну послідовність операцій опрацювання даних для отримання розв'язку задачі. Також кажуть, що алгоритм – це скінченна послідовність команд, які потрібно виконати над вхідними даними для отримання результату. Побудова алгоритму за точною постановкою задачі дає змогу обґрунтувати його математичними методами. Більше того, існують класи задач, які забезпечують формальне перетворення специфікації в алгоритм. Суттєво, що, порівняно з програмою, алгоритм може перебувати на вищому рівні абстракції, бути вільним від певних деталей реалізації, пов'язаних із особливостями мови програмування та конкретної обчислювальної системи. Засоби для зображення алгоритмів називають алгоритмічною мовою. При цьому жодна мова програмування не може цілком замінити алгоритмічну мову, оскільки консервативна за своєю суттю. Модифікація мови програмування призводить до небажаних наслідків: вимагає перероблення системи програмування, знецінює напрацьоване програмне забезпечення. У той же час алгоритмічна мова може створюватись спеціально для певної предметної області, певного класу задач або навіть окремої задачі.

У цьому посібнику розглядається одна з найбільш розповсюджених мов програмування – мова Phyton. Хоча колись була найпопулярнішою мовою С, створена 1972 року як мова для системного програмування, вона отримала широке поширення у всьому світі. Сьогодні вона має багатьох наступників серед мов програмування: C++, Java, C#, C, Phyton, Scala, JavaScript, Perl тощо.

Поняття алгоритму інтуїтивно зрозуміло та часто використовується в математиці та комп'ютерних науках. Говорячи неформально, **алгоритм** – це довільна коректно визначена обчислювальна процедура, на **вхід** якої подається деяка величина або набір величин, а результатом виконання якої є **вихідна** величина або набір значень. Таким чином, алгоритм є послідовністю обчислювальних кроків, які перетворюють вхідні величини у вихідні. Алгоритм можна також розглядати як інструмент, який призначений для вирішення коректно поставленої обчислювальної задачі. У постановці задачі в загальних рисах визначаються відношення між входом та виходом. В алгоритмі описується конкретна обчислювальна процедура, за допомого якої можна досягнути виконання вказаних відношень. Можна навести загальні риси алгоритму:

а. *Дискретність інформації.* Кожний алгоритм має справу з даними: вхідними, проміжними, вихідними. Ці дані подають у вигляді скінченних слів деякого алфавіту.

b. *Дискретність роботи алгоритму.* Алгоритм виконується по кроках та при цьому на кожному кроці виконується тільки одна операція.

c. *Детермінованість алгоритму.* Система величин, які отримуються в кожний (не початковий) момент часу, однозначно визначається системою величини, які були отримані в попередні моменти часу.

d. *Елементарність кроків алгоритму.* Закон отримання наступної системи величин з попередньої повинен бути простим та локальним.

e. *Виконуваність операцій.* В алгоритмі не має бути не виконуваних операцій. Наприклад, неможна в програмі призначити значення змінній "нескінченність", така операція була би не виконуваною. Кожна операція опрацьовує певну ділянку у слові, яке опрацьовується.

f. *Скінченність алгоритму.* Опис алгоритму повинен бути скінченним.

g. *Спрямованість алгоритму.* Якщо спосіб отримання наступної величини з деякої заданої величини не дає результату, то має бути вказано, що треба вважати результатом алгоритму.

h. *Масовість алгоритму.* Початкова система величин може обиратись з деякої потенційно нескінченної множини.

Розглянемо для прикладу задачу сортування послідовності чисел у зростаючому порядку. Ця задача часто виникає на практиці і, фактично, буде центральною проблемою першого розділу даного курсу. **Задача сортування** визначається формально наступним чином.

Вхід: послідовність *n* чисел $< a_1, a_2, ..., a_n >$

Вихід: перестановка $< a'_1, a'_2, ..., a'_n >$ вхідної послідовності таким чином, що для всіх її членів виконується співвідношення $a'_1 \le a'_2 \le ... \le a'_n$.

Наприклад, якщо на вхід подається послідовність <31, 41, 59, 26, 11, 58>, то вивід алгоритму сортування повинен бути таким: <11, 26, 31, 41, 58, 59>. Подібна вихідна послідовність називається **екземпляром задачі** сортування. Взагалі, екземпляр задачі складається із входу, який необхідний для розв'язання задачі та який задовольняє усім обмеженням, які присутні в постановці задачі. В комп'ютерних науках сортування є основною операцією (у багатьох програмах вона використовується в якості проміжного кроку), в результаті чого з'явилось багато якісних алгоритмів сортування. Вибір найбільш адекватного алгоритму залежить від багатьох факторів, в тому числі й від кількості елементів для сортування, від їх порядку у вхідній послідовності, від можливих обмежень, які накладаються на членів послідовності. Кажуть, що алгоритм є **коректним**, якщо для кожного входу результатом його роботи є коректний вивід. Тоді коректний алгоритм **розв'язує** дану обчислювальну задачу. Якщо алгоритм некоректний, то для деяких входів він може взагалі не завершити свою роботу, або видати відповідь, яка відрізняється від очікуваної.

Перелічимо переваги використання алгоритмізації в програмуванні.

По-перше, алгоритми є життєво необхідними складовими для рішення будь-яких задач з різноманітних напрямків комп'ютерних наук. Алгоритми відіграють ключову роль у сучасному розвитку технологій. Тут можна згадати такі розповсюджені задачі, як:

- розв’язання математичних рівнянь різної складності, знаходження добутку матриць, обернених матриць;
- знаходження оптимальних шляхів транспортування товарів та людей;
- знаходження оптимальних варіантів розподілення ресурсів між різними вузлами (виробниками, верстатами, працівниками, процесорами тощо);
- знаходження в геномі послідовностей, які співпадають;
- пошук інформації в глобальній мережі Інтернет;
- прийняття фінансових рішень в електронній комерції;
- опрацювання та аналіз аудіо та відео інформації.

Цей список можна продовжувати й продовжувати і, власно кажучи, майже неможливо знайти таку галузь комп’ютерних наук та інформатики, де б не використовувались ті або інші алгоритми.

По-друге, якісні та ефективні алгоритми можуть бути каталізаторами проривів у галузях, які є на перший погляд далекими від комп’ютерних наук (квантова механіка, економіка та фінанси, теорія еволюції).

І, по-третє, вивчення алгоритмів це також неймовірно цікавий процес, який розвиває математичні здібності та логічне мислення.

Тепер розглянемо поняття ефективності алгоритму. Припустимо, швидкодія комп’ютеру та об’єм його пам’яті можна збільшувати до нескінченності. Чи була би тоді необхідність у вивченні алгоритмів? Так, але тільки для того, щоб продемонструвати, що метод розв’язку має скінченний час роботи і що він дає правильну відповідь. Якщо б комп’ютери були необмежено швидкими, підійшов би довільний коректний метод рішення задачі. Звісно, тоді найчастіше обирався би метод, який найлегше реалізувати.

Сьогодні є дуже потужні комп’ютери, але їх швидкодія не є нескінченно великою, як і пам’ять. Таким чином, час обчислення – це такий само обмежений ресурс, як і об’єм необхідної пам’яті. Цими ресурсами слід користуватись розумно, чому й сприяє застосування алгоритмів, які ефективні в плані використання ресурсів часу та пам’яті.

Алгоритми, які розроблені для розв’язання однієї та тієї самої задачі, часто можуть дуже сильно відрізнятись за ефективністю. Ці відмінності можуть бути набагато більш суттєвими, ніж ті, які викликані застосуванням різного апаратного та програмного забезпечення.

Для прикладу розглянемо та порівняємо задачі сортування. Перший алгоритм, який буде розглядатись – сортування включенням, для своєї роботи вимагає часу, кількість якого оцінюється як c_1n^2, де n – розмір вхідних даних (кількість елементів у послідовності для сортування), c_1 – деяка стала. Цей вираз вказує на те, як залежить час роботи алгоритму від об’єму вхідних даних. У випадку сортування включенням ця залежність є квадратичною. Другий алгоритм – сортування злиттям – потребує часу, кількість якого оцінюється як $c_2n\log_2n$. Зазвичай константа сортування включенням менше константи сортування злиттям, тобто $c_1<c_2$, але як пересвідчимось у наступних темах, ці константи не відіграють ролі у порівнянні швидкодії різних алгоритмів. Адже зрозуміло, що функція n^2 зростає швидше зі збільшенням n, аніж функція $n\log_2n$.

І для деякого значення $n = n_0$ буде досягнуто такий момент, коли вплив різниці констант перестане мати значення і надалі функція $c_2 n\log_2 n$ буде менша за $c_1 n^2$ для будь-яких $n > n_0$.

Для демонстрації цього розглянемо два комп'ютери – А та Б. Комп'ютер А більш швидкий і на ньому працює алгоритм сортування, а комп'ютер Б більш повільний і на ньому працює алгоритм сортування методом злиття. Обидва комп'ютери повинні виконати сортування множини, яка складається з мільйона чисел. Припустимо, що комп'ютер А виконує мільярд операцій в секунду, а комп'ютер Б – лише десять мільйонів, тобто А працює в 100 разів швидше за Б. Щоб різниця стала більш відчутною, припустимо що код методу включення написаний найкращим програмістом в світі із використанням команд процесору, і для сортування n чисел за цим алгоритмом потрібно виконати $2n^2$ операцій (тобто c_1=2). Сортування методом злиття на комп'ютері Б написано програмістом початківцем із використанням мови високого рівня і отриманий код потребує $50n\log_2 n$ операцій (тобто c_2=50). Таким чином для сортування мільйона чисел комп'ютеру А буде потрібно $\frac{2\left(10^6\right)^2 comands}{10^9 comands/s} = 2000\,s$, а комп'ютеру Б – $\frac{50\left(10^6\right)\log_2 10^6 comands}{10^7 comands/s} \approx 100s$.

Тож, використання коду, час роботи якого зростає повільніше, навіть при поганому комп'ютері та поганому компіляторі потребує на порядок менше процесорного часу! Для сортування 10 мільйонів чисел перевага сортування злиттям стає ще більш відчутною: якщо сортування включенням потребує для такої задачі приблизно 2,3 дня, то для сортування злиттям – менше 20 хвилин. Загальне правило таке: чим більша кількість елементів для сортування, тим суттєва перевага сортування злиттям. Наведений вище приклад демонструє, що алгоритми, як і програмне забезпечення комп'ютеру, являють собою технологію. Загальна продуктивність системи настільки ж залежить від ефективності алгоритму, як і від потужності апаратних засобів.

Тепер визначимо золоте правило розробників алгоритмів. Для цього розглянемо для прикладу просту задачу, яка відома усім ще з початкової школи, а також метод розв'язання цієї задачі – множення двох цілих чисел. Цю задачу можна описати наступним чином:

Вхід: 2 цілих n-розрядних числа x та y

Вихід: добуток чисел $x \cdot y$

Розглянемо приклад для чисел x = 5678 та y = 1234. Результат відомого з дитинства методу множення в стовпчик буде виглядати наступним чином:

```
  5678
 x1234
 22712
 17034
11356
5678
7006652
```

Легко зауважити, що елементарні операції, які тут використовуються, це –

множення та додавання однорозрядних чисел. Припустивши, що операція множення займає більше часу аніж операція додавання для однієї пари чисел, оцінимо кількість таких елементарних операцій. Всі вони виконується в області, яка вище позначена сірим кольором. В даному прикладі кількість елементарних операцій добутку становитиме $16 = 4^2$, а в загальному випадку становитиме n^2. Тож, кількість операцій для добутку двох цілих n-розрядних чисел методом множення у стовпчик оцінюється як *cn2*, де *c* – деяка стала.

Проте чи можемо покращити цей результат, отримавши метод добутку чисел, який буде працювати швидше? Щоб мотивувати себе для пошуку такого методу, наведемо цитату з книги «Розробка та аналіз комп'ютерних алгоритмів» (Аго, Гопкрофт, Ульман, 1974): «Можливо найбільш важливим принципом для гарного розробника алгоритмів є відмова від того, щоб бути задоволеним результатом». Слідуючи цьому правилу, розглянемо ще раз детальніше природу об'єктів задачі добутку чисел. За умовою на вхід подається два n-розрядних числа. Припустимо розіб'ємо кожне число навпіл, отримавши, так звані, верхнє та нижнє півслова. Тобто, можна записати $x=10^{n/2}a+b$ та $y=10^{n/2}c+d$, де a, b, c, d – цілі $n/2$-розрядні числа. Тоді добуток $x \cdot y$ можна подати так:

$$xy=(10^{n/2}a+b)(10^{n/2}c+d)=10^{n}ac+10^{n/2}(ad+bc)+bd.$$

Таким чином природно підійшли до рекурсивного методу обчислення добутку двох цілих чисел, який зводить обрахунок добутку двох n-розрядних чисел до обрахунку чотирьох добутків $n/2$-розрядних чисел. Спробуємо з'ясувати, чи покращиться таким чином швидкість добутку двох чисел. Кожне з чисел a, b, c, d мають $n/2$ розрядів, а відтак добуток будь-якої їх пари (якщо використовувати для нього старий алгоритм множення у стовпчик) займатиме $c(n/2)^2$ операцій, тобто $cn^2/4$. Чотири таких добутки в сумі знову дадуть початковий результат: $4cn^2/4 = cn^2$. Отже, виграш за часом не було отримано. Невже не можливо покращити результат роботи методу множення чисел у стовпчик? Насправді, можливо і відповіддю на це питання є метод множення Карацуби (1960). Якщо подивитись на формулу xy, то зауважимо, що насправді важливими є не чотири добутки, а три: ac, bd та $(ad + bc)$, тобто елементи ad та bc не цікавлять самі по собі, а лише їх сума. Чи можна отримати їх суму перемноживши лише два числа? Так:

$(a+b)(c+d)=ac+ad+bc+bd=(ad+\text{bc})+ac+bd$ та $ad+bc=(a+b)(c+d)-ac-\text{bd}$

Таким чином, суму $(ad+\text{bc})$ можна отримати з добутку двох $n/2$-розрядних числа (можливо, $n/2 + 1$) $(a+b)$ та $(c+d)$, а також добутків ac та bd, які вже маємо. І, отже, кількість рекурсивних викликів скоротились з 4 до 3. Аналіз швидкості методу множення Карацуби приводить до оцінки $3n^{\log_2 3} \approx 3n^{1,585}$. Цей приклад красномовно свідчить, що простір для руху розробника алгоритмів є набагато більшим ніж може видаватись на початку.

Навчальний посібник містить матеріал для вивчення основних теоретичних засад, функціональних можливостей та практичного застосування теорії алгоритмів та основ програмування, розроблення прикладних засобів та інформаційних систем аналізу та опрацювання інформації за допомогою алгоритмів. Викладення матеріалу супроводжується значною кількістю прикладів, що полегшує його сприйняття та засвоєння. Навчальний посібник

призначається для студентів, що навчаються за спеціальностями 122 «Комп'ютерні науки», 124 «Системний аналіз» та 126 «Інформаційні системи та технології» і споріднених спеціальностей, які пов'язані з вивченням чисельних методів в інформатиці та інформаційних технологій. Може бути використаний аспірантами як підґрунтя для наукових досліджень та викладачами як дидактичний матеріал, а також для самостійного вивчення та підвищення кваліфікації. Книга призначена для спеціалістів із проектування, розроблення та впровадження інтелектуальних систем опрацювання інформаційних ресурсів, науковців у галузі глобальних інформаційних системи, систем штучного інтелекту, Інтернет-технологій, фахівців з електронної комерції, Інтернет-маркетингу та Інтернет-реклами, менеджерів комплексних Web-проектів, а також для здобувачів 3-го (освітньо-наукового) рівня вищої освіти в галузі знань 12 «Інформаційні технології». Кожний розділ закінчується переліком питань для самоконтролю, прикладом тестових питань з відповідями до екзаменаційних білетів та переліком індивідуальних завдань для виконання лабораторних робіт.

Розділ 1 присвячений основам програмування, елементам обчислювальних машин та описує архітектуру комп'ютерів. Детально пояснені та описані архітектурні принципи Джона фон Неймана, машинні та високорівневі мови програмування, парадигми програмування. Приділено увагу класифікації мов програмування, а саме пояснені відмінності між системним програмуванням, структурним та процедурним програмування, модульним та об'єктно-орієнтованим програмування, а також функційним програмуванням. В розділі подано пояснення щодо середовища програмування та програмного забезпечення для Python, а саме робота в інтерактивному режимі, створення скриптів, особливості математики на Python, порядок операцій та використання коментарів в коді програми на Python. Також описано особливості написання програм на Python, пояснені етапи створення програми у файлі на Python, редагування файлів, запуск програм з файлу та робота з функцією input.

Розділ 2 описує позиційні системи числення (двійкова, вісімкова, шістнадцяткова система числення, та взаємозв'язок між ними). В третьому розділі описані типи даних в мові програмування Python (булеві, цілі, дійсні та рядки), змінні, екрановані послідовності та їх форматування.

Розділ 4 присвячений основам алгоритмізації, зокрема, поняттям алгоритму та способи його подання, властивостям та класам алгоритмів. Описано структуру модуля в Python, багатомодульні програми, помилки, техніку налагодження (зневаджування) програм та розроблення алгоритму програми. Наведені приклади бібліотек Python як Matplotlib, NetworkX, csv, NumPy тощо.

Розділ 5 описує особливості побудови логічних виразів на Python, а розділ 6 присвячений циклам while та for в мові Python, а також розкриває важливість відступів у блоковому програмуванні. Робота з файлами описано в розділі 7. Розділ 8 описує особливості роботи з кортежами, списками та словниками. В розділі 9 розкриті питання роботи зі стеком, чергою та деком. Робота з функціями та рекурсією подана в розділі 10. Розділ 11 присвячений питанням роботи з графікою через бібліотеку turtle.

Алгоритми пошуку описані детально в розділі 12, зокрема: лінійний пошук,

двійковий (бінарний) пошук елемента в масиві, пошук методом Фібоначчі, інтерполяційний пошук. Також цей розділ присвячений питанням хешування. Розділ 13 описує відомі алгоритми сортування, а саме: бульбашки, шейкерне, вставкою, метод Шелла, метод Хоара тощо. Розділ 14 описує особливості програмування з використанням класів, розділ 15 – графіка та створенні ігор через бібліотеку tkinter. Створення ігор використанням бібліотеки pygame присвячено розділ 16, роботі з масивами через бібліотеку NymPy – розділ 17. Розділ 18 присвячений особливостям роботи на Python для Web через CGI.

Автори цього навчального посібника висловлюють щиру подяку фахівцям в галузі знань 12 “Інформаційні технології” за плідну співпрацю, вагомі поради та слушні зауваження при формуванні структури книги та її наповнення. Зокрема, дякуємо за підтримку:

Бісікалу О.В. – д.т.н., професору, завідувачу кафедри автоматики та інформаційно-вимірювальної техніки Вінницького національного технічного університету;

Винокуровій О.А. – д.т.н., професору, головному науковому співробітнику Проблемної науково-дослідної лабораторії АСУ Харківського національного університету радіоелектроніки;

Пелешку Д.Д. – доктору технічних наук, професору, проректору з науково-технічної роботи «IT Step University»;

Гожому О.П. – д.т.н., професору кафедри комп’ютерної інженерії Чорноморського національного університету імені Петра Могили.

Литвиненку В.І. – д.т.н., професору, завідувачу кафедри інформатики і комп’ютерних наук Херсонського технічного університету;

Литвину В.В. – д.т.н., професору, завідувачу кафедри інформаційних систем та мереж Національного університету «Львівська політехніка»;

Шароновій Н.В. – д.т.н., професору, завідувачу кафедри інтелектуальних комп’ютерних систем Національного технічного університету «ХПІ»;

Шаховській Н.Б. – д.т.н., професору, завідувачу кафедри систем штучного інтелекту Національного університету «Львівська політехніка»;

Кравцю П.О. – к.т.н., доценту кафедри інформаційних систем та мереж Національного університету «Львівська політехніка»;

Чируну Л.В. – к.т.н., доценту кафедри програмного забезпечення Львівського Національного університету імені Івана Франка;

Ришковцю Ю.В. – к.т.н., доценту кафедри інформаційних систем та мереж Національного університету «Львівська політехніка»;

Голощуку Р.О. – к.т.н., доценту кафедри соціальних комунікацій та інформаційної діяльності Інституту гуманітарних та соціальних наук Національного університету «Львівська політехніка».