Висновки. У дослідженні систематизовано інформацію про ігри з нульовою сумою та перелічено її характерні ознаки. Проаналізовано умови, за яких торгівля криптовалютою може бути грою з нульовою сумою, залежно від типу угод, які здійснює користувач. Оскільки торгівля деривативами вважається грою з нульовою сумою, а приблизно 50 % обсягів на криптобіржах припадає на торгівлю деривативами, можна стверджувати, що криптовалюта часто розглядається як гра з нульовою сумою. Однак, якщо інвестори не використовують кредитне плече та обирають для інвестицій якісні проєкти, це стає безпрограшною ситуацією, де жодна зі сторін не зазнає повних збитків.

Висловлено думку, що уявлення про економіку з нульовою сумою є помилковим, яку економісти вже давно спростували. Насправді ресурси не обмежені, і існують численні сценарії, за яких усі сторони можуть отримати вигоду. Підхід з нульовою сумою часто використовують для обґрунтування протекціонізму та торгових бар'єрів, хоча насправді країни можуть вигравати від співпраці у сфері торгівлі. Людські взаємодії складні, і часто люди об'єднують зусилля для досягнення спільних цілей.

Abstract. The article examines the application of game theory in solving economic problems. It is stated that game theory provides tools for analyzing situations where different parties with conflicting interests interact. The conditions under which cryptocurrency trading can be considered a zero-sum game, depending on the type of transactions carried out by the user, are analyzed. It is argued that the concept of a zero-sum economy is a misconception that economists have long debunked. In reality, resources are not limited, and there are numerous scenarios where all parties can benefit.

Keywords: game theory, player strategy, payoff matrix, cryptocurrency.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Коломієць Г. Застосування теорії ігор в оподаткуванні як сфері узгодження суспільних і приватних інтересів. *Вісник Хмельницького національного університету.* 2020. Вип. 4(3). С. 202–205.

2. Цимбалюк І. О. Обґрунтування підприємницьких рішень та оцінка ризиків: конспект лекцій. ОНП Економіка сталого розвитку. Луцьк: ВНУ ім. Лесі Українки, 2022. 130 с.

3. Angelini P. Financial Decisions Based on Zero-Sum Games: New Conceptual and Mathematical Outcomes. *International Journal of Financial Studies.* 2024. Vol. 2:56. DOI: 10.3390/ijfs12020056.

4. A Comprehensive Insight into Game Theory in relevance to Cyber Security / F. Anwar et al. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI).* 2020. Vol. 8, 189–203. DOI: 10.11591/ijeei.v8i1.1810.

5. Khalid M., Al-Kadhimi A., Singh M. Recent Developments in Game-Theory Approaches for the Detection and Defense against Advanced Persistent Threats (APTs): A Systematic Review Mathematics. 2023. Vol. 11, 1353. DOI: 10.3390/math11061353.

УДК 004:005:51

ОСОБЛИВОСТІ АНАЛІЗУ ПРОДУКТИВНОСТІ АЛГОРИТМІВ ПОШУКУ

М. О. Нестерук, Н. А. Потапова

Анотація. Стаття присвячена питанням аналізу алгоритмів пошуку та їх використанню в практичних задачах. Висвітлюється теоретичний складник операції пошуку та основні категорії теорії алгоритмів пошуку. Розглянуто лінійний та бінарний алгоритми пошуку та оцінку їх складності. Доведено, що найбільш продуктивним є алгоритм бінарного пошуку.

Ключові слова: алгоритм, пошук, складність, продуктивність алгоритму, лінійний пошук, бінарний пошук.

Вступ. Майже у кожному ІТ-проєкті з розробки програмного забезпечення, який вирішує будь-яку проблематику, застосовуються алгоритми з пошуку. Вони є невід'ємною частиною більшості програмних рішень. Для програмної реалізації використовують різні алгоритми пошуку, механізм використання яких залежить від структур даних та способу звернення в ній до складників даних. Під час розробки програмного забезпечення використовують базові алгоритми пошуку, модифікуючи їх окремі блоки.

Метою статті є вирішення задачі порівняння роботи різних алгоритмічних структур пошуку з метою визначення найбільш продуктивного алгоритму.

Основна частина. Однією з найбільш важливих характеристик ІТ-проєкту є його оптимізація, тобто досягнення найменшого часу виконання за мінімальної складності операцій. Проб-

лема оптимізації впливає на продуктивність програмного забезпечення, а в масштабах глобальних проєктів стає джерелом ризиків зниження працездатності. Розуміння структур даних та способів взаємодії з ними є ключовими під час оптимізації проєкту. Алгоритми пошуку є базовими в теорії алгоритмів і характеризують можливість проведення пошукової процедури в межах встановлених параметрів. Основними поняттями, які використовують в алгоритмізації пошуку, є:

Алгоритм пошуку – це послідовність дій, головна мета яких – знайти заданий елемент у певній структурі даних.

Структура даних – це спосіб організації даних для подальшої обробки їх значень або іншого використання. Розглядають такі основні структури даних: масиви, списки, графи та дерева.

Масив – впорядкований набір однотипних даних із фіксованою кількістю елементів.

Зв'язані списки – це структура даних, у якій дані розташовані не упорядковано, але у кожного елементу є інформація про розташування наступного, що дає змогу зв'язати ці дані та взаємодіяти з ними.

Структури даних мають відповідні характеристики, що обумовлюють порядок елементів у них та спосіб звернення до них. Для статичних структур, як-от масиви, найпопулярнішими алгоритмами є алгоритм лінійного пошуку та алгоритм бінарного пошуку. У межах цієї статті нами розглядаються алгоритми пошуку тільки для статичних структур даних.

Оптимальність алгоритму залежить від багатьох параметрів, як-от розмірність сформульованої задачі, кількість і тип змінних, технічні умови виконання алгоритму, характеристики структур даних. Важливо зазначити, що під час розгляду кожного з алгоритмів нами оцінено його за стандартними методами оцінки алгоритмів (O(x), Big O). Це оцінка асимптотичної складності алгоритму. Асимптотична складність визначається функцією, яка показує, наскільки погіршується виконання алгоритму з ускладненням поставленого завдання. Інакше кажучи, вона показує, скільки одиниць часу знадобиться в гіршому випадку для виконання алгоритму. Big (O) – метрика, яка показує верхню межу динаміки обчислювальної складності алгоритму залежно від розміру вхідних даних N. Оцінка складності алгоритму є важливим аспектом під час вибору алгоритму для певної ситуації, і це може бути окремою темою, але в межах цієї статті окреслимо декілька випадків [2]:

– O(1) – це статична оцінка, в якій час виконання алгоритму не залежить від обсягу даних, над якими проводяться операції. В більшості випадків це отримання даних з будь-якої структури даних за ключем.

– O(n) – це оцінка складності алгоритму, в якому час виконання прямо пропорційно залежить від кількості елементів, над якими виконуються операції.

– O(log n) – це оцінка складності алгоритму, в якому частіше за все метод обробки даних пов'язаний з розділенням даних навпіл на кожній з ітерацій.

Розглянемо структури масив та список:

– Масив – структура даних, яка має статичний розмір, а елементи в комірках пам'яті розташовані один поза одним. Під час додавання елементів створюється новий масив, тому краще підходить для даних, які не змінюються з часом. Масив є однорідною структурою, оскільки його елементи мають однаковий тип.

– Список – це більш гнучка структура даних, у якій розташування елементів у пам'яті не мають різниці. В кожному елементі зберігається інформація про кожен наступний елемент, що дає змогу швидко розширювати або змінювати його.

Програмна реалізація алгоритмів нами проводилась за допомогою мови програмування C# та компілятора .NET 8, але ці алгоритми не прив'язані до конкретної мови програмування та можуть буде реалізовані на будь-якій іншій. Кожен алгоритм має свою оцінку складності.

Найпростішим алгоритмом пошуку є ***лінійний алгоритм.*** Суть цього методу в тому, що під час пошуку елементу ми рухаємося по всіх елементах даних, і на кожній ітерації перевіряємо, чи збігається елемент із шуканим. Цей метод реалізує проходження масиву, поетапно збільшуючи ту його частину, в якій шуканий елемент буде відсутній за результатом перевірки. Якщо збігається, то повертаємо індекс цього елементу. Оцінюючи його, можна сказати, що у нього

складність O(n), тобто час, витрачений на пошук елементу, буде прямо пропорційно збільшуватись зі збільшенням розміру даних, серед яких ми будемо намагатися знайти елемент.

Розглянемо реалізацію цього методу. На вході пошуку задано масив чисел:

```
int[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Зробимо припущення про пошук елементу 5, тоді метод для пошуку елементу матиме вигляд:

```
int el = 5;
for (int i = 0; i < array.Length; i++)
{
if (array[i] == el)
{
return i;
}
}
```

Результат буде 4 за умови часу виконання 00:00:00.0002173 мс.

Навіть якщо ми будемо збільшувати масив, то час пошуку цього елементу не збільшиться. Але недолік цього елементу проявиться тоді, коли ми будемо шукати елемент, який знаходиться близько до кінця масиву. Наприклад, під час пошуку 9-го елементу отримаємо результат: 9 за часу виконання 00:00:00.0004207 мс. У цьому випадку час збільшився удвічі. І так буде кожного разу, коли індекс шуканого елементу буде збільшуватись. Перевагою цього методу є те, що для виконання не потрібно попередньої обробки даних. У цьому випадку різниці у швидкості між масивом та списком майже немає, тому що ми проходимо по всіх елементах підряд. Варто зазначити, що в деяких реалізаціях списків вони можуть працювати трохи швидше через внутрішню оптимізацію, але на загальний випадок це не впливає.

Бінарний алгоритм пошуку трохи складніший, але у разі великій кількості даних він має велику перевагу у швидкості. Перед початком цього алгоритму треба переконатися, що дані відсортовані.

Далі алгоритм вибирає елемент із середини масиву і порівнює його з шуканим. Якщо поточний елемент менше шуканого, то він відкидає праву частину даних і переходить до лівої частини. Далі ця операція повторюється, доки шуканий елемент не буде знайдений. На відміну від лінійного, складність цього алгоритму буде залежати від структури даних, яку ми будемо використовувати для зберігання даних. Наприклад, для масиву складність буде O(log n), що набагато краще, ніж у лінійного алгоритму, але все зміниться, якщо ми будемо працювати зі списками. У списках елементи не розташовані один за одним у пам’яті, і для того, щоб переглянути елемент усередині цього списку, нам потрібно отримати поступово доступ до цього елементу. В такому разі всі переваги цього алгоритму втрачаються, а складність буде, як і у лінійного алгоритму, O(n). Метод, який реалізує бінарний алгоритм, має вигляд:

```
int left = 0;
int right = array.Length - 1;
while (left <= right)
{
  int cursor = (right + left) / 2;
  if (array[cursor] == el)
  {
    return cursor;
  }
  else if (array[cursor] < el)
  {
    left = cursor + 1;
  }
  else
  {
    right = cursor - 1;
  }
}
```

Проведемо аналіз алгоритму. Наприклад, на вході заданий масив даних:

```
int[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 }.
```

Як і у попередньому лінійному алгоритмі, спробуємо шукати 9-й елемент. Тоді отримаємо результат: 9 за умови часу виконання 00:00:00.0002716 мс. Отриманий результат вже кращий, ніж у прикладі лінійного пошуку, але найцікавіше буде, якщо ми будемо шукати елемент, розташований ближче до кінця масиву. Наприклад, якщо шукати елемент 20, тоді результат буде такий: 19 за умови часу виконання 00:00:00.0002776 мс.

Проведемо експеримент із більшою кількістю даних у масиві. Згенеруємо масив із 10 000 елементів:

```
var array = Enumerable.Range (0, 10000).ToArray();
```

Під час спроби пошуку 1000-го елементу отримано результат: 1 000 за умови часу 00:00:00.0002873 мс. Час очікувано збільшився, але на відміну від лінійного пошуку, він збільшився на незначну частку. Варто взяти до уваги, що час заміряти важко, і він відносний, оскільки на нього впливають багато факторів: потужність пристрою та оперативні процеси (які майже неможливо контролювати). Під час виміру часу було отримано декілька поточних значень і фіксувалося їх середнє з метою підтвердити різницю швидкості алгоритмів.

Висновки. Отже, кожен алгоритм має своє призначення. Великі об’єми даних від початку варто зберігати впорядковано, і потім під час пошуку конкретного елементу можна застосувати бінарний алгоритм, який заощадить багато часу та потужності. Лінійний пошук підходить краще для невеликих об’ємів даних, де швидкістю можна знехтувати.

Abstract. The article addresses the analysis of search algorithms and their application in practical tasks. It highlights the theoretical aspects of search operations and the main categories of search algorithm theory. The article examines linear and binary search algorithms and evaluates their complexity. It is proven that the most efficient algorithm is the binary search algorithm.

Keywords: algorithm, search, complexity, algorithm efficiency, linear search, binary search.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bhargava A. Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People. Manning Publications, 2016. 256 p.
2. Skiena S. S. The Algorithm Design Manual / Springer; 3rd edition (October 6, 2020). 810 p.
3. Korman T. H. Algorithms. Unlocked. The MIT Press, Massachusetts Institute of Technology, 2013. 207 p.
4. Stephens R. Essential Algorithms: A Practical Approach to Computer Algorithms Using. John Wiley &Sons, 2013. 624 p.
5. Sedgewick R., Wayne K. Algorithms. Addison-Wesley Professional, 2020. 956 p.

УДК 574:59

ОРНІТОФАУНА ПАРКУ-ПАМ’ЯТКИ САДОВО-ПАРКОВОГО МИСТЕЦТВА «ПАРК ІМ. О. І. ЮЩЕНКА» В М. ВІННИЦІ

В. О. Павленко

Анотація. У статті представлено результати досліджень орнітофауни на території парку-пам’ятки садово-паркового мистецтва місцевого значення «Парк ім. О. І. Ющенка». Публікується список орнітофауни, який включає 36 видів. Підтверджено домінування представників дендрофільного угрупування птахів, що є закономірним для рельєфу й рослинного покриву території досліджень. Дослідження є складником комплексної оцінки стану біорізноманіття об’єкта природно-заповідного фонду, яка необхідна для реалізації «Зеленого курсу Вінниці» – стратегії управління природними ресурсами міської територіальної громади.

Ключові слова: орнітофауна, дендрофіли, парк-пам’ятка садово-паркового мистецтва.

Стратегія розвитку Вінницької міської територіальної громади орієнтована на інтеграцію до Європейського Союзу, що відображено, зокрема, у Декларації про Зелений курс Вінниці, яка, зокрема, передбачає: «Збереження та збільшення міських зелених зон, включення питань охорони та збереження міського біорізноманіття в міське планування … має стати важливим елементом Зеленого курсу Вінниці» [1]. Дослідження біорізноманіття об’єктів природно-заповідного фонду міста необхідні як основа екологічного планування.