

## ОПТИМІЗАЦІЯ АЛГОРИТМІВ СОРТУВАННЯ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖ

Вінницький національний технічний університет

### Анотація

У цій роботі розглядається підхід до оптимізації алгоритму QuickSort шляхом використання штучних нейронних мереж. Оскільки продуктивність QuickSort значною мірою залежить від вибору опорного елемента (pivot), запропоновано підхід, у якому нейромережа прогнозує найкращий pivot на основі характеристик масиву. Для цього використовується багатошаровий перцептрон (MLP), навчений на вибірці масивів різної структури. Проведено експериментальне порівняння стандартного QuickSort і його модифікованої версії із нейромережею. Аналіз результатів показав, що використання нейромережі може покращити стабільність алгоритму, особливо для масивів зі складною структурою. Водночас метод має обмеження через додаткові витрати часу на навчання та прогнозування. Результати дослідження демонструють перспективність використання нейромереж у задачах сортування, а також відкривають можливості для подальшої оптимізації алгоритму, зокрема, через адаптивне управління рекурсивними викликами.

**Ключові слова:** алгоритми сортування, швидке сортування, нейромережі, оптимізація алгоритмів, опорний елемент, багатошаровий перцептрон.

### Abstract

This paper addresses the problem of optimizing the QuickSort algorithm using artificial neural networks. Since QuickSort's performance heavily depends on the choice of the pivot element, a novel approach is proposed in which a neural network predicts the optimal pivot based on array characteristics. A multilayer perceptron (MLP) is trained on a dataset of arrays with various structures to achieve this. An experimental comparison between the standard QuickSort and its modified version with a neural network is conducted. The analysis of the results shows that using a neural network can improve the algorithm's stability, particularly for arrays with complex structures. However, the method has limitations due to additional computational costs for training and prediction. The study results highlight the potential of neural networks in sorting tasks and open avenues for further algorithm optimization, particularly through adaptive management of recursive calls.

**Keywords:** sorting algorithms, quicksort, neural networks, algorithm optimization, pivot element, multilayer perceptron (MLP).

### Вступ

Алгоритми сортування відіграють ключову роль у комп'ютерних науках та широко використовуються у практичних застосуваннях, таких як обробка великих даних, бази даних та машинне навчання. Одним із найефективніших алгоритмів сортування є QuickSort, який забезпечує середню часову складність  $O(n \log n)$  та добре підходить для сортування великих обсягів даних.

QuickSort базується на рекурсивному розділенні масиву на дві частини відносно певного опорного елемента (pivot). Від вибору цього елемента значною мірою залежить ефективність алгоритму. Наприклад, у найкращому випадку (коли pivot обирається оптимально) QuickSort працює за  $O(n \log n)$ , тоді як у найгіршому випадку (коли pivot постійно обирається як найбільший або найменший елемент) його складність зростає до  $O(n^2)$ .

Традиційні методи вибору pivot, такі як вибір першого/останнього елемента, випадковий вибір, або медіана з трьох, не завжди гарантують оптимальну продуктивність, особливо для масивів зі складною структурою. У цій роботі пропонується підхід, що використовує штучну нейромережу для прогнозування найкращого pivot, враховуючи характеристики масиву.

Використання нейромереж у цій задачі дозволяє аналізувати структуру масиву та адаптивно обирати опорний елемент, що може зменшити ризик потрапляння у найгірший сценарій. Для реалізації використовується багатошаровий перцептрон (MLP), який навчається на великій вибірці масивів різної структури.

## Класичний алгоритм QuickSort

Алгоритм QuickSort є одним із найшвидших та найефективніших методів сортування, який широко використовується в комп'ютерних науках. Його основний принцип базується на стратегії "розділяй і володарюй", де масив рекурсивно ділиться на дві частини відносно опорного елемента (pivot). У підсумку, після декількох ітерацій такого поділу, масив стає повністю відсортованим.

Робота алгоритму починається з вибору pivot, після чого всі елементи, менші за нього, переміщуються в ліву частину масиву, а більші — у праву. Далі до кожної з отриманих частин застосовується той самий алгоритм, поки довжина підмасивів не стане рівною одиниці. Цей підхід забезпечує ефективну обробку даних із середньою складністю  $O(n \log n)$ .

Однак вибір pivot суттєво впливає на продуктивність алгоритму. У найкращому випадку, коли pivot щоразу розбиває масив на дві рівні частини, QuickSort працює за  $O(n \log n)$ , що є оптимальним показником. У середньому випадку, навіть при нерівномірному поділі, продуктивність залишається близькою до  $O(n \log n)$ . Проте, у найгіршому сценарії, якщо pivot обирається таким чином, що одна з частин масиву залишається майже порожньою (наприклад, якщо масив уже відсортований і pivot завжди є найбільшим або найменшим елементом), алгоритм деградує до  $O(n^2)$ , що робить його повільним для великих наборів даних.

Для покращення ефективності традиційно використовуються різні стратегії вибору pivot, зокрема рандомізація та медіана з трьох елементів. Рандомізація допомагає уникнути найгіршого випадку, зменшуючи ймовірність вибору невдалого pivot. Медіана з трьох передбачає вибір pivot як середнього значення серед першого, останнього та середнього елемента масиву, що також сприяє кращому балансуванню розбиття. Проте, навіть ці підходи не завжди гарантують оптимальну ефективність у всіх можливих ситуаціях.

Таким чином, хоча QuickSort залишається одним із найкращих алгоритмів сортування, його продуктивність значною мірою залежить від вибору pivot. Це відкриває можливість застосування нових підходів до оптимізації, зокрема використання нейромереж, які здатні аналізувати структуру масиву та прогнозувати найкращий опорний елемент для сортування.

## Оптимізація QuickSort за допомогою нейромереж

Одним із ключових аспектів оптимізації алгоритму QuickSort є вибір опорного елемента (pivot), який визначає ефективність розбиття масиву. Невдалий вибір може суттєво знизити продуктивність алгоритму, особливо у випадку вже впорядкованих або частково впорядкованих масивів. Традиційні підходи, такі як випадковий вибір pivot чи медіана з трьох, не завжди забезпечують найкращу продуктивність, оскільки вони не враховують статистичні властивості конкретного масиву.

Для вирішення цієї проблеми пропонується використовувати штучну нейронну мережу, яка навчається на великій кількості різних масивів і прогнозує, який елемент доцільно використовувати як pivot. Такий підхід дозволяє алгоритму адаптуватися до структури вхідних даних та вибирати найбільш ефективний опорний елемент для кожного конкретного випадку.

Нейромережа отримує на вхід характеристики масиву, такі як розподіл елементів, середнє значення, стандартне відхилення та рівень впорядкованості. На основі цих параметрів вона вчиться визначати, який елемент найкраще підходить на роль pivot. Для реалізації використовується багатосаровий перцептрон (MLP), який добре підходить для вивчення взаємозв'язків між даними та прогнозування значення на основі вхідних ознак.

Процес навчання моделі передбачає використання вибірки масивів різної структури, включаючи випадкові, впорядковані, частково впорядковані та обернено впорядковані масиви. Це дозволяє нейромережі адаптуватися до різних сценаріїв та ефективно визначати pivot.

Далі представлено код реалізації алгоритму QuickSort з використанням нейромережі для вибору оптимального опорного елемента (pivot):

```
def quicksort_nn(arr):
    if len(arr) <= 1:
        return arr
    pivot = neural_network_pivot(arr)
    less = [x for x in arr if x < pivot]
    greater = [x for x in arr if x > pivot]
    return quicksort_nn(less) + [pivot] + quicksort_nn(greater)
```

## Висновок

Використання нейромереж у алгоритмах сортування, зокрема в модифікації алгоритму QuickSort, значно покращує стабільність та продуктивність. Прогнозування оптимального опорного елемента (pivot) на основі характеристик масиву дозволяє уникнути ситуацій, коли алгоритм працює в найгіршому випадку ( $O(n^2)$ ), що є важливою перевагою для великих або складних наборів даних. Використання нейромережі забезпечує адаптивний підхід до вибору pivot, що покращує ефективність сортування та стабільність результатів.

Цей підхід має перспективи для застосування в інших алгоритмах сортування, де вибір ключових елементів або стратегії розбиття може впливати на загальну продуктивність. Окрім вибору pivot, нейромережі можуть бути корисними для оптимізації інших аспектів алгоритмів, таких як вибір порядку рекурсивних викликів або балансування підмасивів. Це відкриває нові можливості для вдосконалення класичних алгоритмів сортування, зокрема через автоматичне коригування їх поведінки на основі аналізу структури вхідних даних.

У майбутньому, подальша оптимізація алгоритмів сортування може включати використання нейромереж для прогнозування не лише pivot, а й найбільш ефективного порядку рекурсивних викликів. Це дозволить створювати адаптивні алгоритми, які динамічно налаштовуватимуть свою поведінку в залежності від типу та структури вхідних масивів, що зробить процес сортування ще більш ефективним і надійним.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Швидке сортування — Wikipedia. URL: <https://en.wikipedia.org/wiki/Quicksort>
2. Швидке сортування: принцип роботи та оптимізація. EPAM. URL: <https://campus.epam.ua/ua/blog/483>
3. Швидке сортування: історія виникнення та розвитку найшвидшого алгоритму сортування. URL: <https://phm.cuspu.edu.ua/nauka/naukovo-populiarni-publikatsii/824-quicksort-istoriia-vynyknennia-ta-rozvytku-naishvydshoho-alhorytmu-sortuvannia.html>
4. Швидке сортування з використанням нейромереж. Київський політехнічний інститут. URL: <https://ela.kpi.ua/bitstreams/b8d4c18b-4c38-4003-98c7-0ec1874f13dc/download>
5. Павлюк О. О. Алгоритми сортування за допомогою штучного інтелекту. Магістерська робота. Київський політехнічний інститут, 2021. URL: [https://ai.kpi.ua/ua/masters/thesis/28521smai-pavliuk\\_magistr.pdf](https://ai.kpi.ua/ua/masters/thesis/28521smai-pavliuk_magistr.pdf)

**Возняк Андрій Олександрович** – студент групи 2KI-24mc2, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: [vozniak.andrii01@gmail.com](mailto:vozniak.andrii01@gmail.com).

**Добровольська Наталія Вікторівна** – доцент кафедри обчислювальної техніки Вінницького національного технічного університету, м. Вінниця, e-mail: [natali0212@ukr.net](mailto:natali0212@ukr.net).

**Andrii Oleksandrovych Vozniak** - student of group 2CI-24ms2, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: [vozniak.andrii01@gmail.com](mailto:vozniak.andrii01@gmail.com).

**Dobrovolska Nataliia**- Associate Professor of Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e - mail: [natali0212@ukr.net](mailto:natali0212@ukr.net).