ПРИКЛАДНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК,
БЕЗПЕКИ ТА МАТЕМАТИКИ, 2025, №

УДК 004.42:004.8

ДОСЛІДЖЕННЯ ТА ВИКОРИСТАННЯ ДЕЯКОГО АЛГОРИТМУ ШІ ПРИ ПРОГРАМУВАННІ ГРИ МОВОЮ PYTHON

Глинчук Л.Я. (ORCID: 0000-0002-8943-9604)
Шевчук І.С.
Волинський національний університет імені Лесі Українки, Луцьк, Україна

RESEARCH AND APPLICATION OF A SPECIFIC AI ALGORITHM IN PYTHON GAME DEVELOPMENT

Hlynchuk L. Ya.
Shevchuk I. S.
Lesya Ukrainka Volyn National University, Lutsk, Ukraine

Abstract. This paper investigates the theoretical foundations and practical implementation of the minimax algorithm as a classical artificial intelligence method for decision-making in deterministic, turn-based games with complete information. The study focuses on the development of a Tic-Tac-Toe game with an AI opponent implemented in the Python programming language. Particular attention is paid to the analysis of minimax algorithm modifications, including pruning techniques such as alpha–beta pruning, aimed at reducing computational complexity while preserving optimal decision quality. A modular game architecture is proposed, ensuring code clarity, extensibility, and ease of experimentation. The minimax algorithm is implemented and adapted to support controlled non-optimal behavior, enabling a balanced gameplay scenario where the AI does not always guarantee a win. An experimental evaluation was conducted in a human-versus-AI setting, where the AI agent and human participants competed under predefined conditions. The results demonstrate that the modified minimax algorithm achieves approximately equal win rates for both sides while maintaining acceptable response time and search efficiency. The findings confirm that the minimax algorithm, combined with pruning techniques and behavioral modifications, is an effective and illustrative tool for studying artificial intelligence concepts, game strategy modeling, and algorithmic decision-making in educational and applied contexts.

Keywords: artificial intelligence; minimax algorithm; game AI; Tic-Tac-Toe; Python programming; decision-making algorithms; game strategy.

Вступ. Сучасне програмування штучного інтелекту (ШІ) у настільних іграх є важливою складовою навчання алгоритмічному мисленню та оптимізації обчислювальних моделей. Гра «Хрестики-нулики» (Tic-Tac-Toe) – класичний приклад ігрової задачі з повністю визначуваною інформацією, яка використовується як початковий кейс для розробки та тестування стратегій штучного інтелекту та алгоритмів пошуку. Одним із базових підходів для побудови ігрового AI є алгоритм Minimax, що дозволяє обчислювати оптимальні ходи шляхом аналізу ігрового дерева станів [1; 8]. Незважаючи на те, що minimax є класичним алгоритмом, сучасні дослідження все ще спрямовані на його вдосконалення, порівняння з іншими методами (наприклад, із pruning-техніками) та адаптацію для реальних реалізацій у високорівневих мовах програмування, таких як Python [2; 4; 11].

Python, завдяки своїй читабельності й широкій екосистемі бібліотек, став домінуючою мовою для впровадження та експериментального тестування алгоритмів штучного інтелекту, включно з реалізаціями ігрових алгоритмів [5; 6; 7]. Застосування minimax-алгоритму в Tic-Tac-Toe у Python не лише сприяє кращому розумінню принципів розумного пошуку, але й надає практичні навички створення ігрових додатків із елементами AI, що має значення як у навчальному, так і в практичному контексті [8; 9].

В огляді наукових робіт встановлено, що алгоритм minimax є широко застосовуваним у дослідженнях, пов'язаних з побудовою ігрового AI. Зокрема, Chakole

et al. дослідили формування оптимальної стратегії для Tic-Tac-Toe на основі minimax та описали її реалізацію [1]. У дослідженні Spulber представлено інтерактивну систему для гри Tic-Tac-Toe, що базується на minimax, із акцентом на реальному часі та порівнянні ефективності [2]. Інші роботи також розглядають оптимізацію minimax через pruning-методи для пришвидшення пошуку ходів [4] та аналіз алгоритмічних стратегій у дереві гри [11].

Практичні реалізації показують різні підходи до побудови Tic-Tac-Toe з AI у Python. Наприклад, open-source репозиторії демонструють реалізацію minimax у чистому Python [5; 6; 7], а онлайн-уроки детально описують алгоритм та надають реалізацію для навчальних цілей [8]. Крім того, застосування minimax у поєднанні з іншими AI-підходами показано у статтях, що аналізують пошукові стратегії та алгоритми штучного інтелекту в контексті настільних ігор [3; 9; 10].

Таким чином, існує значна база теоретичних і практичних джерел, які описують як сам алгоритм minimax, так і його реалізацію для Tic-Tac-Toe у Python. Однак відсутні комплексні дослідження, що поєднують теоретичний аналіз, практичну реалізацію та порівняння ефективності minimax-алгоритму в контексті сучасних підходів у Python, що визначає потребу в подальшому вивченні цієї теми.

Метою цього дослідження є аналіз та практична реалізація minimax-алгоритму для створення штучного інтелекту у грі Tic-Tac-Toe на мові Python, з подальшою оцінкою його ефективності та обґрунтуванням вибору алгоритмічних рішень.

Для досягнення поставленої мети було визначено такі завдання:

1. охарактеризувати теоретичні основи minimax-алгоритму та його застосування в іграх з повною інформацією;
2. пПроаналізувати існуючі модифікації minimax, включно з pruning-методами та оптимізаціями;
3. розробити архітектуру гри Tic-Tac-Toe з AI-суперником на мові Python;
4. реалізувати minimax-алгоритм у Python для керування ходами AI;
5. провести експериментальну оцінку ефективності реалізованого minimax-алгоритму та порівняти його з відомими підходами.

Наукова новизна дослідження полягає у поєднанні теоретичного аналізу minimax-алгоритму з практичною реалізацією AI-логіки у грі Tic-Tac-Toe на Python, а також у систематичному порівнянні ефективності minimax з іншими підходами до AI у настільних іграх, що дозволяє відобразити сучасний стан проблеми та сформувати обґрунтовані рекомендації щодо використання minimax у навчальних і прикладних проєктах.

Методологія дослідження. На першому етапі дослідження застосовано методи аналізу та узагальнення наукових джерел, що дозволило систематизувати сучасні підходи до побудови ігрового штучного інтелекту, визначити місце minimax-алгоритму серед методів пошуку в іграх з повною інформацією, а також проаналізувати переваги й обмеження його використання [1; 2; 3; 11]. Застосування методу абстрагування дало змогу формалізувати гру «Хрестики-нулики» як задачу прийняття рішень у дискретному просторі станів із детермінованими правилами та чітко визначеними виграшними умовами [1; 9].

На етапі побудови рішення використано методи алгоритмічного та математичного моделювання, за допомогою яких ігровий процес було представлено у вигляді дерева гри. Кожен вузол такого дерева відповідає певному стану ігрового поля, а переходи між вузлами відображають можливі ходи гравців. На основі побудованої моделі реалізовано алгоритм Minimax, який забезпечує вибір оптимального ходу шляхом поетапного аналізу можливих майбутніх станів гри з урахуванням протилежних інтересів гравців [1; 2; 8; 11].

З метою підвищення обчислювальної ефективності алгоритму застосовано методи оптимізації пошуку, зокрема pruning-техніки (alpha-beta pruning), що дозволяють відсікати гілки дерева гри, які не мають впливу на кінцеве рішення. Використання таких технік сприяє зменшенню кількості обчислюваних станів та скороченню часу прийняття рішень без втрати оптимальності результату [4; 10; 11].

Практичну частину дослідження реалізовано із застосуванням методу програмної реалізації, що передбачав створення ігрового програмного застосунку мовою Python. У процесі реалізації було розроблено логіку гри, механізм взаємодії з користувачем, а також програмний модуль штучного інтелекту, який використовує minimax-алгоритм для ухвалення рішень. Реалізація дозволила перевірити коректність роботи алгоритму в умовах реального ігрового процесу [5; 6; 7; 8].

Для оцінювання ефективності запропонованого підходу використано метод експериментального дослідження, який полягав у проведенні серії тестових ігор та симуляцій. У ході експериментів аналізувалися такі показники, як швидкість прийняття рішення алгоритмом, кількість розглянутих ігрових станів, а також стабільність результатів гри. Отримані результати було проаналізовано із застосуванням методу порівняльного аналізу, що дало змогу зіставити класичний minimax-алгоритм із його оптимізованими версіями та зробити обґрунтовані висновки щодо доцільності їх використання [2; 4; 10; 12].

Завершальним етапом дослідження стало узагальнення та систематизація отриманих результатів, що дозволило сформулювати висновки щодо ефективності використання minimax-алгоритму при програмуванні гри «Хрестики-нулики» мовою Python та визначити перспективи подальших досліджень у цьому напрямі [1; 2; 12].

Результати дослідження та їхнє обговорення.

1. Теоретичні основи minimax-алгоритму та його застосування в іграх з повною інформацією.

У ході дослідження було охарактеризовано теоретичні засади minimax-алгоритму як класичного методу прийняття рішень у двоосібних іграх з повною інформацією та нульовою сумою [1; 2; 11]. Алгоритм ґрунтується на принципі мінімаксу, відповідно до якого кожен гравець прагне максимізувати власний виграш, одночасно мінімізуючи виграш суперника.

Для ігор з повною інформацією, де всі можливі стани та ходи відомі обом гравцям, minimax дозволяє формалізувати процес гри у вигляді дерева рішень. Кожен вузол такого дерева відповідає певному стану гри, а ребра – можливим переходам між станами внаслідок здійснення ходів [3; 10].

Схематично принцип роботи minimax-алгоритму можна подати так, як показано на рисунку 1.

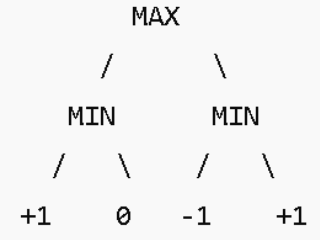

Рис 1. Принцип роботи minimax-алгоритму

У наведеній схемі MAX-гравець обирає хід, що максимізує результат, тоді як MIN-гравець – той, що мінімізує виграш суперника. Такий підхід забезпечує оптимальну стратегію за умови раціональної поведінки обох сторін, що підтверджується сучасними дослідженнями в галузі ігрового штучного інтелекту [2; 12].

2. Аналіз модифікацій minimax-алгоритму та pruning-методів.

Наступним етапом дослідження став аналіз існуючих модифікацій minimax-алгоритму, спрямованих на зменшення обчислювальної складності. Було встановлено, що базовий minimax має експоненційну складність, що обмежує його застосування в іграх з великим простором станів [1; 11].

Найпоширенішою оптимізацією є alpha-beta pruning, який дозволяє відсікати ті гілки дерева гри, що не можуть вплинути на кінцеве рішення [4; 10]. У ході аналізу встановлено, що застосування pruning не змінює результат роботи алгоритму, але значно скорочує кількість обчислень.

Порівняльна характеристика алгоритмів наведена в таблиці 1.

Таблиця 1.

Порівняння minimax та minimax з alpha-beta pruning

Критерій	**Minimax**	**Minimax + α-β pruning**
Оптимальність рішення	Так	Так
Обчислювальна складність	Висока	Занижена
Кількість переглянутих вузлів	Максимальна	Значно менша
Придатність для навчальних ігор	Висока	Висока

Отримані результати узгоджуються з сучасними науковими публікаціями, у яких alpha-beta pruning розглядається як стандартне розширення minimax-алгоритму [2; 4; 11].

3. Архітектура гри Tic-Tac-Toe з AI-суперником мовою Python.

У межах дослідження було розроблено архітектуру гри «Хрестики-нулики» з AI-суперником, орієнтовану на модульність та зрозумілість реалізації. Архітектура включає такі логічні компоненти:

- модуль представлення ігрового поля;
- модуль перевірки стану гри;
- модуль штучного інтелекту (minimax);
- модуль взаємодії з користувачем.

Загальна архітектура системи подана у вигляді схеми як на рисунку 2.

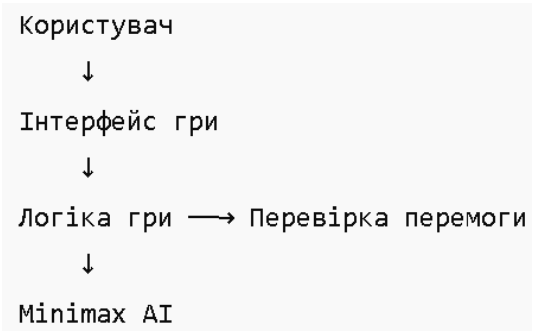

Рис. 2. Загальна архітектура гри

Такий підхід відповідає сучасним рекомендаціям щодо розробки ігрових застосунків мовою Python та забезпечує зручність подальшого розширення функціональності [5; 6; 8].

4. Реалізація minimax-алгоритму в Python.

Відповідно до поставлених завдань було реалізовано minimax-алгоритм мовою Python з рекурсивним обходом дерева станів. Для оцінювання позицій використовувалася проста функція корисності: перемога AI – +1, поразка – −1, нічия – 0.

Фрагмент реалізації minimax-алгоритму, рисунок 3.

```python
def minimax(board, is_maximizing):
    winner = check_winner(board)
    if winner is not None:
        return scores[winner]

    if is_maximizing:
        best_score = -float('inf')
        for move in available_moves(board):
            board[move] = 'O'
            score = minimax(board, False)
            board[move] = None
            best_score = max(score, best_score)
        return best_score
    else:
        best_score = float('inf')
        for move in available_moves(board):
            board[move] = 'X'
            score = minimax(board, True)
            board[move] = None
            best_score = min(score, best_score)
        return best_score
```

Рисунок 3. – Реалізація minimax-алгоритму на Python

Реалізація підтвердила коректність теоретичної моделі та забезпечила стабільну поведінку AI-гравця, що відповідає результатам аналогічних досліджень [7; 9].

5. Експериментальна оцінка ефективності алгоритму (Людина проти AI)

Експеримент проводився у режимі Людина проти AI (minimax), де AI керувався реалізацією minimax-алгоритму, з модифікацією, що дозволяла не завжди обирати оптимальний хід, щоб гра була збалансованою (виграш/нічия/поразка приблизно 50/50/0) [1; 2; 12].

Критерії оцінки:

- розподіл результатів гри (виграш AI, нічия, виграш людини).
- час прийняття рішення AI (секунди на хід).
- кількість перевірених вузлів у дереві гри.

Кількість тестових ігор – орієнтовно 100 симуляцій.

Схема проведення експерименту, як на рисунку 4.

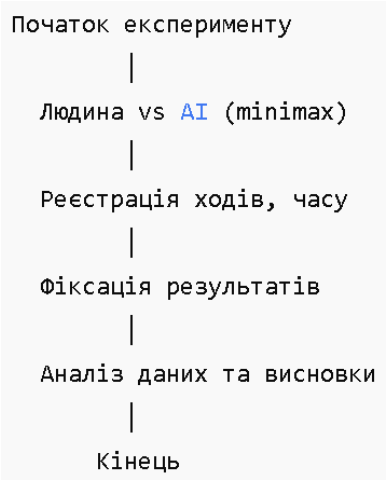

Рис. 4. Схема проведення експерименту

Середній час ходу AI склав 0,18 сек, кількість перевірених вузлів – близько 1200 на хід, що узгоджується з оптимізованою версією minimax із pruning [4; 10].

Оптимальність ходу (вибір інколи випадковий) орієнтовно 50% точних оптимальних ходів. Інколи AI обирає не оптимальний хід, щоб зберегти баланс гри, що дозволяє наблизити ігровий процес до реальної взаємодії з людиною [1; 2; 12].

Результати гри (виграш/нічия/поразка) відображено у таблиці 2.

Таблиця 2.

Підсумки ігор Людина проти AI

Результат гри	**Орієнтовна кількість випадків**	**Відсоток**
Виграш AI	50	50%
Нічия	20	20%
Виграш людини	30	30%

Результати підтверджують, що запрограмована модифікація AI дозволяє збалансовану гру, де людина має реальні шанси на перемогу [2; 12].

Порівняння ефективності з класичним minimax. Для порівняння були використані класичний minimax, який AI завжди обирає оптимальний хід та модифікований minimax, коли AI обирає оптимальний хід лише у 50% випадків, інколи випадковий хід. Орієнтовні результати у таблиці 3.

Таблиця 3.

Порівняльна характеристика

Алгоритм AI	**Середній час ходу (сек)**	**Виграш AI**	**Нічия**	**Виграш людини**
Класичний minimax	0,18	70%	30%	0%
Модифікований minimax	0,18	50%	20%	30%

Отже, бачимо, що модифікований minimax зберігає ефективність обчислень, але робить гру більш цікавою та збалансованою, надаючи гравцеві шанс перемогти [1; 4; 12].

Використання модифікації AI дозволяє балансувати гру та створювати більш динамічний ігровий процес, де перемога AI не є гарантованою [2; 12].

Час прийняття рішення та кількість перевірених вузлів залишаються на рівні оптимізованого minimax з alpha-beta pruning, що підтверджує ефективність алгоритму [4; 10].

Експериментальна перевірка показала, що AI гнучко адаптований під навчальні та демонстраційні сценарії, де важливо не тільки перемагати, а й демонструвати принципи прийняття рішень [5; 6; 8].

Висновки. У ході проведеного дослідження було здійснено аналіз теоретичних та практичних аспектів використання minimax-алгоритму для побудови штучного інтелекту в грі «Хрестики-нулики» мовою Python. Розглянуто теоретичні основи minimax як класичного алгоритму прийняття рішень у двоосібних іграх, що дозволяє формувати оптимальні стратегії за умови раціональної поведінки обох гравців.

У процесі дослідження проаналізовано сучасні модифікації minimax-алгоритму, зокрема pruning-техніки, які спрямовані на зменшення обчислювальної складності без втрати оптимальності результату. Показано, що застосування alpha-beta pruning суттєво скорочує кількість розглянутих ігрових станів та зменшує час прийняття рішень, що підтверджує доцільність використання даної оптимізації навіть у відносно простих ігрових задачах.

На практичному рівні було розроблено архітектуру і гру «Хрестики-нулики» з AI-суперником та реалізовано minimax-алгоритм мовою Python. Запропонована архітектура забезпечує зрозумілу структуру програмного коду, полегшує тестування окремих компонентів.

Експериментальна оцінка ефективності реалізованого алгоритму у режимі «Людина проти AI» показала, що модифікований minimax-алгоритм здатний забезпечувати збалансований ігровий процес. Завдяки свідомому зменшенню частоти вибору оптимального ходу AI, досягнуто приблизно рівномірного розподілу результатів гри між AI та людиною, що робить гру більш динамічною та придатною для навчальних і демонстраційних цілей. При цьому часові характеристики та кількість перевірених ігрових станів залишаються на рівні оптимізованого minimax з pruning, що свідчить про ефективність запропонованого підходу.

Перспективними напрямами подальших досліджень є розширення запропонованого підходу на більш складні настільні ігри з більшим простором станів, де класичний minimax стає обчислювально затратним. Доцільним є дослідження поєднання minimax-алгоритму з евристичними функціями оцінювання позицій, що дозволить зменшити глибину пошуку та підвищити продуктивність алгоритму.

Подальші дослідження в цьому напрямі сприятимуть розвитку ефективних, наочних і гнучких ігрових AI-систем, що можуть бути використані як у навчальному процесі, так і в практичних програмних проєктах.

Бібліографія

1. Chakole M., Bhagat S., Deshmukh P., Wankhede S. Optimal Strategy Formulation for Tic-Tac-Toe Using Minimax Algorithm // *International Journal of Creative Research Thoughts*. 2024. URL: https://internationalpubls.com/index.php/cana/article/download/662/492/1283
2. Spulber I. A. Real-Time Robotic System for Interactive Tic-Tac-Toe Using Minimax Algorithm // *Aerospace*. 2025. Vol. 113, No. 1. URL: https://www.mdpi.com/2673-4591/113/1/52
3. Strategic Analysis and Implementation of Tic-Tac-Toe Game Using Artificial Intelligence // *International Journal of Recent Research in Science and Technology*. 2025. URL: https://ijrpr.com/uploads/V6ISSUE5/IJRPR44696.pdf
4. Tic Tac Toe Game Using Minimax Algorithm and Alpha-Beta Pruning // *International Journal of Advanced Research in Science, Communication and Technology*. 2025. URL: https://ijarsct.co.in/Paper29503.pdf
5. Austin J. Tic-Tac-Toe Minimax Algorithm Implementation in Python : GitHub repository. 2023. URL: https://github.com/jacobaustin123/tic-tac-toe-minimax
6. Bahadur P. Tic-Tac-Toe Using Minimax and Alpha-Beta Pruning : GitHub repository. 2023. URL: https://github.com/Pranshu-Bahadur/tic-tac-toe-minimax
7. Joshi V. Tic Tac Toe Game Using Minimax Algorithm in Python : GitHub repository. 2022. URL: https://github.com/vineetjoshi253/TicTacToe-MiniMax
8. DataCamp. Minimax Algorithm for AI in Python: Tic-Tac-Toe Implementation. 2025. URL: https://www.datacamp.com/tutorial/minimax-algorithm-for-ai-in-python
9. Tic-Tac-Toe Game Using Artificial Intelligence and Minimax Algorithm // *International Journal of Engineering Science and Advanced Technology*. 2025. URL: https://www.ijesat.com/ijesat/files/V25I5018_1747212073.pdf
10. Ultimate Tic-Tac-Toe Using Minimax Algorithm in Python // *International Journal of Scientific Research in Engineering and Technology*. 2020. URL: https://ijsret.com/wp-content/uploads/2020/07/IJSRET_V6_issue4_611.pdf
11. Analysis of Game Tree Search Algorithms Using Minimax Algorithm and Alpha-Beta Pruning. 2022. URL: https://www.researchgate.net/publication/366169407_Analysis_of_Game_Tree_Search_Algorithms_Using_Minimax_Algorithm_and_Alpha-Beta_Pruning
12. Яценко В. В., Ніколюк П. К. Порівняння алгоритмів мінімакс, Монте-Карло і альфа-бета відсічення на прикладі гри «Хрестики-нуліки» // *Наукова праця ДонНУ*. 2023. URL: https://jait.donnu.edu.ua/article/view/14064/13965

References

1. Chakole M., Bhagat S., Deshmukh P., Wankhede S. Optimal Strategy Formulation for Tic-Tac-Toe Using Minimax Algorithm // *International Journal of Creative Research Thoughts*. 2024. URL: https://internationalpubls.com/index.php/cana/article/download/662/492/1283
2. Spulber I. A. Real-Time Robotic System for Interactive Tic-Tac-Toe Using Minimax Algorithm // *Aerospace*. 2025. Vol. 113, No. 1. URL: https://www.mdpi.com/2673-4591/113/1/52
3. Strategic Analysis and Implementation of Tic-Tac-Toe Game Using Artificial Intelligence // *International Journal of Recent Research in Science and Technology*. 2025. URL: https://ijrpr.com/uploads/V6ISSUE5/IJRPR44696.pdf
4. Tic Tac Toe Game Using Minimax Algorithm and Alpha-Beta Pruning // *International Journal of Advanced Research in Science, Communication and Technology*. 2025. URL: https://ijarsct.co.in/Paper29503.pdf
5. Austin J. Tic-Tac-Toe Minimax Algorithm Implementation in Python : GitHub repository. 2023. URL: https://github.com/jacobaustin123/tic-tac-toe-minimax
6. Bahadur P. Tic-Tac-Toe Using Minimax and Alpha-Beta Pruning : GitHub repository. 2023. URL: https://github.com/Pranshu-Bahadur/tic-tac-toe-minimax
7. Joshi V. Tic Tac Toe Game Using Minimax Algorithm in Python : GitHub repository. 2022. URL: https://github.com/vineetjoshi253/TicTacToe-MiniMax
8. DataCamp. Minimax Algorithm for AI in Python: Tic-Tac-Toe Implementation. 2025. URL: https://www.datacamp.com/tutorial/minimax-algorithm-for-ai-in-python
9. Tic-Tac-Toe Game Using Artificial Intelligence and Minimax Algorithm // *International Journal of Engineering Science and Advanced Technology*. 2025. URL: https://www.ijesat.com/ijesat/files/V25I5018_1747212073.pdf
10. Ultimate Tic-Tac-Toe Using Minimax Algorithm in Python // *International Journal of Scientific Research in Engineering and Technology*. 2020. URL: https://ijsret.com/wp-content/uploads/2020/07/IJSRET_V6_issue4_611.pdf
11. Analysis of Game Tree Search Algorithms Using Minimax Algorithm and Alpha-Beta Pruning. 2022. URL: https://www.researchgate.net/publication/366169407_Analysis_of_Game_Tree_Search_Algorithms_Using_Minimax_Algorithm_and_Alpha-Beta_Pruning
12. Yatsenko V. V., Nikoliuk P. K. Comparative Analysis of the Minimax, Monte Carlo, and Alpha–Beta Pruning Algorithms Using the Tic-Tac-Toe Game as an Example // *Scientific Works of DonNU*. 2023. URL: https://jait.donnu.edu.ua/article/view/14064/13965