МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЧЕРНІЙЧУК ГАЛИНА ПЕТРІВНА

Допускається до захисту:
в. о. завідувача кафедри,
Прикладної математики
_____ Трофименко О.Д.
«____» _____20___ р.

АЛГОРИТМІЧНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ДЕЯКИХ КОМБІНАТОРНИХ ІГОР

Спеціальність 113 Прикладна математика

Кваліфікаційна (бакалаврська) робота

Керівник:
Вєтров О. С., старший викладач
кафедри Прикладної математики
_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2021

АНОТАЦІЯ

Чернійчук Г. П. Алгоритмічна та програмна реалізація деяких комбінаторних ігор. Спеціальність 113 «Прикладна математика», спеціалізація «Прикладна математика». Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній (бакалаврській) роботі досліджено комбінаторні ігри та алгоритми комбінаторних ігор. Показано роботу досліджуваних алгоритмів на прикладі комбінаторних ігор нім, гекс, так-тікс та клоббер. Розроблена програма для ігор нім та так-тікс.

Ключові слова: комбінаторна гра, дерево рішень, жадібний алгоритм, симетричний алгоритм, нім, гекс, так-тікс, клоббер.

62 с., 1 табл., 25 рис., 24 джерел.

ABSTRACT

Cherniychuk H. Algorithmic and software implementation of some combinatorial games. Specialty 113 “Applied mathematics”, specialization “Applied mathematics”. Vasyl` Stus Donetsk National University, Vinnytsia, 2021.

Combinatorial games and algorithms of combinatorial games are studied in the qualification (bachelor's) work. The work of the studied algorithms is shown on the example of combinatorial games nim, hex, tak-tiks and clobber. Developed a program for playing nim and tak-ticks.

Keywords:combinatorial game, decision tree, greedy algorithm, symmetric algorithm, nim, hex, tak-ticks, clobber.

ЗМІСТ

ВСТУП ……4
РОЗДІЛ 1. КОМБІНАТОРНІ ІГРИ ТА ЇХ АЛГОРИТМИ ……6
1.1 Комбінаторні ігри ……..6
1.2 Класифікація комбінаторних ігор ……8
1.3 Алгоритми комбінаторних ігор ……9
1.3.1 Дерево рішень ……..9
1.3.2 Жадібний алгоритм ……...10
1.3.3 Симетричний алгоритм ……11
1.3.4 Асимптотична складність алгоритму ……..11
1.4 Формальне визначення гри ……12
1.4.1 Визначення додавання ……..15
1.4.2 Визначення мінуса ……15
1.4.3 Визначення еквівалентності гри ……..15
1.4.4 Визначення нерівності ……..17
Висновки до розділу 1 ……18
РОЗДІЛ 2. АНАЛІЗ КОМБІНАТОРНИХ ІГОР ……19
2.1 Гра «Нім» ……..19
2.2 Гра «Так-Тікс» ……..24
2.3 Гра «Гекс» ……29
2.4 Гра «Клоббер» ……..36
Висновки до розділу 2 ……37
РОЗДІЛ 3. СТВОРЕННЯ ПРОГРАМИ ДЛЯ КОМБІНАТОРНИХ ІГОР «НІМ» ТА «ТАК-ТІКС» ……//……38
3.1 Аналіз програми та коду ……...38
3.2 Приклад програмної реалізації ……52
Висновки до розділу 3 ……....60
ВИСНОВКИ ……..61
СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ ……..62

Вступ

Теорія ігор – це розділ прикладної математики, який досліджує моделі прийняття рішень в умовах неспівпадіння інтересів сторін (гравців) [1].

Комбінаторна теорія ігор – це математична теорія, що вивчає ігри двох осіб, де у кожен момент часу є позиція, яку гравці почергово змінюють, для досягнення виграшу [4].

Застосування комбінаторної теорії ігор до певної позиції полягає у визначенні оптимальної послідовності ходів для обох гравців аж до кінця гри, і таким чином визначення оптимального ходу в кожній позиції [3].

Теорія ігор застосовується в різних галузях людської діяльності: в біології, політології, кібернетиці, економіці, тощо [17]. Теорію ігор та конфліктні ситуації найчастіше застосовують в економіці. В кожного гравця є стратегії, які він може застосовувати. Перетин стратегій двох гравців створюють ситуацію, де кожен з гравців отримує результат: виграш або програш [3]

Людству комбінаторні ігри відомі досить давно. До комбінаторних ігор належать такі відомі ігри як шахи (VI ст. н.е., Індія), шашки, Го (~4000 років тому, Китай) та багато інших. Дані ігри є досить популярними в наш час, оскільки правила таких ігор доволі прості, але самі ігри потребують знань, логічного мислення та правильного розрахунку ходів.

Початок вивчення комбінаторних ігор було покладено статтею Ч. Баутона «Нім, гра з повною математичною теорією» в 1902 році [15]. Розвиток математики дав поштовх для розвитку математичної теорії комбінаторних ігор.

Комбінаторні ігри також є досить цікавими з точки зору штучного інтелекту. Тривалий період вважалося, що штучний інтелект досить сильно поступається людському розумі. За допомогою штучного інтелекту вже є розроблені програми, які використовують математичну теорія комбінаторних ігор, виграють у найбільш відомих шахістів Г. Каспарова та В. Крамнікова. Також були розроблені алгоритми, які змогли перемогти чемпіона Го – Лі Седоля.

Метою дипломної роботи є аналіз алгоритмів та програмна реалізація комбінаторних ігор.

Предметом дослідження є комбінаторні ігри.

Об’єктом дослідження є алгоритми комбінаторних ігор.

Актуальність: дослідження комбінаторних ігор є досить актуальним. Комбінаторні ігри відомі людству досить давно. В наш час вони є досить популярними для гравців по всьому світу. Ігри такого типу мають відносно прості правила, завдяки яким є досить популярними. Комбінаторні ігри є досить актуальними з точки зору розвитку штучного інтелекту та математики. Маючи складну структуру, величезну кількість можливих ходів та станів, неможливо побудувати напряму систему прийняття рішень, яка б брала до уваги всі можливі комбінації та можливі дії гравців. Розвиток науки та інформаційних технологій, в деякій мірі спрощує дослідження комбінаторних ігор. Таким чином, дослідження алгоритмів та пошук стратегій в комбінаторних іграх актуальними, як дослідницькі задачі, для розвитку технологій штучного інтелекту, також має прикладне значення для розробників комбінаторних ігор.

Для досягнення поставленої мети потрібно виконати наступні **завдання:**

1. Проаналізувати алгоритми для комбінаторних ігор.
2. Розглянути застосування обраних алгоритмів для деяких комбінаторних ігор.
3. Зробити програмну реалізацію для комбінаторних ігор.

Бакалаврська робота складається із вступу, трьох розділів, загального висновку, списку літератури та додатку. Загальний розмір роботи – 61 сторінок.

РОЗДІЛ 1
КОМБІНАТОРНІ ІГРИ ТА АЛГОРИТМИ ДЛЯ НИХ

1.1. Комбінаторні ігри.

Комбінаторні ігри – це ігри, в яких приймають участь два гравці без випадкових ходів та з повною інформацією [18]. Мається на увазі, що позиції та можливі ходи відомі обом гравцям. Дана гра визначається множиною позицій і гравцем, чия черга робити хід. В множину позицій включається, також і початкова позиція. Гравці роблять ходи по черзі, таким чином змінюючи позиції від однієї до іншої, поки не буде досягнута кінцева позиція. Кінцевою позицією називається позиція, в якій немає можливих ходів. При досягненні кінцевої позиції один з гравців оголошується переможцем, а інший тим, хто програв.

Комбінаторна теорія ігор, це математична теорія, що вивчає ігри двох осіб, в яких у кожен момент часу є позиція (стан), яку гравці змінюють таким чином, щоб досягти перемоги. Дана теорія не вивчає ігри, пов’язані з випадковістю. Комбінаторна теорія ігор вивчає такі ігри, в яких позиція і всі можливі ходи відомі обом гравцям.

Комбінаторна гра, це гра яка задовольняє такі умови [18]:

1. Існує два гравці, часто їх називають лівим та правим гравцем. Не має місця створення ігрових коаліцій.
2. Гра має обмежену кількість позицій і досить часто конкретне вихідне положення.
3. Правила гри є однаковими для обох гравців. Кожна позиція, яка переходить в іншу позицію, є дозволеним ходом.
4. Гравці ходять по черзі.

5. При досягненні позиції, з якої немає можливих ходів, гра буде завершена. Для нормальної гри виграє той, хто зробив останній хід, для мізерної гри той, хто зробив останній хід, програє.
6. Після кінцевого числа ходів, незалежно від ходу самої гри, гра закінчується.
7. Гравці знають, що відбувається на ігровому полі - наявна повна інформація. Нема місця для блефу.
8. Немає випадкових ходів, роздачі карт та кидання гральних кубиків.

Ludo, Snakes-and-Ladders, Нарди – мають повну інформацію, але містять випадкові ходи, так як в цих іграх використовують гральні кубики для ходів.

Лінкори, камінь-ножиці-папір – не мають випадкових ходів, але також немає повної інформації про хід суперника. Оскільки суперники не знають повної інформації про положення фігур або пальців. Окрім того, в таких іграх як камінь-ножиці-папір, гравці ходять не по черзі а одночасно.

Хрестики-нолики не відповідають умові 5, оскільки гравець який не має ходів не завжди програє, можливі нічиї.

Шахи також не відповідають умові 5 і містять позиції, які пов’язані з нічиєю і позиції, які розігруються нескінченним числом.

Монополія також не відповідає декільком умовам. Як і в грі Ludo може бути більше двох гравців. У гравців нема повної інформації про розміщення карт і гра, теоретично, може продовжуватися досить довго.

В покері інтерес виникає за рахунок незавершеності інформації, можливості ходів в коаліцій.

Міст – гра для двох людей, але у гравців немає повної інформації про власні карти.

Ігри які повністю підпадають під визначення комбінаторних – нім, клоббер, гекс, так-тікс. [18]

1.2. Класифікація комбінаторних ігор

В теорії комбінаторних ігор є чотири напрямки досліджень: неупереджені ігри з нормальними умовами завершення, упереджені ігри з нормальними умовами завершення, неупереджені ігри з умовами завершення мізер та упереджені ігри з умовами завершення мізер [9].

Неупереджена гра – це рівноправна комбінаторна гра з наступними властивостями [9]:

- дія або хід, який доступний гравцеві залежить лише від стану гри, але не залежить від того, хто зараз робить хід;
- якщо гравець не має можливості зробити хід то він програє;
- гра завжди має закінчуватися.

Умову, при якій закінчується гра можна змінювати, для отримання так званої гри у мізер. Мізер-гра у порівнянні з простою грою має лише одну відмінність: гравець, який зробив останній хід програє.

До такого типу ігор можна віднести гру Нім, Так-Тікс, в яких присутні два гравці, що беруть спільні предмети по черзі. Оскільки предмети на ігровому полі спільні для обох гравців – то стан гри не залежить від кроку, того гравця, який ходить, гра залежить від кількості предметів на ігровому полі.

Упереджена гра або ж партизанська гра – нерівноправна комбінаторна гра, в якій кожний гравець в заданій позиції має різні набори можливих ходів.

Шахи, шашки є упередженими, оскільки гравці мають власні фігури і ними роблять ходи [8].

Комбінаторні ігри класифікують також по матеріальним ознакам [8]:

- ігри з фішками: гравці роблять ходи за допомогою фішок, вони можуть бути спільними для обох гравців, так і різного виду для гравців.

- ігри на дошці: для такого типу ігор використовують дошку, як ігровий простір. Дошка ділиться на певні позиції, які становлять основу гри.

- ігри з монетами: для такого типу ігор монети використовують в якості фішок, виконуються певні дії над монетами (ходи) для досягнення перемоги.

- ігри з картами: гравці виконують дії за допомогою пластикових, дерев'яних, паперових карт, або карти є основою для гри. Карти не обов'язково мають бути гральними.

- ігри з папером та ножицями та/або з олівцями: за допомогою ножиць та паперу можна створити власні ігри, або зробити фішки для одного з вищевказаних видів ігор.

1.3. Алгоритми комбінаторних ігор.

1.3.1 Дерево рішень

Ухвалення рішення – процес раціонального або ірраціонального вибору альтернатив, що має на меті досягнення результату. Одним з алгоритмів аналізу є алгоритм дерева рішень.

Дерево рішень – схематичне представлення проблеми ухвалення рішень, яке використовується для вибору найкращого напряму дій з наявних варіантів. Даний алгоритм використовує модель, що розгалужується за певними умовами. Така модель є графічним представленням зв'язків основних та подальших варіантів, за допомогою даного алгоритму створюється дерево гри. Дерево рішень має переваги порівняно з іншими алгоритмами. За допомогою даного алгоритму можна візуально оцінити результати роботи алгоритму і обирати найкращий з варіантів [5].

Визначення 1.1 Враховуючи гру G з початковою позицією S, дерево гри, пов'язане з (G,S), є напіввпорядкованим кореневим деревом, визначеним наступним чином [13]:

- Корінь вершини відповідає початковій позиції.

- Усі можливі ігрові позиції, до яких можна дістатися ліворуч (відповідно праворуч) за один хід від початкової позиції S, встановлюються як ліві (відповідно праві) діти кореня.
- Попереднє правило рекурсивно застосовується для кожної дитини кореня.

Перевагами дерева рішень є: можливість відразу бачити результати роботи алгоритму.

Недоліками дерева рішень є те, що можуть з’явитися занадто складні конструкції, які не достатньо точно будуть представляти дані.

1.3.2 Жадібний алгоритм

Жадібний алгоритм [16] є простим і прямолінійним алгоритмом, який приймає краще рішення, виходячи з наявних на даному етапі даних. Гравець, який застосовує даний алгоритм, не турбується про можливі наслідки, а сподівається отримати оптимальне рішення. Іншими словами, гравець намагається забрати як можна більше фішок з поля за один хід. Жадібний алгоритм є простим в реалізації і часто дуже ефективний за часом виконання.

Якщо грати за допомогою такого алгоритму, то гра в якійсь мірі не є досить цікаво. Звичайно в будь-якій грі є елемент жадібності.

Жадібний алгоритм базується на п’яти принципах:

- Набір можливих варіантів, з яких робиться вибір;
- Функція вибору, за допомогою якої знаходиться найкращий варіант;
- Функція придатності, яка визначає придатність отриманого вибору;
- Функція цілі, оцінює цінність рішення, не виражається явно;
- Функція розв’язку, яка вказує на те, що знайдено кінцеве рішення.

Жадібний алгоритм краще підходить для простих ігор. Для більш складних ігор, такий алгоритм є не доцільним, оскільки можуть привести не до оптимального розв’язку, а навпаки видати гірший з можливих розв’язків.

1.3.3 Симетричний алгоритм

Симетричний алгоритм [16] в більшій мірі є інтуїтивним алгоритмом. Кожного разу коли суперник робить хід в одній частині дошки, то потрібно імітувати такий хід в іншій частині дошки.

Для успішної гри, не потрібно залишати супернику хід, який дозволить усунути імітуючий хід.

Одним із найбільш цікавим прикладом симетричної гри, є парі невідомої шахістки Дж. Пеунпушер з Г. Каспаровим та А. Карповим.

Вона пропонує зіграти дві партії, граючи одночасно білими фігурами проти Г. Каспарова і чорними проти А. Карпова. Дж. Пеунпушер просто чекала поки А. Карпов зробить хід, і щоб він не зробив, вона робить такий самий хід, проти Г. Каспарова. Коли Каспаров відповідає, вона програє його відповідь проти Карпова. Таким чином, можна навіть не знати правила, а виграти, хоча б в одного суперника.

1.3.4 Оцінка складності алгоритму або асимптотична складність

Складність алгоритму зазвичай оцінюють за часом виконання або по використаній пам’яті. В обох випадках складність залежить від розмірів вхідних даних. Чим більше даних тим більше часу потрібно для виконання алгоритму. Точний час не є цікавим, більше уваги приділяють саме асимптотичній складності [14].

Асимптотична складність – складність при прагненні розміру вхідних даних до нескінченності. Позначається O(f(n)), формально це означає, що час роботи алгоритму (або обсяги зайнятої пам’яті) росте в залежності від обсягу вхідних даних не швидше, ніж деяка константа, помножена на f(n).

Приклади асимптотичних складностей [7]:

- O(1) – сталий час роботи не залежно від розміру задачі;
- O(log log n) – дуже повільне зростання необхідного часу;
- O(log n) – логарифмічне зростання – подвоєння розміру задачі збільшує час роботи на сталу величину;
- O(n) – лінійне зростання – подвоєння розміру задачі подвоїть і необхідний час;
- O(n log n) – лінеаритмічне зростання – подвоєння розміру задачі збільшить необхідний час трохи більше ніж вдвічі;
- O(n^2) – квадратичне зростання – подвоєння розміру задачі вчетверо збільшує необхідний час;
- O(n^3) - кубічне зростання - подвоєння розміру задачі збільшує необхідний час у вісім разів;
- O(c^n) – експоненціальне зростання – збільшення розміру задачі на 1 призводить до c-кратного збільшення необхідного часу; подвоєння розміру задачі підносить необхідний час у квадрат;

1.4. Формальне визначення гри

Грою називається множина G(«позицій») разом з виділеним елементом («початкова позиція») і фіксованими для кожної позиції двома підмножинами («допустимі ходи лівого та правого (перший та другий гравець) з позиції»). Також важливо, щоб кількість кінцевих ходів була скінченою. [2]

Визначення 1.2 Нехай є позиція та гравець, що починає гру. Тоді в одного з двох гравців завжди є виграшна стратегія, тобто він може виграти не залежно від дій суперника. [2]

Визначення 1.3 Для будь якої позиції реалізується одна з наступних можливостей: [2]

- Лівий має виграшну стратегію не залежно від того, хто починає гру;
- Правий має виграшну стратегію не залежно від того, хто починає гру;
- Гравець, який почав гру, має виграшну стратегію;
- Другий гравець має виграшну стратегію.

Гра – впорядкована пара наборів ігор [6]:

$$G = \{\{G^{L_1}, G^{L_2}, \ldots\} \mid \{G^{R_1}, G^{R_2}, \ldots\}\} \quad (1.1)$$

де G^L та G^R набори ігор для правого та лівого гравця (першого та другого гравця)

Основні поняття:

- Гра G це набори $\{G^L \mid G^R\}$ (1.2)
- Сума двох ігор $G + H \stackrel{\text{def}}{=} \{G^L + H, G + \square^L \mid G^R + H, G + \square^R\}$ (1.3)
- Мінус гри $-G \stackrel{\text{def}}{=} \{-G^L \mid -G^R\}$ (1.4)
- Дві однакові гри G = H, якщо (□X) G + X має такий самий результат, що і H + X (1.5)
- Одна гра надає перевагу іншій грі G ≥ H, (□X) лівий виграє G + X коли завгодно лівий виграє H + X (1.6)

Саме визначення поняття гри є досить стислим та рекурсивним. Також визначення визначає гру $G = G^L = G^R = \emptyset$, як нульову. Однією з переваг рекурсивного визначення поняття гри без явних базових випадків є те, що багато індуктивних доказів на основі даного визначення також не потребуватимуть явних базових випадків.

Визначення 1.4 День народження гри $G = \{G^L \mid G^R\}$ (1.2) визначається рекурсивно як 1 плюс максимальний день народження будь-якої гри $G^L \cup G^R$. У базовому випадку, якщо $G^L = G^R = \emptyset$, то день народження дорівнює 0 [16].

День народження гри – це висота ігрового дерева. Гра $0 \stackrel{\text{def}}{=} \{ \mid \}$ – єдина гра, яка народилася на 0 день. Важливо пам'ятати, що $\{ \mid \} = \{\emptyset \mid \emptyset\}$ це гра де G^L та

G^R порожні. Це не те саме, що {0|0}, де будь-яка гра має можливість перейти до 0.

За допомогою рекурсивного визначення гри, можна перерахувати чотири гри, народженні до першого дня:

$$0 \stackrel{\text{def}}{=} \{ \mid \}; \tag{1.7}$$

$$1 \stackrel{\text{def}}{=} \{0 \mid \}; \tag{1.8}$$

$$-1 \stackrel{\text{def}}{=} \{ \mid 0\}; \tag{1.9}$$

$$* \stackrel{\text{def}}{=} \{0 \mid 0\}; \tag{1.10}$$

Ігри народженні до другого дня, повинні мати усі свої ліві та праві опції, народженні до першого дня. З 16 можливих підмножин чотирьох ігор, народжених до першого дня, задається 256 ігор, народжених до другого дня. Багато з цих ігор є рівними з огляду на поняття (1.5), насправді існує лише 22 різні гри народження до другого дня.

Лема 1.1 Число ігор, народжених до дня n, є кінцевою та позначається g(n) [16].

Доведення: Оскільки гра, народжена до дня n, задається парою наборів ігор, народжених до дня n-1, кількість ігор, народжених до дня n, становить не більше 2g(n-1) * 2g(n-1). За індукцією g(n-1) є скінченним числом, тому можна зробити висновок, що g(n) також є скінченним числом.

Теорема 1.1 Існує 22 гри народженні до другого дня.[16]

Висновок: Якщо G має кінцевий день народження, тоді G повинен мати кінцеве число можливих варіантів.

Доведення: Гра G з кінцевим днем народження n має мати варіанти, народженні в день n-1 та вибір є лише g(n-1) варіантів. Можна зробити висновок, що G^L та G^R є скінченими.

1.4.1 Визначення додавання

$$G + H \stackrel{\text{def}}{=} \{G^L + H, G + \square^L \mid G^R + H, G + \square^R\} \quad (1.3)$$

G та H – ігри, G^L, G^R, H^R та H^R – набори ігор. Визначається додавання однієї гри G до набору ігор S, як набір ігор, отриманих додаванням G до кожного з елементів S [15]:

$$G + S = \{G + X\}_{X \in S} \quad (1.11)$$

Теорема 1.2 $G + 0 = G$ (1.12)

Доведення: $G + 0 = \{ G^L \mid G^R \} + \{ \mid \} = \{ G^L + 0, G + \emptyset \mid G^R + 0, G + \emptyset \}$

Але для будь-якого набору S, $S + \emptyset = \emptyset$, та за індукцією, $G^L + 0 = G^L$ та $G^R + 0 = G^R$, спрощується до виразу $\{G^L \mid G^R\} = G$.

Теорема 1.3 Додавання також буває комутативним та асоціативним.

1) $G + H = H + G$ (1.13)
2) $(G + H) + J = G + (H + J)$ (1.14)

1.4.2 Визначення мінуса гри

$$-G \stackrel{\text{def}}{=} \{-G^L \mid -G^R\} \quad (1.4)$$

Визначення мінуса гри відповідає зміні ролей двох гравців під час гри. Міняються місцями ліві та праві набори та рекурсивно змінюються їхні ролі у всіх параметрах [15].

1.4.3 Визначення еквівалентності гри

G = H (1.5), якщо (□X) G + X має такий самий результат, що і H + X. В цілому G = H, якщо G діє як H у будь-якій сумі ігор [15].

Теорема 1.4 G = 0 тоді і тільки тоді, коли G знаходиться в P-позиції (G – виграш для другого гравця).

Доведення:

⇒ (пряме доведення) Якщо G = 0, то G + X має такий самий результат, як 0 + X = X для всіх X. В тому числі, коли X = 0, це означає, що G має такий самий результат, як 0, що є P-позицією.

⇐ (доведення від супротивного) Нехай існує P–позиція. Фіксується будь-яке положення X; потрібно показати, що G + X має такий самий результат, що і X.

Якщо ліві можуть виграти переміщення другим на X, ліві також можуть виграти переміщення другим на G + X використовуючи майже ту саму стратегію. При відтворенні G + X, коли відтворюється на першому компоненті, ліва відповідає на цей компонент, роблячи вигляд, що G грає поодинці. Якщо права грає на другому компоненті, ліва відповідає локально, вдаючи, що грає X поодинці. Ліва може отримати останній хід на G, а на X, отримує останній хід на G + X. якщо ліві можуть виграти рухаючись першими на X, то можуть виграти і на G + X рухаючись першими, та зробивши такий самий хід на другий компонент G + X. Використаємо відповідну симетрію і для правого, якщо правий виграє X то він виграє і G + X.

Отже, X та G+ X мають однаковий результат.

Висновок: G – G = 0 [15].

Доведення: другий гравець виграє на G - G, граючи в стратегію Tweedledum-Tweedledee. Якщо перший гравець переміщується з G – G на G - H, другий гравець може відповідати варіанту в іншій компоненті, переходячи на H – H та може перемогти за допомогою індукції.

Теорема 1.5 Виправити ігри G, H та J. Тоді G = H тоді і лише тоді, коли G + J = H + J [15].

Доведення:

⇒ (пряме доведення) Нехай G = H. Потрібно показати, що для всіх X, (G + J) + X має такий самий результат, що і (H + J) + X. Якщо обрати X` = (J + X) і застосувати визначення G = H: G + X` = H + X`; тобто G + (J + X) = H + (J + X) та за допомогою асоціативності (теорема 1.3) дає (G + J) + X = (H + J) + X

⇐ (доведення від супротивного) Припускається, що G + J = H + J. Використавши пряме доведення даної теореми, можна зробити висновок, що (G + J) + (-J) = (H + J) + (-J). За допомогою асоціативності та J – J = 0, маємо наступні рівності:

$$G = G + 0 \quad (1.15)$$
$$= G + (J - J) \quad (1.16)$$
$$= (G + J) – J \quad (1.17)$$
$$= (H + J) – J \quad (1.18)$$
$$= H + (J - J) \quad (1.19)$$
$$= H + 0 \quad (1.20)$$
$$= H \quad (1.5)$$

Висновок: G = H (1.5) тоді і тільки тоді, коли G – H = 0; тобто G – H є P-позицією.

Доведення: Щоб довести дану рівність потрібно додати до лівої та правої частини –H та використати асоціативність.

Даний наслідок є досить важливим, оскільки таким чином можна чітко визначити, чи G = H. На практиці це найпростіший спосіб перевірити, чи є рівними дві гри.

1.4.4 Визначення нерівності

G ≥ H, (□X) лівий виграє G + X, коли лівий виграє H + X. (1.6)

$G \leq H$, ($\square X$) правий виграє $G + X$, коли правий виграє $H + X$ [15]. (1.6.1)

Теорема 1.6 Наведено наступні еквівалентності:

1) $G \geq 0$

2) Ліві перемагають рухаючись другими на G

3) Для всіх ігор X, якщо ліва перемагає рухаючись другою на X, тоді ліва перемагає рухаючись лівою на $G + X$

4) Для всіх ігор X, якщо ліва перемагає рухаючись першою на X, то ліва перемагає рухаючись першою на $G + X$

Доведення:

$1 \Leftrightarrow 3$ та 4 (еквівалентності є рівноцінними): Клас результату гри визначається тим, чи перемагають ліві рухаючись першими чи другими

$2 \Rightarrow 3$ (з еквівалентності 2 випливає еквівалентність 3): Припустимо, ліві можуть виграти рухаючись другими на G та на X. Щоб виграти на $G + X$, граючи другим, використовується принцип зв'язування однієї руки.

$2 \Rightarrow 4$ (з еквівалентності 2 випливає еквівалентність 4): Доводиться аналогічно попередньому пункту

$3 \Rightarrow 2$ (з еквівалентності 3 випливає еквівалентність 2):Вибрати $X = 0$

$4 \Rightarrow 2$ (з еквівалентності 4 випливає еквівалентність 2):Розглядається еквівалент, протилежний четвертій умові: «якщо лівий не виграє рухаючись першим на $G + X$, а потім лівий не виграє рухаючись першим на X». $X = -G$. Оскільки $G - G = 0$, лівий не виграє, рухаючись першим на $G - G$. Тому лівий не виграє рухаючись першим на $-G$. Також діє еквівалентність по відношенню до правого: правий не виграє, рухаючись першим на G, тобто ліві перемагають, рухаючись другими на G.

Теорема 1.7 $G \geq H$ тоді і лише тоді, коли , $G + J \geq H + J$, для всіх ігор G, H та J.

Доведення: Доведення є аналогічним, до доведення теореми 1.5

Теорема 1.8 G ≥ H (1.6) тоді і тільки тоді, коли ліві перемагають, рухаючись другими на G - H

Доведення: G ≥ H тоді і тільки тоді, коли G - H ≥ H – H = 0 (тобто G - H ≥ 0)

Висновки до розділу 1

В цьому розділі було розглянуте визначення комбінаторних ігор та математичної теорії комбінаторних ігор, також була розглянута їх класифікація та алгоритми за допомогою яких можна вирішувати комбінаторні ігри.

РОЗДІЛ 2
АНАЛІЗ КОМБІНАТОРНИХ ІГОР

2.1. Гра «Нім»

Нім – одна із найстаріших математичних ігор [6]. Щоб грати у дану гру, потрібно два гравці, ігрове поле та набір фішок. Фішками може бути все що завгодно: камінчики, монетки, загалом все що є під рукою. В найбільш відомому варіанті гри 12 фішок розкладають в три ряди (рис. 2.1)

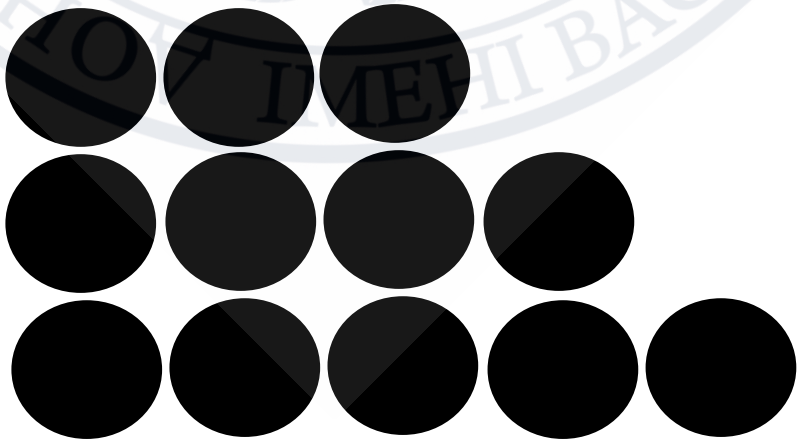


Рисунок 2.1 – Позиції фішок для гри по схемі 3-4-5

Гравці по черзі забирають одну чи декілька фішок з будь якого ряду. За один хід можна взяти фішки тільки з одного ряду, брати по кілька фішок з кількох рядів заборонено. Виграє той, хто забере останню фішку або фішки.

Повний аналіз гри нім з узагальненням на будь яку кількість рядів та з будь якою кількістю фішок в кожному ряді вперше опублікував професор математики Гарвардського університету Чарльз Л. Бутон в 1901 році [15]. Відкрита ним оптимальна стратегія заснована на двійковій системі числення. Позиції фішок Бутон визначив як небезпечну і безпечну. Позиція яка утворилася після ходу гравця, і гарантує перемогу, називається безпечною, якщо не гарантує перемоги – небезпечна. Бутон довів, що будь яку небезпечну позицію можна перетворити на безпечну, якщо правильно зробити хід. Якщо перед ходом гравця вже склалася безпечна позиція, то будь який хід перетворює позицію в небезпечну. Оптимальна стратегія полягає в наступному: кожним ходом потрібно перетворювати небезпечну позицію в безпечну і змушувати суперника руйнувати її [].

Використовувати оптимальну стратегію доцільно, коли гравець відкриває гру і початкова позиція фішок небезпечна або коли він робить другий хід, а початкова позиція безпечна.

Для визначення безпечна чи небезпечна позиція, потрібно записати в двійковій системі числення кількість фішок в кожному ряді. Якщо сума в кожному стовпці дорівнює нулю або є парною – позиція безпечна. Якщо сума непарна хоча б в одному стовпці – позиція небезпечна.

Таблиця 2.1 – Двійковий запис чисел від 1 до 20

	16	**8**	**4**	**2**	**1**		**16**	**8**	**4**	**2**	**1**
1					1	**11**		1	0	1	1
2				1	0	**12**		1	1	0	0
3				1	1	**13**		1	1	0	1
4			1	0	0	**14**		1	1	1	0
5			1	0	1	**15**		1	1	1	1
6			1	1	0	**16**	1	0	0	0	0
7			1	1	1	**17**	1	0	0	0	1

8		1	0	0	0	**18**	1	0	0	1	0
9		1	0	0	1	**19**	1	0	0	1	1
10		1	0	1	0	**20**	1	0	1	0	0

В двійковій системі записується число фішок (кількість фішок в ряді), які розставлені по схемі 3-4-5 та сумуються:

	4	2	1
3		1	1
4	1	0	0
5	1	0	1
	2	1	2

Сума цифр в середньому стовпці непарна, дорівнює 1. Отже дана позиція є небезпечною. Щоб зробити дану позиція безпечною першому гравцеві потрібно забрати з першого ряду дві фішки. Таким чином сума стає парною:

	4	2	1
1			1
4	1	0	0
5	1	0	1
	2	0	2

Якщо перший гравець забере не дві фішки з першого ряду, а наприклад, дві фішки з третього ряду:

	4	2	1
3		1	1
4	1	0	0
3	0	1	1
	1	2	2

Сума ряду непарна, оскільки сума в першому стовпці рівна 1.

Перебір інших можливих комбінацій, не дав позитивних результатів. Отже, тільки забравши дві фішки першого ряду можна зробити позицію безпечною.

Аналіз дерева рішень схеми 3-4-5 є досить громіздким, оскільки є багато можливих ходів для гравців. Тому проаналізуємо дерево рішень для схеми

2-3. Дану схему можна використовувати як повноцінну гру, також дана схема є однією з частин схеми 3-4-5.

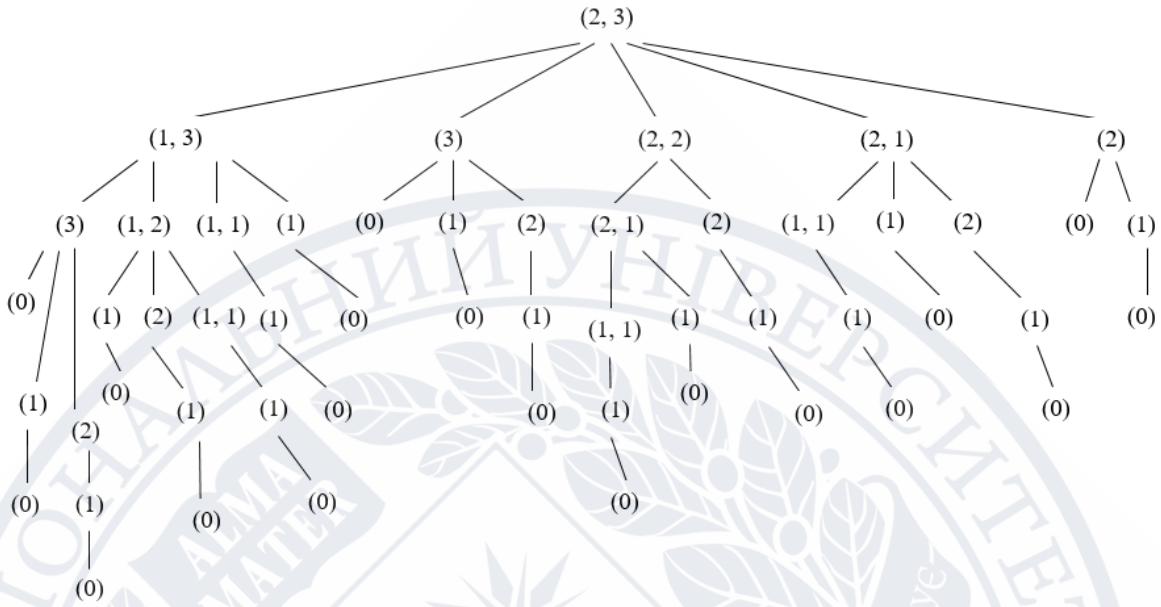


Рисунок 2.2 – Дерево рішень за схемою 2-3

Дане дерево рішень, аналізує кількість можливих позицій для гри. Варто зауважити, що для більшої кількості рядків та відповідно більшої кількості фішок в рядках, дерево рішень буде більшим і аналізувати його буде важче. В цілому дерево рішень дієвий алгоритм, але в більшості варіантів громіздкий та забирає багато часу.

Жадібний алгоритм, в порівнянні з деревом рішень, не такий складний та не забирає багато часу. Одна з гілок дерева рішень наочно показує дію жадібного алгоритму (рис. 2.1).

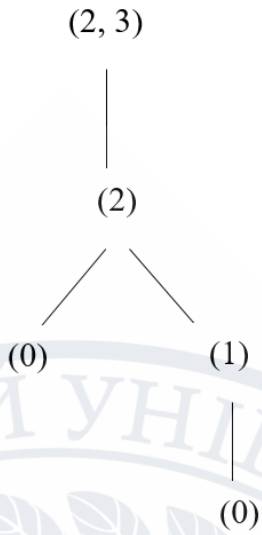


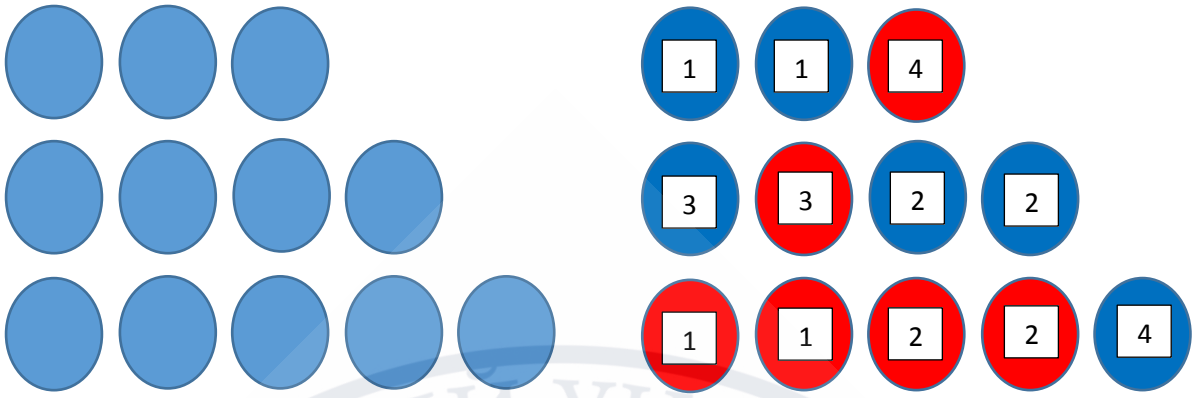
Рисунок 2.3 – Фрагмент дерева рішень для схеми 2-3

Перший гравець програє, якщо буде грати за таким алгоритмом, оскільки другий гравець забере дві останні фішки.

З більшою кількістю рядів та фішок такий алгоритм може принести і виграш.

Повернемося до схеми 3-4-5 та проаналізуємо, за даною схемою, симетричний алгоритм. Ходи першого гравця зафарбовані синім кольором, ходи другого гравця – червоним.

Другий гравець обрав симетричний алгоритм для гри. В результаті гри виграє другий гравець, оскільки робить останній хід. Можна зробити висновок, що симетричний алгоритм є досить дієвим для гри.



а) б)

Рисунок 2.4 – Ігрове поле гри «Нім»: а) початкова ігрова поле; б) ігрове поле після гри (цифри – номера ходів гравців)

В 1940 році була створена одна з найперших машин для гри в нім «Німатрон». Одним із її творців був Е. У. Кондон. «Німатрон» виставлявся на виставці в Нью-Йорці, було зіграно 100 000 партій з яких машина виграла 90 000.

2.2. Гра «Так-Тікс»

Гра Так-Тікс [6, 11] відноситься до логічних ігор або математичних головоломок. В основі гри лежить ідея рядів фішок, які перетинаються між собою. Гравець для гри може обрати поле будь якої розмірності, суть гри не змінюється, а от складність зростає.

Звичайна гра є досить тривіальною через простоту стратегії, тому в Так-Тікс грають «навпаки» - програє той, хто робить останній хід. Якщо гра ведеться на квадратній дошці з непарною кількістю клітинок ігрового поля, гра є виграшною для першого гравця, якщо він забере фішку з центральної клітинки і буде робити ходи симетричні ходам суперника. Якщо гра ведеться на дошці з парною кількістю клітинок ігрового поля, виграє другий гравець роблячи ходи симетричні ходам суперника.

В книзі М. Гарднера [6] описана теорія для гри Так-Тікс та наведені деякі приклад гри, з поясненням для ходів гравців. Також було висунуто припущення, що на будь якій дошці, яка має хоча б одну непарну сторону, виграє той хто починає гру, за умови, що забере на першому ході весь середній ряд. Якщо дошка має всі парні сторони, виграє завжди другий гравець. Існує гіпотеза, що на дошці з парною кількістю полів виграє другий гравець, на полі з непарною кількістю полів – виграє перший гравець.

На даний момент ні гіпотеза ні припущення не є доведеними. Гра Так-Тікс досі не має повної теорії.

У класичному варіанті гри, який запропонував П. Хейн, гра ведеться но дошці розміром 4x4.

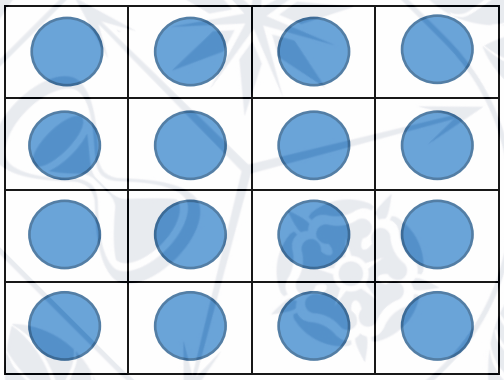


Рисунок 2.5 – Ігровий простір для гри «Так-Тікс»

Два гравці по черзі роблять ходи. За один хід гравець може зняти з дошки певну кількість фішок. Кількість фішок, які можна зняти за один хід, визначається гравцями на початку гри, але їх кількість не може перевищувати кількості фішок в ряді або стовпці. Для гри 4x4 за один хід можна зняти від 1 до 4 фішок. Варто зазначити, що якщо за один хід гравець знімає більше однієї фішки з ігрового поля, то такі фішки мають розташовуватися поруч. Гравець знімає фішки з горизонтального або вертикального ряду, або частини ряду [6, 10-11].

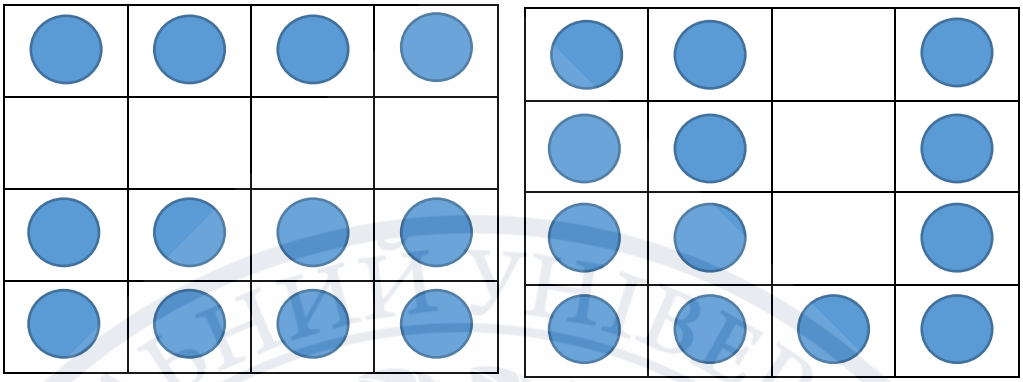


Рисунок 2.6 – Можливі ходи гравців в грі Так-Тікс

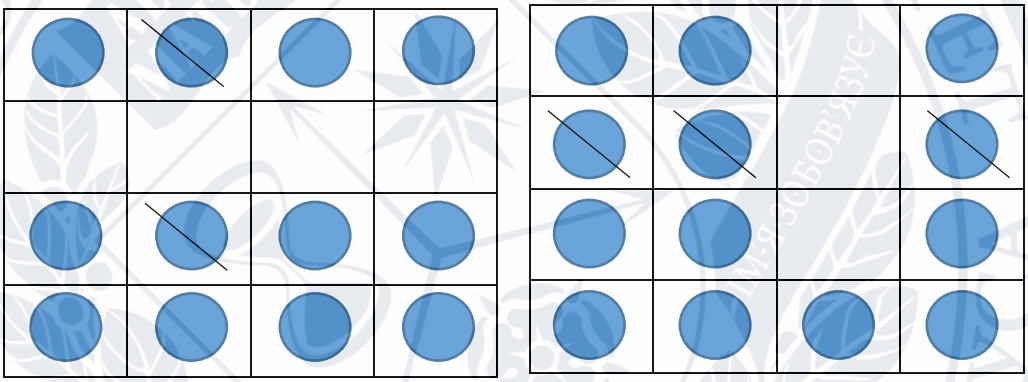


Рисунок 2.7 – Не можливі ходи для гравців в грі Так-Тікс

Проаналізуємо застосування симетричного алгоритму для гри. Ходи першого гравця позначаються синім кольором, ходи другого гравця позначається червоним кольором. Другий гравець дотримується симетричного алгоритму відносно ходів першого гравця.

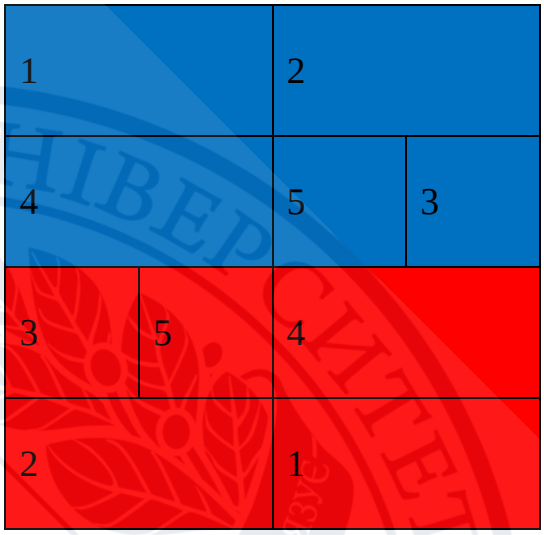
I	II
1-2	15-16
3-4	13-14
8	9
5-6	11-12

7	10

Другий гравець програв, оскільки на останньому ході забрав останню фішку.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

а)



б)

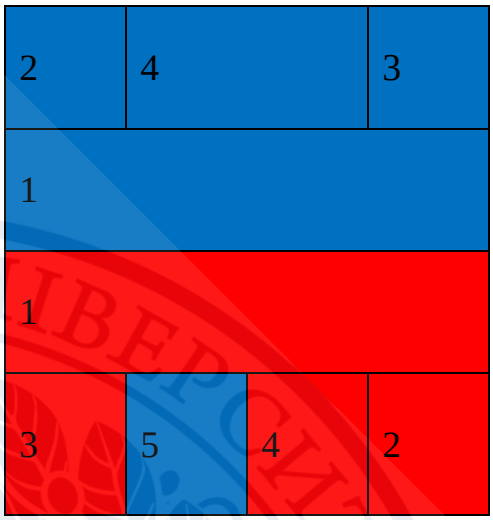
Рисунок 2.8 – Ігрове поле гри Так-Тікс: а) початкова ігрова дошка; б) ігрова дошка після гри (цифри – послідовність ходів гравців)

В даній таблиці наведено послідовність ходів гравців, другий гравець дотримувався симетричної стратегій, але на 4 ході змінив її та виграв.

I	II
5-6-7-8	9-10-11-12
1	16
4	13
2-3	15
14	

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

а)



б)

Рисунок 2.9 – Ігрове поле гри Так-Тікс: а) початкова ігрова дошка; б) ігрова дошка після гри (цифри – послідовність ходів гравців)

Симетричний алгоритм є досить простим у реалізації та не займає багато часу для аналізу та самої гри. Звичайного для повного перебору всіх можливих позицій, потрібно досить багато часу та зусиль, але за допомогою симетрії грати досить просто, головне блокувати ходи суперника, які можуть привести його до перемоги.

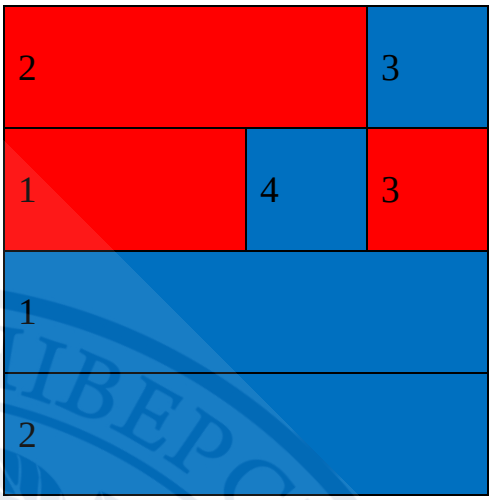
Жадібний алгоритм: за правилами максимальна кількість фішок, які можна вилучити з ігрового поля, рівна 4 по горизонталі або вертикалі. Перший гравець буде дотримуватися жадібного алгоритму, другий гравець не буде робити ходи за допомогою даного алгоритму.

I	II
9-10-11-12	5-6
13-14-15-16	1-2-3
4	8
7	

Останнім хід робить перший гравець і відповідно програє.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

а)



б)

Рисунок 2.10 – Ігрове поле гри Так-Тікс: а) початкова ігрова дошка; б) ігрова дошка після гри

2.3. Гра «Гекс»

Гекс – математична гра на дошці у формі ромба, яка має гексагональну сітку [6].

Гра створена П. Хейном в 1942 році та незалежно від нього Дж. Нешом в 1948 році. До 1950-х років гра мала назву «Багатокутники», грали гру не на дошках а на папері. З часом почали виготовляти дошки для гри, спеціальні блокноти в яких були надруковані зображення дошок для гри. В 1950-х роках гра отримала назву гекс.

В гекс грають на дошці у вигляді ромба, який покритий шестикутними клітинками. Дошка може бути будь якого розміру і будь якої форми, але традиційно використовують форму ромба розмірності n x n, найчастіше 11x11.

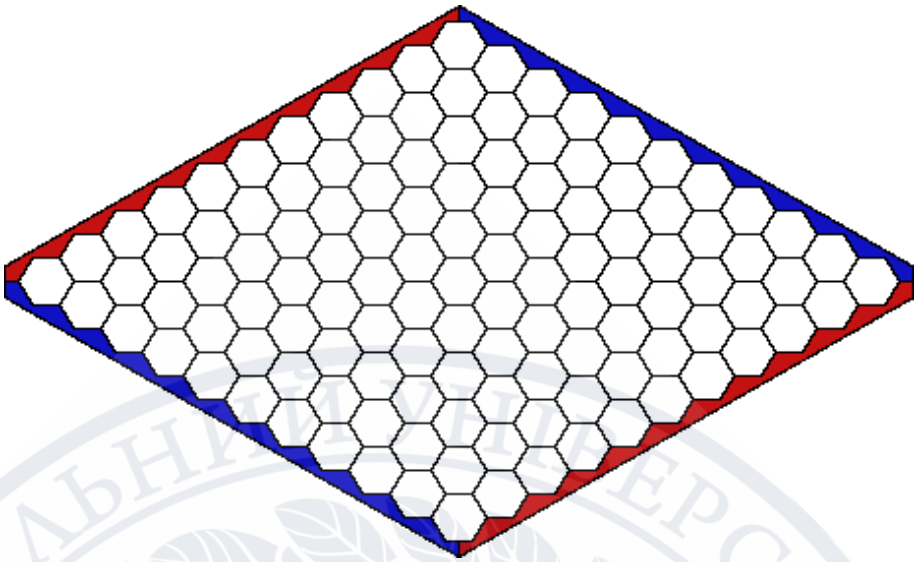


Рисунок 2.11 – Дошка для гри Гекс, розмірності 11x11

Також популярними є дошки розмірності 14x14 та 19x19. Джон Неш вважав, що оптимальним розміром дошки є 14x14 [19]. Один з гравців грає синіми фішками (каменями або монетами), інший – червоними (фішки можуть бути будь якого іншого кольору). Гравці по черзі ставлять фішки свого кольору на будь яке вільне поле. Починають сині. Дві протилежні сторони дошки зафарбовані синім та червоним кольором відповідно. Ігрові поля на краях дошки відносяться до обох сторін. Для того щоб виграти, гравець повинен побудувати ланцюг із своїх фішок, та з'єднати ним сторони свого кольору.

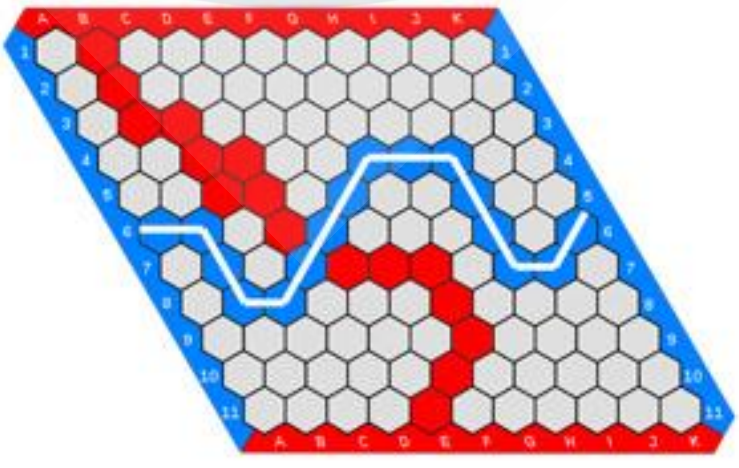


Рисунок 2.12 – Виграшна партія в Гекс для гравця, який грав синіми фішками

Використання жадібного алгоритму не є доцільним для даної гри, оскільки за один хід гравець може поставити лише одну фішку на ігрове поле. Загалом в цій грі він і не потрібний, оскільки основна ціль гри, побудувати з’єднання між двох протилежних сторін, а не захопити як можна більше ігрового простору.

Перебрати всі можливі комбінації за допомогою дерева рішень, також складно. Оскільки дошка розмірності 11x11, має 121 ігрове поле. Крім того, не завжди доцільно йти по прямій лінії, тобто кількість фішок які будуть використані для побудови ланцюга з’єднання не є завчасно відомою.

Симетричний алгоритм можливий, але варто уважно робити ходи і намагатися передбачити ходи суперника.

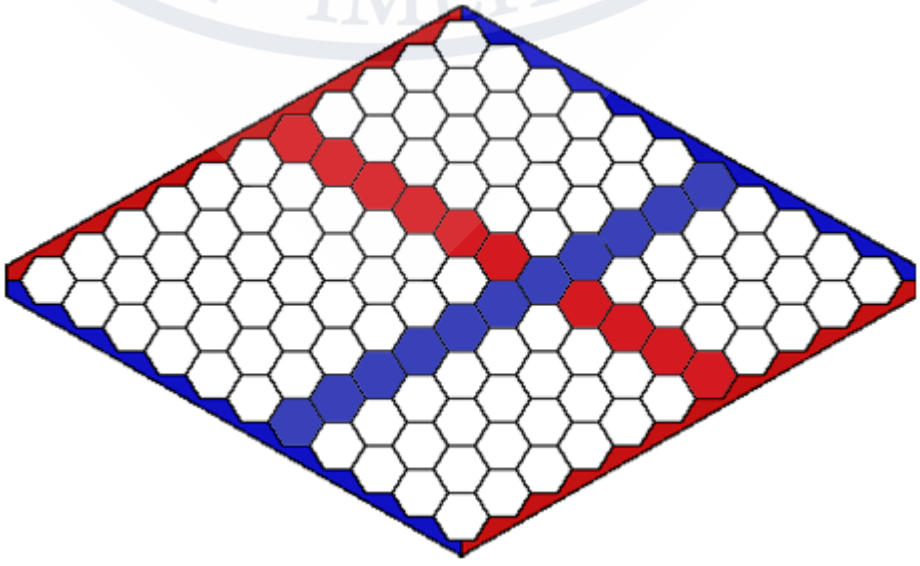


Рисунок 2.13 – Симетричний алгоритм для гри Гекс

Існує декілька варіантів гри гекс:

-**гра Y:** узагальнення гри гекс, використовують трикутну дошку; метою гри є побудова ланцюга який зі своїх каменів, що з’єднує три сторони [20].

Рисунок 2.14 – Дошка для гри Y

- **Гавана:** створена К. Фрілінгом в 1976 році, відноситься до того самого класу, що і гекс; використовується шестикутна дошка та існує декілька виграшних формувань: кільце – замкнутий ланцюжок з каменів свого кольору, міст – ланцюжок, що з’єднує будь-які дві кутові клітинки дошки або вилка – ланцюжок, що з’єднує три сторони дошки .

Рисунок 2.15 – Дошка для гри Гавана з виграшними позиціями

- **Пекс:** гра майже ідентична гексу, головна відмінність, в тому, що використовують дошку розбиту на п’ятикутники неправильної форми, унікальна особливість такої дошки в тому, що одна половина клітинок ігрового поля має по п’ять «сусідів», інша половина – сім «сусідів»; дану особливість використовують в унікальних тактичних ходах, які не існують в гексі [21].

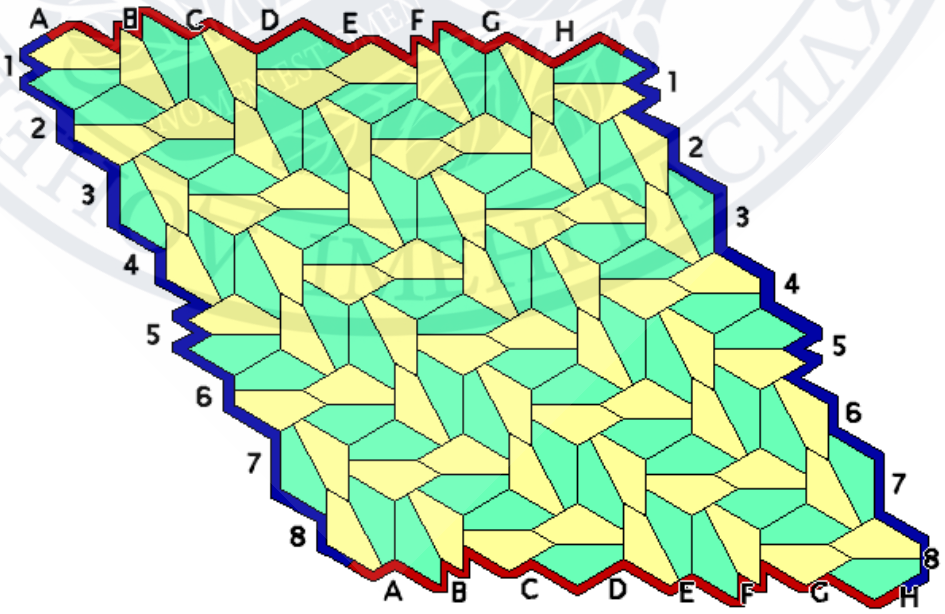


Рисунок 2.16 – Дошка для гри Пекс

- **Атлолл:** узагальнення гри гекс, використовують дошку спеціальної форми; дошка в даній грі містить більшу кількість сторін, але мета гри співпадає з метою гри гекс – побудувати ланцюжок із камінців свого кольору, який з’єднує дві протилежні сторони того ж кольору; виграшна позиція може мати декілька частин, які з’єднують кожну з протилежних сторін [22].

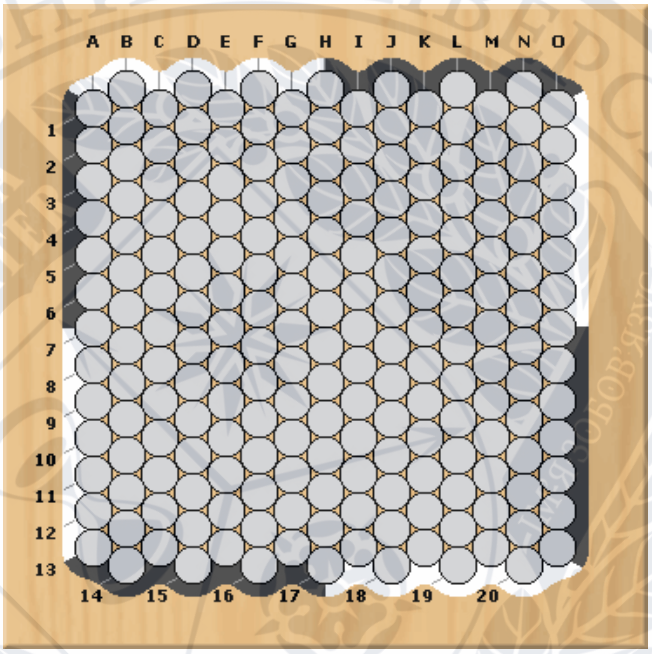
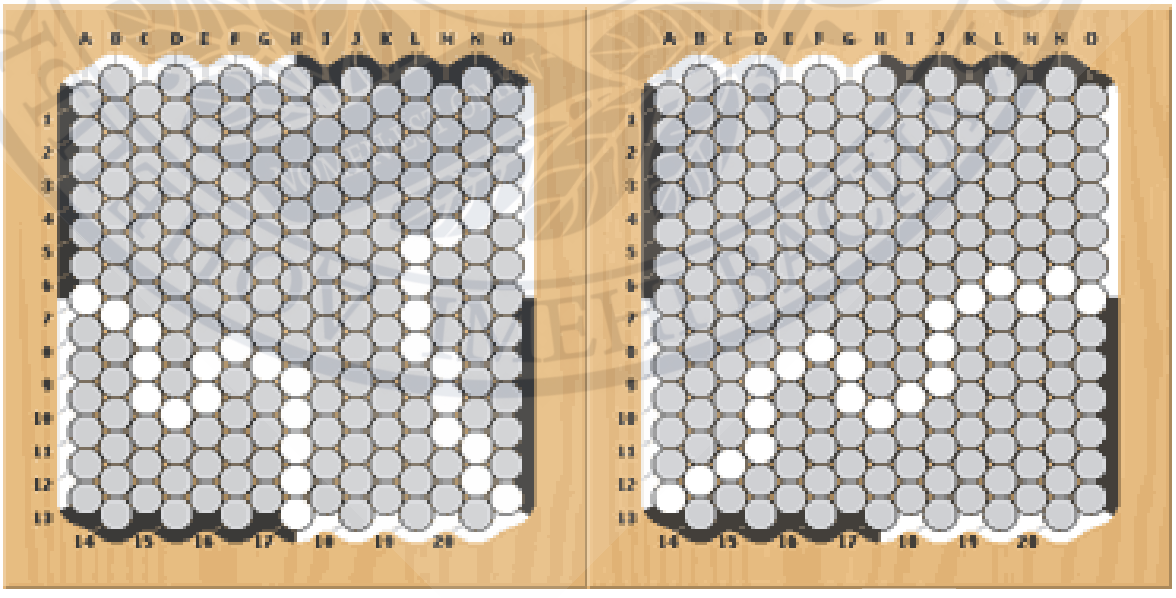


Рисунок 2.17 – Дошка для гри Аттол



а) б)

Рисунок 2.18 – Приклади виграшу у грі Аттол: а) дві сторони з’єднані за допомогою третьої сторони; б) пряме з’єднання

- Некс: в даній гра окрім фішок двох кольорів використовують фішки нейтрального кольору (не належить жодному з гравців); існує два можливих варіанта ходу: покласти один камінь свого кольору і один камінь нейтрального кольору або замінити два нейтральних камені на дошці на камені свого кольору і замінити один із своїх каменів на камінь нейтрального кольору [23].

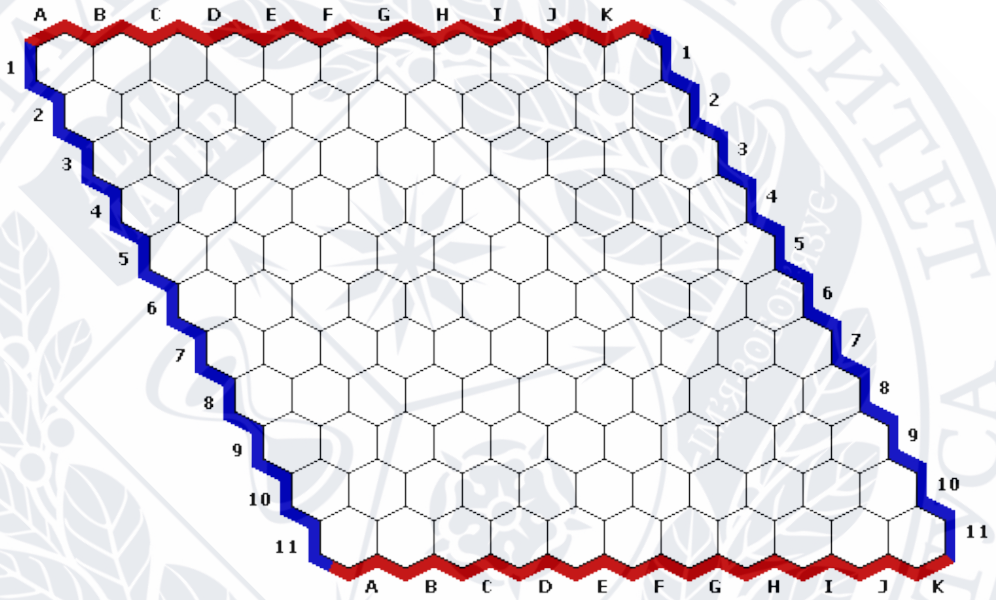


Рисунок 2.19 – Дошка для гри Некс

2.4. Гра «Клоббер»

Клоббер – абстрактна стратегічна гра для двох гравців, винайдена у 2001 році теоретиками комбінаторних ігор М. Х. Альбертом, Дж. П. Гроссманом та Р. Новаковським [24].

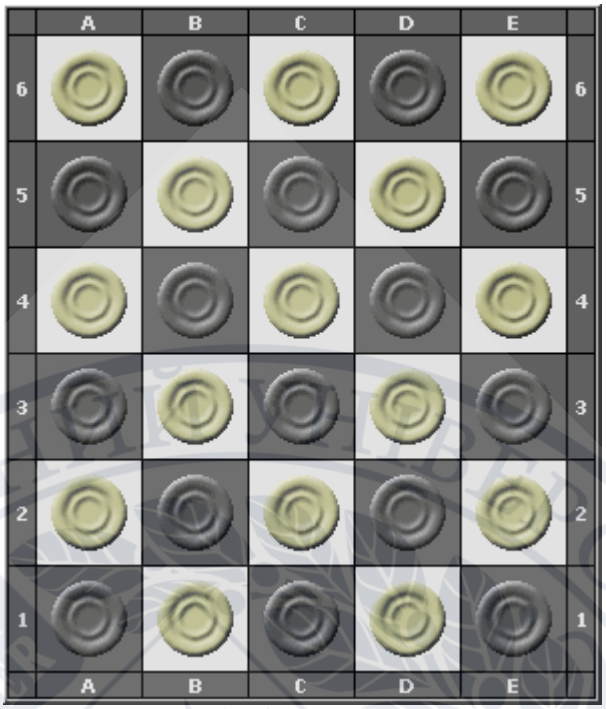


Рисунок 2.20 – Дошка для гри Клоббер

Гра для двох гравців, яку грають на прямокутній біло-чорній шаховій дошці. На початку гри кожен з квадратів зайнятий фішками або каменем. Білі фішки розташовані на білих клітинках, чорні фішки на чорних квадратах. Гравці по черзі роблять ходи, переміщують одну із своїх фішок на ортогональну сусідню фішку супротивника, вилучаючи її з гри. Переможцем вважається той, хто робить останній хід.

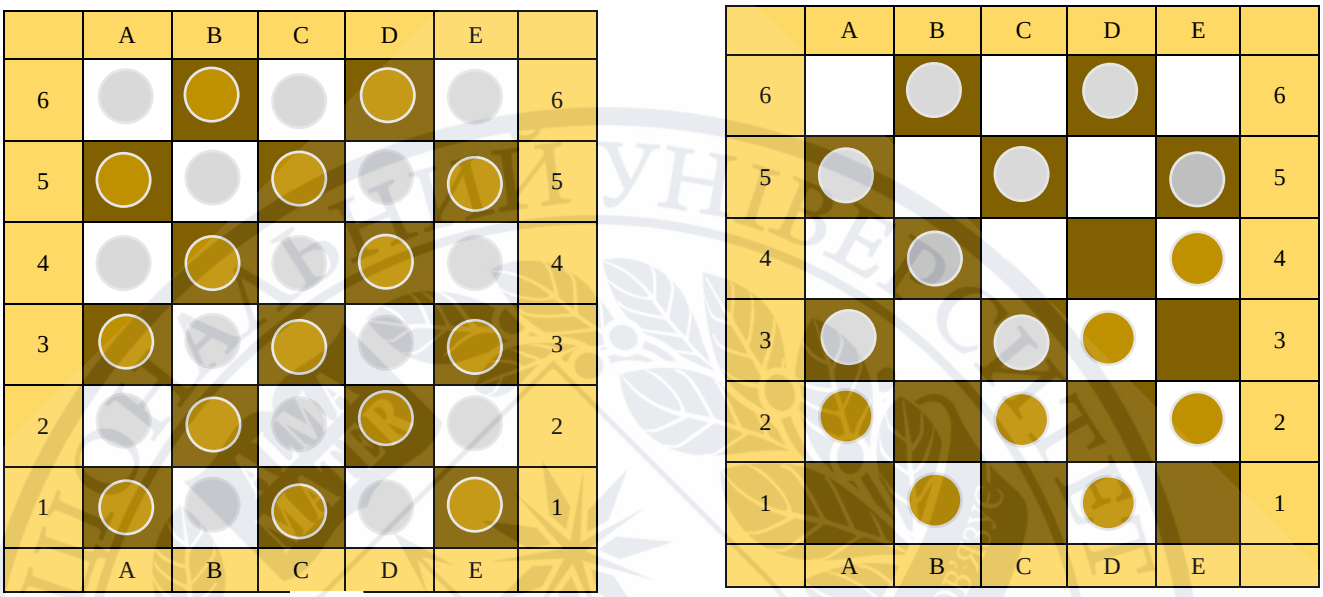
Проаналізувати гру клоббер за допомогою дерева рішень є досить складно. Є велика кількість можливих комбінацій для ходів, дерево рішень є досить великим та складним.

Жадібний алгоритм не є доцільним, оскільки гравець за один хід може захопити лише одну фішку суперника.

Симетричний алгоритм є доцільним. Переможцем є перший гравець, оскільки він зробив останній хід.

I	II
A5	E2
B6	D1
C5	C2
D6	B1

E5	A2
B4	D3
A3	E4
C3	



а) б)

Рисунок 2.21 – Приклад гри клоббер: а) початкова дошка; б) дошка після гри

Висновок до розділу 2

В даному розділі були розглянуті деякі з комбінаторних ігор, а саме нім, так-тікс, гекс та клоббер. Були розглянуті правила, особливості та варіації тієї чи іншої гри. Також було розглянуто застосування того чи іншого алгоритму для гри.

РОЗДІЛ 3
СТВОРЕННЯ ПРОГРАМИ ДЛЯ КОМБІНАТОРНИХ ІГОР «НІМ» ТА «ТАК-ТІКС»

3.1. Аналіз програми та коду

Код програми для гри написаний на мові Python3. Програма має консольний інтерфейс. В програмі представлено наступні ігри: «Так-Тікс» розмірністю 5x5 та 4x4, а також гра «Нім». Гравець має можливість при запуску програми обрати гру, яку бажає зіграти (рис. 3.1).

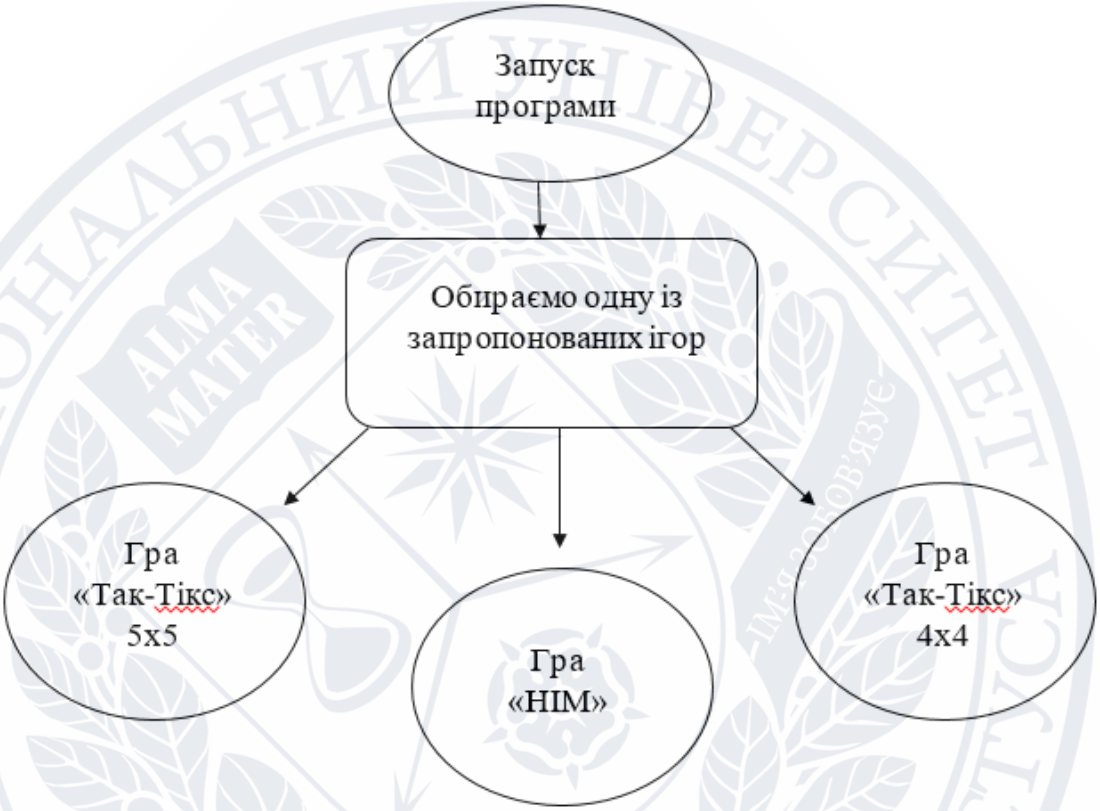


Рисунок 3.1 – Запуск програми та вибір однієї з представлених ігор

Лістинг 3.1

```
print("""
Виберіть гру яку бажаєте зіграти:
        1 - "Так-Тікс" 5x5
        2 - "Так-Тікс" 4x4
        3 - "Нім"
""")
chosen_game = None
error_text = ""
while chosen_game != '1' and chosen_game != '2' and
chosen_game != '3':
    chosen_game = input(error_text)
    error_text = "Неправильный варіант, введіть 1 чи 2 або
3\n"

chosen_game = int(chosen_game)
```

Рисунок 3.2 – Запуск гри «Так-Тікс»

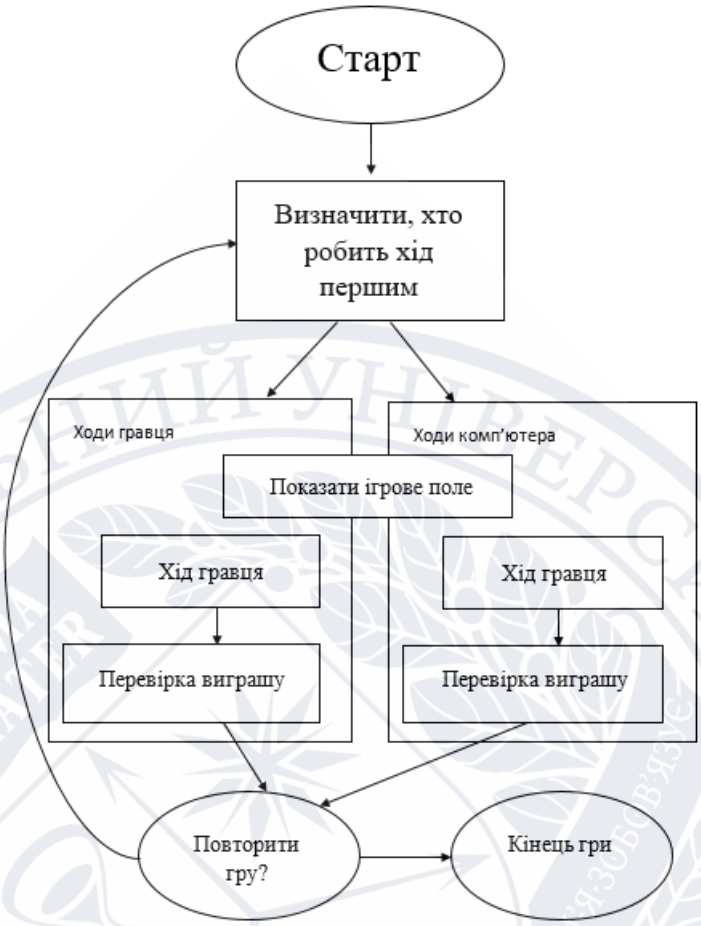


Рисунок 3.3 – Запуск гри «Нім»

Ігрове поле

Ігровим полем для гри Так-Тікс є дошка розмірності n x n. Полем для гри Нім є ряд з n монет. Під час гри буде ігрове поле буде оновлюватися відповідно до ходів.

Для гри Так-Тікс поле представлене і вигляді списку рядків. В кожному рядку по чотири комірки, де будуть розташовуватися мітки гри, в кожної комірки є відповідні індекси.

Індекси комірок, ігрового поля, представлені наступним чином (рис. 3.4)

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Рисунок 3.4 – Індексація комірок ігрового поля

Для побудови ігрового поля, гри Так-Тікс, була використана функція drawBoard(board).

Лістинг 3.2

```
def drawBoard(board):
        print ('    |    |    |    |')
        print (' '  + board[21] + '  |  '  + board[22] + ' |
' + board[23] + '| ' + board[24]  + '  | ' + board[25])
        print ('    |    |    |    |')
        print ('------------------------')
        print ('    |    |    |    |')
        print (' '  + board[16] + '  |  '  + board[17] + ' |
' + board[18] + '| ' + board[19]  + '  | ' + board[20])
        print ('    |    |    |    |')
        print ('------------------------')
        print ('    |    |    |    |')
        print (' '  + board[11]  + '  | ' + board[12]  + '  |
' + board[13] + '| ' + board[14]  + '  | ' + board[15])
        print ('    |    |    |    |')
        print ('------------------------')
        print ('    |    |    |    |')
        print (' '  + board[6]  + '  | ' + board[7]   + '  |
' + board[8]  + '| ' + board[9]  + '  | ' + board[10])
        print ('    |    |    |    |')
        print ('------------------------')
        print ('    |    |    |    |')
```

```
        print (' '  + board[1]  + '  | ' + board[2]   + '  |
' + board[3]  + '| ' + board[4]  + '  | ' + board[5])
        print ('    |    |    |    |')
```

Побудова ігрового поля для гри Нім, відбувається за допомогою функції display_board(board).

Лістинг 3.3

```
def displayboard(board):
    print("\n-------------------------------------------------
-----------------------------")
    print("      ",*board, sep="   ")
    print("-------------------------------------------------------
-------------------------\n")
```

Використані функції, хоч і мають різні назви, але виконують однакове застосування.

Вибір ігрових позначень

Для гри Нім ігрових позначень немає, оскільки маємо сталий набір монеток, які по черзі гравці забирають з поля. Додаткових позначень для даного процесу не потрібно.

Гра Так-Тікс передбачає захоплення певного ігрового простору. Саме тому для

Наочності будуть використані певні позначення. Після запуску програми та вибору гри, гравець матиме можливість обрати мітку, якою буде грати. Функція inputPlayerLetter робить запит до гравця, яке позначенні він обирає X чи O. Гра не розпочнеться, якщо гравець не введе відповідну літеру. Таким чином, за допомогою даної функції, буде повернено список з двох елементів, один з яких належить гравцеві, а інший – комп’ютеру.

Лістинг 3.4

```
def inputPlayerLetter():
        # Гравець обирає якою міткою (буквою) буде
заповнювати клітинки
        letter = ''
        while not(letter == 'X' or letter == 'O'):
            print ('Яку літеру обираєте X чи O?')
            letter = input().upper()
        # Перший елемент відповідає гравцю, а другий елемент
відповідає комп'ютеру
            if letter == 'X':
                return ['X','O']
            else:
                return ['O','X']
```

Найважчий вибір: хто ходить першим

Щоб визначити хто буде ходити першим були використані наступні функції: whoGoesFirst() та ?.

Функція whoGoesFirst() визначає за допомогою рандому, хто буде робити першим хід. Якщо випадає 0 – першим ходить комп’ютер, якщо випадає 1 – першим ходить гравець.

Лістинг 3.5

```
def whoGoesFirst():
        if random.randint(0,1) == 0:
            return 'computer'
        else:
            return 'player'
```

Функція whostartsfirst не використовує рандом для визначення прешості, гравцеві потрібно ввести з клавіатури літеру F(f) – якщо бажає бути першим або літеру S(s) – якщо не бажає розпочинати гру першим. Відповідно до того яку літеру обрав гравець, комп’ютеру присвоюється інша .

Лістинг 3.6

```
def whostartsfirst(question):
    choice = None
    while choice not in ("F", "S","f","s"):
        choice = input(question)
```

```
        if choice.upper() == "F":
            f_player = "USER"
            s_player = "USER"
        elif choice.upper() == "S":
            f_player = "CMP"
            s_player  = "CMP"
        else:
            print("\nНеправильний вибір. Зробіть вибір ще
раз.")

    return f_player, s_player
```

Зіграти ще раз?

Іноді гра так захоплює, що не можливо відірватися. В таких випадках використовують функцію повтору гри. Таку функцію можна записати кількома варіантами.

Функція playAgain() використовується у грі Так-Тікс, якщо гравець бажає зіграти ще раз, він повинен ввести з клавіатури літеру y або написати yes. У випадку коли гравець відмовляється грати ще раз, потрібно ввести літеру n або написати no. Будь-яка інша відповідь автоматично буде визначатися помилкою та буде завершувати гру.

Лістинг 3.7

```
def playAgain():
        # Функція для повторної гри (коли гравець хоче
зіграти ще раз)
        print('Бажаєте зіграти ще раз?(yes чи no)')
        return input().lower().startswith('y')
```

Для гри Нім використовується функція keepplaying(). Щоб продовжити гру потрібно ввести літеру y або Y, для завершення гри n або N. Інша відповідь автоматично рахується, як завершення гри.

Лістинг 3.8

```
def keepplaying():
    while True:
        another_go = input("\nБажаєте зіграти ще раз?[Y/N]:
")
```

```
        if another_go in ("y","Y"):
            return True
        elif another_go in ("n","N"):
            return False
        else:
            print("\nНедійсний вибір. Повторіть спробу.")
```

Виграш та завершення гри

В грі Так-Тікс програє той, хто робить останній хід (заповнює останню клітинку на ігровому полі). Використана функція isWinner, перевіряє, хто останній робив хід, та робить висновок про поразку. На даному етапі гра є завершеною і виникає питання чи буд гравець грати ще раз. Нічиї бути не може.

Лістинг 3.9

```
def isWinner(board, letter, getComputerMove):
        # Програє той, хто заповнює останню клітинку поля
        if isBoardFull(board):
            return True
        return False
```

Для гри Нім використана функція game_over(). За правилами гри програє той, хто забирає останню монетку. Таким чином, функція перевіряє довжину рядка з монетками, і за допомогою циклу робить наступні висновки: якщо на момент ходу гравця залишається одна монетка, то він програв; якщо на момент ходу комп'ютера залишилася одна монетка, то відповідно програв комп'ютер. Нічиї буди не може.

Лістинг 3.10

```
def gameover(board,f_player):
    if len(board) == 1 and f_player == "USER":
        winner = "КОМП'ЮТЕР"
        loser = "ГРАВЕЦЬ"
    elif len(board) == 1 and f_player == "CMP":
        winner = "ГРАВЕЦЬ"
        loser = "КОМП'ЮТЕР"
    else:
```

```
        winner = None

    return winner, loser
```

Оновлення ігрового простору

Важливим є також оновлення ігрового простору. Оскільки гравці по черзі роблять ходи, важливо наочно бачити скільки можливих варіантів залишилося. Для гри Нім оновленні ігрового поля є досить простим. Після кожного ходу гравців, рахується довжина ігрового поля. Також варто пам’ятати про обмежену кількість монеток, які можна взяти за один хід. При перевищенні максимально допустимого числа буде з’являтися попередження.

Лістинг 3.11

```
def updateboard(board,move):
    valid_moves = (1,2)
    if len(board) > move:
        for i in  range(move):
            board.remove("O")
        return board
    elif len(board) <= move:
        if move in valid_moves:
            for i in range(move):
                board.remove("O")
            return board
        else:
            print("\n" + str(move) + " монет не можна
взяти.\nЗроблений хід є неприпустимим. Повторіть ще раз.")
            return board
```

Гра Так-Тікс має деякі нюанси оновлення ігрового простору. За правилами, можна обирати клітинки які розташовані поруч, потрібно перевіряти чи є вільне місце на ігровому поля, також потрібно перевіряти чи на скільки заповнене ігрове поле. Якщо у всіх клітинках є значення то гра – завершена.

Лістинг 3.12

```
def getBoardCopy(board):
        dupeBoard = []
        for i in board:
            dupeBoard.append(i)
```

```
        return dupeBoard

    def isSpaceFree(board,  move):
        return board[move] == ' '
```

Ходи

Ходи гравця та комп'ютера майже не відрізняють. Гравець і комп'ютер мають дотримуватися правил прописаних у грі.

У грі Так-Тікс використовується функція getComputerMove() для ходів комп'ютера, в якій також визначається якою літерою буде позначатися хід. Також для правильного ходу використовується функція chooseRandomMoveFtomList(). Дана функція визначає вільне місце на ігровому полі і надає можливість робити рандомний хід відповідно до створеного списку ходів. Після кожного наступного ходу дана функція оновлюється.

Лістинг 3.13

```
def chooseRandomMoveFromList(board, moveList):
        possibleMoves = []
        for i in moveList:
            if isSpacefree(board, i):
                possibleMoves.append(i)
            if len(possibleMoves) !=0:
                return random.choice(possibleMoves)
            else:
                return None

    def getComputerMove(board, computerLetter):
        if computerLetter == 'X':
            playerLetter = 'O'
        else:
            playerLetter = 'X'
        for i in range(1,26):
            copy = getBoardCopy(board)
            if isSpaceFree(copy, i):
                makeMove(copy, computerLetter, i)
                return i
        raise Exception("Board is full")
```

Хід гравця є простішим, оскільки гравцеві не потрібно додаткових функцій для здійснення ходу. Використовується функція PlayerMove(), яка просить ввести номер комірки або комірок для створення ходу. В даній функції використаний цикл, який дозволяє обрати більше однієї клітинки (оскільки це є дозволено правилами), якщо гравець не хоче за раз обирати всі можливі клітинки то просто потрібно натиснути Enter.

Лістинг 3.14

```
def getPlayerMove(board, move_index, previous_moves):
        def isValidMove(move):
            moves = previous_moves + [move]
            return all( (m-1) // 5 == (moves[0]-1) // 5 for m
in moves ) or all( m % 5 == moves[0] % 5 for m in moves)
        move = ' '
        possible_moves ='1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25' .split()
        while move not in possible_moves or not
isSpaceFree(board, int(move)) or not isValidMove(int(move)):
            if move_index == 0:
                print ('Який наступний хід? (1-25)')
            else:
                if move in possible_moves and not
isValidMove(int(move)):
                    print('Ви можете обрати тільки ті
клітинки, що стоять в одному рядку або стовбці з вже
вибраннними')
                text_cells = 'клітинки'
                if 5 - move_index == 1:
                    text_cells = 'клітинку'
                print ('Ви можете обрати ще ' + str(5 -
move_index) + ' ' + text_cells + ' (1-25). Нажміть ентер щоб
більше не вибирати')
                if len(move) == 0:
                    return None
            move = input()
        return int(move)
```

Для гри Нім ходи гравця та комп’ютера також є досить схожими.

Лістинг 3.15 (Хід гравця)

```
def userspick(question,minStick,maxStick,userIn,board):
    print("\n--ХІД ГРАВЦЯ--")
    while userIn not in range(minStick, maxStick) or userIn
>= len(board):
```

```
        try:
            userIn = int(input(question))
            if userIn not in range(minStick, maxStick) or
userIn >= len(board):
                print("\nВи не можете обрати стільки монет!")
                userIn = int(input(question))

        except Exception as e:
            print("\nВиникла помилка.\nError: " + str(e) +
"\nПовторіть вибір ще раз.")

    f_player = "СМР"

    return userIn, f_player
```

Лістинг 3.16 (Хід комп'ютера)

```
def
computersmove(board,userIn,s_player,winning_position,earlier_
move):
    best_move = 1
    print("\n--ХІД КОМП'ЮТЕРА--")

    if s_player == "СМР" and winning_position == False:
        if len(board) % 4 ==1:
            best_move = 0
            while best_move not in range(1,len(board)+1):
                best_move = random.randint(1,3)
        else:
            if userIn + earlier_move < 4:
                best_move = 4 - (userIn + earlier_move)
                winning_position = True
            elif userIn + earlier_move > 4:
                best_move = 8 - (userIn + earlier_move)
                winning_position = True

    elif s_player == "USER" or winning_position == True:
        best_move = 4 - userIn

    earlier_move = best_move

    print("\nКомп'ютер робить хід ......")

    print(": " + str(best_move) + " монету(и).")
    for sticks in range(best_move):
        board.remove("O")

    f_player = "USER"
    return board, f_player, winning_position, earlier_move
```

Створені функції не можуть самі по собі створити повноцінну програму, їх потрібно об’єднати для роботи програми в цілому. Для цього створюють головну функцію main().

Лістинг 3.17 (Головна функція для гри «Так-Тікс»)

```
while True:
        theBoard = [' ']*26
        playerLetter, computerLetter = inputPlayerLetter()
        turn = whoGoesFirst()
        print('  ' + turn + ' буде першим')
        gameIsPlaying = True
        while gameIsPlaying:
            if turn == 'player':
                drawBoard(theBoard)
                prev_moves = []
                for i in range(5):
                    move = getPlayerMove(theBoard, i,
prev_moves)
                    if move is None:
                        turn = 'computer'
                        break
                    makeMove(theBoard, playerLetter, move)
                    prev_moves.append(move)
                    if isBoardFull(theBoard):
                        drawBoard(theBoard)
                        print('Ви програли')
                        gameIsPlaying = False
                    else:
                        turn = 'computer'
            else:
                move = getComputerMove(theBoard,
computerLetter)
                makeMove(theBoard, computerLetter, move)

                column_or_row = random.choice(['column',
'row'])
                amount = random.randint(0, 4)
                for i in range(amount):
                    if column_or_row == 'row':
                        new_move = move + i+1
                        if (new_move-1) // 5 != (move-1) //
5:
                            break
                    else:
                        new_move = move + 5*(i+1)
```

```
                            print('nm: ' + str(new_move) + ' ' +
str(move))
                            if new_move < 0 or new_move > 25:
                                break
                        if isSpaceFree(theBoard, new_move):
                            makeMove(theBoard, computerLetter,
new_move)
                        else:
                            break

                    if isBoardFull(theBoard):
                        drawBoard(theBoard)
                        print('Ви виграли')
                        gameIsPlaying = False
                    else:
                        turn = 'player'
            if not playAgain():
                break
```

Лістинг 3.18 (головна функція для гри «Нім»)

```
def main():
    anotherGo = True
    welcome_message()
    while anotherGo == True:
        earlier_move = 0
        winning_position = False
        nimBoard = ["O"]*17
        winner = None
        userIn = 0
        print("")
        display_board(nimBoard)
        print("\nГра починається " + str(len(nimBoard)) + "
монет.")
        f_player, s_player = who_starts("\nБажаєте грати
першим чи другим? [F - якщо першим/S - другим]: ")
        while len(nimBoard) != 1:
            if f_player == "USER":

                userIn,f_player = users_pick("\nВведіть свій
вибір [1, 2 чи 3]: ", 1, 4, 20,nimBoard)
                nimBoard = update_board(nimBoard,userIn)

                display_board(nimBoard)
                print("\nЗалишилося " + str(len(nimBoard)) +
" монет.")
            else:
```

```
                nimBoard, f_player,winning_position,
earlier_move =
computersmove(nimBoard,userIn,s_player,winning_position,earli
er_move)
                displayboard(nimBoard)
                print("\nЗалишилося " + str(len(nimBoard)) +
" монет.")

        print("\nКінець...\nЗалишилася тільки одна
монета...")
        game_winner, game_loser = gameover(nimBoard,f_player)

        print("\n " + game_loser + " програв, оскільки
залишилася одна монета")
        print("\nГру виграв...\n: " + str(game_winner))
        anotherGo = keepplaying()

    print("\nДякую за гру!")
    print("\n--ГРА ЗАВЕРШЕНА--")

if __name__ == "__main__":
    main()
```

3.2. Приклад програмної реалізації

Для прикладу було зіграно одну гру «Нім» та одну гру «Так-Тікс» 5 х 5.

Приклад гри «Так-Тікс»:

```
Виберіть гру яку бажаєте зіграти:
        1 - "Так-Тікс" 5x5
        2 - "Так-Тікс" 4x4
        3 - "Нім"

1

------------------------------------------------------------
-----------------
                        ГРА "ТАК-ТІКС"
------------------------------------------------------------
-----------------
ПРАВИЛА:
     1 - Гравець грає з комп'ютером
     2 - За один раз можна заповнити від 1 до 5 клітинок
     3 - Той хто заповнює останню клітинку - програє
```

```
      ГАРНОЇ ГРИ!
------------------------------------------------------------
-----------------

Яку літеру обираєте X чи O?
O
  player буде першим
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     |
     |     |     |     |
Який наступний хід? (1-25)
1
Ви можете обрати ще 4 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
6
Ви можете обрати ще 3 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
11
Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати

Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
nm: 7 2
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
```

```
    |     |     |     |
 O  |     |     |     |
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |     |     |
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |     |     |
    |     |     |     |
Який наступний хід? (1-25)
3
Ви можете обрати ще 4 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
4
Ви можете обрати ще 3 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
5
Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати

Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
    |     |     |     |
    |     |     |     |
    |     |     |     |
-------------------------
    |     |     |     |
    |     |     |     |
    |     |     |     |
-------------------------
    |     |     |     |
 O  |     |     |     |
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |    X|  X  |  X
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |    O|  O  |  O
    |     |     |     |
Який наступний хід? (1-25)
12
Ви можете обрати ще 4 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
13
```

```
Ви можете обрати ще 3 клітинки (1-25). Нажміть ентер щоб
більше не вибирати

Ви можете обрати ще 3 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     |
     |     |     |     |
-------------------------
     |     |     |     |
 O   | O   |    O| X   |
     |     |     |     |
-------------------------
     |     |     |     |
 O   | X   |    X| X   | X
     |     |     |     |
-------------------------
     |     |     |     |
 O   | X   |    O| O   | O
     |     |     |     |
Який наступний хід? (1-25)
21
Ви можете обрати ще 4 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
22
Ви можете обрати ще 3 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
23
Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
24
Ви можете обрати ще 1 клітинку (1-25). Нажміть ентер щоб
більше не вибирати
25
nm: 20 15
nm: 25 15

     |     |     |     |
 O   |   O |    O| O   | O
     |     |     |     |
-------------------------
     |     |     |     |
     |     |     |     | X
     |     |     |     |
-------------------------
```

```
    |     |     |     |
 O  |  O  |    O|  X  |  X
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |    X|  X  |  X
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |    O|  O  |  O
    |     |     |     |
Який наступний хід? (1-25)
16
Ви можете обрати ще 4 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
17
Ви можете обрати ще 3 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
18
Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати

Ви можете обрати ще 2 клітинки (1-25). Нажміть ентер щоб
більше не вибирати
    |     |     |     |
 O  |  O  |    O|  O  |  O
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  O  |    O|  X  |  X
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  O  |    O|  X  |  X
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |    X|  X  |  X
    |     |     |     |
-------------------------
    |     |     |     |
 O  |  X  |    O|  O  |  O
    |     |     |     |
Ви виграли
Бажаєте зіграти ще раз?(yes чи no)
```

Приклад гри «Нім»:

```
Виберіть гру яку бажаєте зіграти:
          1 - "Так-Тікс" 5x5
          2 - "Так-Тікс" 4x4
          3 - "Нім"

3

------------------------------------------------------------
--------------
                    Гра "Нім"
------------------------------------------------------------
--------------
Правила:
          1 - Гравець грає проти комп'ютера
          2 - Одночасно з ігрового поля дозволяється забрати
від 1 до 3 монет
          3 - Програє той, хто робить останній хід
          ГАРНОЇ ГРИ!
------------------------------------------------------------
--------------


------------------------------------------------------------
--------------
          O   O   O   O   O   O   O   O   O   O   O   O   O
O   O   O   O
------------------------------------------------------------
--------------

Гра починається із17 монет.

Бажаєте грати першим чи другим? [F - якщо першим/S - другим]:
f

--ХІД ГРАВЦЯ--

Введіть свій вибір [1, 2 чи 3]: 2

------------------------------------------------------------
--------------
          O   O   O   O   O   O   O   O   O   O   O   O   O
O   O
------------------------------------------------------------
--------------
```

```
Залишилося 15 монет.

--ХІД КОМП'ЮТЕРА--

Комп'ютер робить хід ......
: 2 монету(и).

---------------------------------------------------------------
---------------
         O   O   O   O   O   O   O   O   O   O   O   O   O
---------------------------------------------------------------
---------------

Залишилося 13 монет.

--ХІД ГРАВЦЯ--

Введіть свій вибір [1, 2 чи 3]: 3

---------------------------------------------------------------
---------------
         O   O   O   O   O   O   O   O   O   O
---------------------------------------------------------------
---------------

Залишилося 10 монет.

--ХІД КОМП'ЮТЕРА--

Комп'ютер робить хід ......
: 1 монету(и).

---------------------------------------------------------------
---------------
         O   O   O   O   O   O   O   O   O
---------------------------------------------------------------
---------------

Залишилося 9 монет.

--ХІД ГРАВЦЯ--

Введіть свій вибір [1, 2 чи 3]: 2

---------------------------------------------------------------
---------------
         O   O   O   O   O   O   O
```

```
-------------------------------------------------------------
--------------

Залишилося 7 монет.

--ХІД КОМП'ЮТЕРА--

Комп'ютер робить хід ......
: 2 монету(и).

-------------------------------------------------------------
--------------
        O   O   O   O   O
-------------------------------------------------------------
--------------

Залишилося 5 монет.

--ХІД ГРАВЦЯ--

Введіть свій вибір [1, 2 чи 3]: 2

-------------------------------------------------------------
--------------
        O   O   O
-------------------------------------------------------------
--------------

Залишилося 3 монет.

--ХІД КОМП'ЮТЕРА--

Комп'ютер робить хід ......
: 2 монету(и).

-------------------------------------------------------------
--------------
         O
-------------------------------------------------------------
--------------

Залишилося 1 монет.

Кінець...
Залишилася тільки одна монета...
```

```
 ГРАВЕЦЬ програв, оскільки залишилася одна монета

Гру виграв...
: КОМП'ЮТЕР

Бажаєте зіграти ще раз?[Y/N]:
```

Висновки до розділу 3

В даному розділі було описано програмну реалізацію програми, яка дозволяє зіграти такі комбінаторні ігри як Так-Тікс та Нім. Були описані функції, які використовувалися для написання програми. Також були наведені приклади гри.

ВИСНОВКИ

В бакалаврській роботі було розглянуто алгоритмічну та програмну реалізацію комбінаторних ігор. Визначено основні ознаки, за якими можна віднести ту чи іншу гра саме до комбінаторних ігор:

- Існує два гравці.
- Гра має обмежену кількість позицій.
- Правила гри є однаковими для обох гравців.
- Гравці ходять по черзі.
- При досягненні позиції, з якої немає можливих ходів, гра буде завершена. Для нормальної гри виграє той, хто зробив останній хід, для мізерної гри той, хто зробив останній хід, програє.
- Після кінцевого числа ходів, незалежно від ходу самої гри, гра закінчується.
- Наявна повна інформація про гру.
- Немає випадкових ходів, роздачі карт та кидання гральних кубиків.

Були розглянуті алгоритми для комбінаторних ігор, а саме: дерево рішень, жадібний та симетричний алгоритм. Не для всіх ігор представлені алгоритми є дієвими, а також не завжди дієвий алгоритм приносить 100% перемогу. Іноді для перемоги варто будувати стратегію гри з урахуванням кількох алгоритмів.

Також розглянуті приклади комбінаторних ігор з використанням вищевказаних алгоритмів: Нім, Так-Тікс, Гекс та Клоббер.

Зроблена програмна реалізація ігор Так-Тікс та Нім мовою програмування Python3 з консольним інтерфейсом.

СПИСОК ЛІТЕРАТУРИ

1. Губко М.В., Новиков Д.А. Теория игр в управлении организационными системами. М.: ИПУ, 2005. – 138 с.
2. Деорнула П., Комбинаторная теория игр. — М.: МЦНМО, 2017. — 40 с. ISBN 978-5-4439-1172-4
3. Драч І.В., Зегельман М.М Модифікація комбінаторної гри Баше – гра з «особливим» ходом, 2018 р.
4. Кріль С.О., Зегельман М.М. Комбінаторна гра – «Зв'язна незв'язність». Серія: Фізико-математичні науки. Випуск 18. 2018 р.
5. Левитин А. В. Глава 10. Ограничения мощи алгоритмов: Деревья принятия решения // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 409—417. — 576 с. — ISBN 978-5-8459-0987-9
6. Мартин Гарднер, Математические головоломки и развлечения 2-е издание, Москва «Мир», 1999. 447с
7. Оцінка складності алгоритмів [Електронний ресурс] / Режим доступу: https://echo.lviv.ua/dev/53
8. Фролов И. С., Введение в теорию комбинаторных игр: учебное пособие, Матем. обр., 2012. 202 с.
9. Фролов И.С., Введение в теорию комбинаторных игр. Простейшие комбинаторные игры, Матем. обр., 2021, выпуск 3(63), 38-52
10. Чернійчук Г.П., Вєтров О.С. Алгоритмічна реалізація комбінаторних ігор. Матеріали всеукраїнської науково-практичної конференції для студентів, аспірантів, та молодих вчених (29 квітня 2020 р.). Вінниця, 2020. С. 188-189.
11. Чернійчук Г.П. Аналіз алгоритму та розробка програми для гри «Так-Тікс». Курсова робота. Вінниця, 2020. 36 с.
12. Aaron N. Siegle, Combinatorial game theory, 1977. 540 p.
13. Aumann R.J. Lectures on Game Theory. – San Francisco: Westview Press, 1989. – 120 с.

14. Big-O Algorithm Complexity Cheat Sheet. [Електронний ресурс] / Режим доступу: http://bigocheatsheet.com/
15. Charles L. Bouton Nim, A Game with a Complete Mathematical Theory Annals of Mathematics Second Series, Vol.3 №. ¼ (1901-1902), pp. 35-39 (5 pages)
16. Michael H. Albert, Richard J. Nowakowski, David Wolfe Lessons in play: an introduction to combinatorial game theory, 2007. 307 p.
17. Pu-yan Nie Game Theory and Applications in Economics / Puyan Nie, Takashi Matsuhisa, X. Henry Wang, Pei-ai Zhang //Journal of Applied Mathematics. Volume 2014 , Article ID 936192, 2 pages.
18. Richard K. Guy American Mathematical Society, Columbus, Ohio, 1990. 247 p.
19. Sylvia Nazar Mind games, 2016. 752 p.
20. https://en.wikipedia.org/wiki/Y_(game)
21. http://www.iggamecenter.com/info/ru/pex.html
22. http://www.iggamecenter.com/info/ru/atoll.html
23. http://www.iggamecenter.com/info/ru/nex.html
24. http://www.iggamecenter.com/info/ru/clobber.html