МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЧЕРНІЙЧУК ГАЛИНА ПЕТРІВНА

Допускається до захисту:
в. о. завідувача кафедри,
прикладної математики
_____ Вєтров О.С.
«____» _____2022 року

АНАЛІЗ АЛГОРИТМІВ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОМБІНАТОРНИХ ІГОР

Спеціальність 113 Прикладна математика

Кваліфікаційна (магістерська) робота
(відповідно до стандарту спеціальності та ОП)

Науковий керівник:
І. Г. Крикун, доцент кафедри
прикладної математики,
к. ф.-м. н., доцент
_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця 2022

АНОТАЦІЯ

Чернійчук Г. П. Аналіз алгоритмів та програмна реалізація комбінаторних ігор. Спеціальність 113 «Прикладна математика», освітня програма «Прикладна математика». Донецький Національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній роботі досліджено комбінаторні ігри, алгоритми та виграшні стратегії для комбінаторних ігор. Показано роботу досліджуваних алгоритмів та виграшних стратегій на прикладі комбінаторних ігор «Нім», «Гекс», «Так-Тікс», «Клоббер» та «Хакенбуш». Розроблена програма для ігор «Нім» та «Так-Тікс».

Ключові слова: комбінаторна гра, дерево рішень, жадібний алгоритм, симетричний алгоритм, виграшні стратегії.

65 с., 1 табл., 27 рис., 28 джерел.

ABSTRACT

Cherniychuk H. Algorithmic and software implementation of some combinatorial games. Specialty 113 “Applied mathematics”, programme “Applied mathematics”. Vasyl` Stus Donetsk National University, Vinnytsia, 2022.

The qualification work investigates combinatorial games, algorithms and winning strategies for combinatorial games. The work of the studied algorithms and winning strategies is shown on the example of combinatorial games nim, hex, tactix, clobber and hackenbush. Developed a program for games nim and tactix.

Keywords: combinatorial game, decision tree, greedy algorithm, symmetric algorithm, winning strategies.

Pages 65. Tabl. 1. Fig. 27. Bibliography: 28 items.

ЗМІСТ

ВСТУП ……5
РОЗДІЛ 1. КОМБІНАТОРНІ ІГРИ ТА ЇХ АЛГОРИТМИ ……9
1.1 Комбінаторні ігри ……..9
1.2 Класифікація комбінаторних ігор …… 10
1.3 Правила гри та ігрові скорочення ..……12
1.3.1 Різні ігрові значення ……..12
1.3.2 Визначення додавання ……..15
1.3.3 Визначення мінуса ……15
1.3.4 Визначення еквівалентності гри ……..15
1.3.5 Визначення нерівності ……..17
1.4 Алгоритми комбінаторних ігор ……..18
1.4.1 Дерево рішень ……19
1.4.2 Жадібний алгоритм ……...20
1.4.3 Симетричний алгоритм ……21
Висновки до розділу 1 ……21
РОЗДІЛ 2. АНАЛІЗ ДЕЯКИХ КОМБІНАТОРНИХ ІГОР ……..……22
2.1 Гра «Нім» ..……22
2.1.1 Функція Шпрага-Гранді ……..27
2.1.2 Теореми Баутона …..……..28
2.1.3 Методи побудови виграшних стратегій для Нім-подібних ігор ..28
2.2 Гра «Так-Тікс» ……..……30
2.2.1 Аналіз алгоритмів для гри «Так-Тікс» ……..33
2.2.2 Обчислення виграшних стратегій для гри «Так-Тікс» ……36
2.3 Гра «Гекс» ……36
2.4 Гра «Клоббер» ……41
2.5 Гра «Хакенбуш» ……..43
Висновки до розділу 2 ……45

РОЗДІЛ 3. СТВОРЕННЯ ПРОГРАМИ ДЛЯ КОМБІНАТОРНИХ ІГОР «НІМ» ТА «ТАК-ТІКС» …….……46

3.1 Аналіз програми та коду ……..46

3.2 Приклад програмної реалізації ……60

Висновки до розділу 3 ……....65

ВИСНОВКИ ……66

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…...……67

Вступ

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

Теорія ігор є самостійною дисципліною, яка використовується в суспільних науках [11]: біологія, інженерія, політика, інформатика, філософія; в прикладній математиці. Отримавши широке визнання як важливий інструмент у різних галузях, розвиток теорії ігор значно розширюється. Дана теорія може описувати ряд специфічних явищ: міжособистісні стосунки, конкуренція, війна та політика. У працях древніх філософів теорія ігор застосовувалася для розробки теорій етичної та нормативної поведінки.

Актуальність використання теорії ігор в сучасному житті досить велика. В економіці теорію ігор застосовують для аналізу економічної конкуренції, наприклад переговори, аукціони, голосування, визначення стратегій [21]. Теоретико-ігрові моделі – потенційний інструмент для аналізу рішень, що приймаються компаніями. Таким чином створені моделі змушують кожного гравця враховувати дії суперника при виборі стратегії.

У політичній сфері теорія ігор застосовується у міжнародній політиці, військовій стратегії, у голосуванні, теорії суспільного вибору тощо. Таким чином проводиться аналіз будь-якої конфліктної ситуації між окремими особами, державами, політичними партіями.

Цікавим напрямком застосування теорії ігор можна назвати – філософію. Дану теорії використовували як інструмент у філософських дискусіях.

Також теорія ігор є корисним і потенційним інструментом для людських відносин. Теорія викладена як частина загальної теорії раціональної поведінки.

Теорія прийняття рішень – це аналіз людської поведінки, який зосереджується на визначенні «найкращого» варіанту рішень для особи, яка приймає рішення [11]. Теорія рішень разом з теорією прийняття рішень аналізує взаємозалежні проблеми прийняття рішень між раціональними, стратегічними представниками (гравцями).

Теорія ігор – це формальне дослідження конфлікту та співпраці між розумними раціональними особами, які приймають рішення [11]. Узагальнивши можна сказати, що дана теорія досить потужний аналітичний інструмент, що допомагає зрозуміти явища, які виникають при взаємодії осіб, що приймають рішення. Успішне застосування в різних сферах життя, допомогло загострити інтуїцію, дає можливість раціонально моделювати різні ідеї, норми та цінності серед гравців.

Застосування комбінаторної теорії ігор до певної позиції полягає у визначенні оптимальної послідовності ходів для обох гравців аж до кінця гри, таким чином визначивши оптимальний хід у кожній позиції [3].

Розвиток математики дав поштовх до розвитку математичної теорії комбінаторних ігор. Початок вивчення комбінаторних ігор було покладено статтею англійського математика Ч. Баутона «Nim, A Game with a Complete Mathematical Theory» («Нім, гра з повною математичною теорією»), яка була опублікована 1902 року [12].

Актуальність теми дослідження. Вивчення комбінаторних ігор на сьогодні є актуальним питанням, оскільки ігри такого типу є моделлю поведінки двох сторін у конкурентній (в тому числі економічній, політичній та військовій) боротьбі і тому є запитаними в світі. Комбінаторні ігри є досить популярними з точки зору їх дослідження з використанням штучного інтелекту. Через складну структуру та велику кількість можливих ходів та станів, як правило складно напряму побудувати систему прийняття рішень, яка б охоплювала всі можливі комбінації та можливі дії гравців. Тому дослідження алгоритмів та виграшних стратегій для комбінаторних ігор є актуальним питанням, і має прикладне застосування для розробників комбінаторних ігор та дослідників різноманітних конкурентних процесів.

Об’єкт дослідження – комбінаторні ігри.

Предмет дослідження – математичні моделі, алгоритми та виграшні стратегії для комбінаторних ігор.

Мета дослідження – вивчити існуючі алгоритми та процес побудови виграшних стратегій для деяких комбінаторних ігор; вдосконалити наявні виграшні стратегії та втілити побудовані алгоритми досягнення виграшу за допомогою програмного забезпечення.

Завдання дослідження:

– проаналізувати наявні в літературі алгоритми для деяких комбінаторних ігор;

– дослідити існуючі виграшні стратегії для деяких комбінаторних ігор та вдосконалити їх;

– використати отримані виграшні стратегії за допомогою програмної реалізації комбінаторних ігор.

Наукова новизна: аналізуючи існуючі в літературі алгоритми та виграшні стратегії для деяких комбінаторних ігор, було знайдено можливості для покращення існуючих алгоритмів та побудовано оптимальні виграшні стратегії для деяких комбінаторних ігор.

Практичне значення отриманих результатів.

За допомогою отриманих в дослідженні покращених виграшних стратегій для деяких комбінаторних ігор, була розроблена програмна реалізація цих виграшних стратегій для деяких комбінаторних ігор. Також результати дослідження можуть бути використані при вивченні практичного застосування комбінаторних ігор у конкурентній боротьбі (в економіці, політиці та військовій сфері зокрема).

Апробація результатів дослідження.

Основні результати, отримані в магістерській роботі, були опубліковані у статті в закордонному фаховому виданні [13] та доповідались на міжнародній конференції, що проводилась за кордоном [14].

Структура магістерської роботи. Магістерська робота складається із вступу, трьох розділів, в першому з яких наводяться основні поняття з комбінаторної теорії ігор, в другому описуються комбінаторні ігри та приклади застосування алгоритмів та виграшних стратегій, в третьому описується

розроблена програма, зокрема використані функції та алгоритми, наводяться приклади програмної реалізації ігор, загальних висновків до дослідження та списку використаних джерел. Загальний розмір магістерської роботи – 69 сторінок, а розмір основної частини – 64 сторінок. Магістерська робота містить 27 ілюстрацій. Список використаних джерел складається з 28 джерел.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ ПО НАПРЯМКУ ДОСЛІДЖЕННЯ

1.1. Комбінаторні ігри.

Комбінаторні ігри – це ігри, в яких приймають участь два гравці без випадкових ходів та з повною інформацією [28]. 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 [28]:

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 грального кубиків.

«Нім», «Так-Тікс», «Клоббер», «Гекс» та «Хакенбуш» повністю підпадають під визначення комбінаторних ігор, оскільки повістю відповідають всім умовам комбінаторних ігор [28].

«Камінь-ножиці-папір» – немає випадкових ходів, але також немає повної інформації про хід суперника. Оскільки суперники не знають повної інформації про положення пальців. Окрім того, 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.

1.2. Класифікація комбінаторних ігор

Комбінаторна теорія ігор має 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 [8].

Неупереджена гра – це рівноправна комбінаторна гра з наступними властивостями [8]:

- 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: програє гравець, який зробив останній хід.

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

Комбінаторні ігри класифікують також за матеріальними ознаками [7]:

- 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.

З математичної точки зору дослідження комбінаторних ігор можна розділити на декілька категорій [16].

Перша категорія – аналіз та знаходження відповідних алгоритмів для вирішення гри алгебраїчним шляхом. До такої категорії можна віднести гру «Нім» – це гра, яка має чіткий зв'язок з математичною двійковою системою.

Друга категорія – графічні або інженерні методи знаходження алгоритмів для вирішення гри. Наприклад гра «Хакенбуш», в якій графічні знання можуть крок за кроком пояснити виграшні рішення.

Третя категорія – більш складна, оскільки вимагає знань в різних математичних концепцій та методів для розробки вірних стратегій для перемоги у грі.

1.3. Правила гри та ігрові скорочення

1.3.1 Формальне визначення гри.

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

Гра – це впорядкована частина наборів ігор, гра записується як {ходи лівої сторони | ходи правої сторони} або {варіанти лівого | варіанти правого} або $G = \{G_L \mid G_R\}$. Наприклад порожня множина ігор, тому $\{\mid\} = 0$ – гра яку виграв другий гравець. Дві множини ігор: { } та {0} тому можна сформувати ігри $\{0 \mid\} = 1$ – виграє лівий гравець, $\{\mid 0\} = -1$ – виграє правий [16].

Визначення 1.1. [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.

Визначення 1.2. [2] Для будь-якої позиції реалізується одна з наступних можливостей:

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

Гра – впорядкована пара наборів ігор [6]:

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

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

Основні поняття [15]:

$$G = \{G_L \mid G_R\} \quad (1.2)$$

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

$$-\mathrm{G} = \{-\mathrm{G_L} \mid -\mathrm{G_R}\} \quad (1.4)$$

$$\mathrm{G} = \mathrm{H}, \text{ якщо } \mathrm{G} - \mathrm{H} = 0 \quad (1.5)$$

$$G \geq H, (\forall X) \quad (1.6)$$

Еквівалентно, G = H якщо G + H має такий самий результат у найкращій грі, як і H + G для всіх ігор K

Якщо G = 0, то перший гравець, який зробив хід програє: 0 = { | }

Позитивне значення гри – Лівий переміг: 1 = { 0 | }

Негативне значення гри – Правий переміг: -1 = { | 0 }

З вищезазначеного випливає, що:

- $0 = \{ \mid \}$
- $1 = \{0 \mid \}, 2 = \{1 \mid \}, 3 = \{2 \mid \}$
- $-1 = \{ \mid 0\}, -2 = \{ \mid -1\}, -3 = \{ \mid -2\}$
- $n = \{n - 1 \mid \}$
- $* = \{0 \mid 0\}$
- $* = -*$
- $* n = \{* 0, * 1, \dots, * n - 1 \mid * 0, * 1, \dots, * n - 1\}$
- $\uparrow = \{0 \mid *\}$
- $\downarrow = -\uparrow$
- G > 0 - Лівий переміг
- G < 0 – Правий переміг
- G = 0 – Перший гравець програв
- G ↕ 0 – Другий гравець програв

Визначення 1.3. [10] 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 загальному випадку, якщо $G^L = G^R = \emptyset$, то день народження дорівнює 0

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. [10]. Кількість ігор, народжених до дня n, є скінченним. Будемо позначати ці кількість $g(n)$.

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

Теорема 1.1. [10] Існує 22 гри, народжені до другого дня.

Висновок: Якщо G має скінченний день народження, тоді G повинен мати скінченне 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.

1.3.2 Визначення додавання

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

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 [17]: $G + S = \{G + X\}_{X \in S}$

Теорема 1.2. $G + 0 = G$.

Доведення: G + 0 = { G^L | G^R } + { | } = { G^L + 0, G + ∅ | G^R + 0, G + ∅ }

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 є комутативним та асоціативним:

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

1.3.3 Визначення мінуса гри

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

Визначення мінуса гри полягає в тому, що ходи Лівого та Правого (Першого та Другого) гравців міняються місцями. Тобто зміна ролей двох гравців під час гри].

1.3.4 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 [16].

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 [10]**:** G – G = 0 .

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

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:

$$G = G + 0 \quad (1.11)$$

$$= G + (J - J) \quad (1.12)$$

$$= (G + J) - J \quad (1.13)$$

$$= (H + J) - J \quad (1.14)$$

$$= H + (J - J) \quad (1.15)$$

$$= H + 0 \quad (1.16)$$

$$= H \quad (1.17)$$

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.

1.3.5 Визначення нерівності

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

G ≤ H, (∀X) правий виграє G + X, коли правий виграє H + X [17].

Теорема 1.6. Наступні твердження є еквівалентними:

1) G ≥ 0
2) Лівий перемагає рухаючись другим на G
3) Для всіх ігор X, якщо лівий перемагає рухаючись другим на X, тоді лівий перемагає рухаючись на G + X
4) Для всіх ігор X, якщо лівий перемагає рухаючись першим на X, то лівий перемагає рухаючись першим на G + X

Доведення:

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

2 ⇒ 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 ⇒ 4 (з еквівалентності 2 випливає еквівалентність 4): Доводиться аналогічно попередньому пункту

3 ⇒ 2 (з еквівалентності 3 випливає еквівалентність 2): Вибрати X = 0

4 ⇒ 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 ≥ H тоді і лише тоді, коли , G + J ≥ 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)

1.4. Алгоритми комбінаторних ігор.

Найбільш поширеними алгоритмами для аналізу комбінаторних ігор є жадібний алгоритм, симетричний алгоритм та дерево рішень.

1.4.1 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].

Методи застосування дерева рішень дозволяє розглядати різноманітні сценарії розвитку подій, викликані впливом різних факторів розвитку.

Дерево рішень – це графічне зображення послідовності рішень і станів середовища з вказівкою відповідних ймовірностей і виграшів для будь-яких комбінацій, альтернатив і станів середовища. Складається із:

– «Стовбур» – основа, головне питання, на яке потрібно відповісти;
– «Гілки» – це стрілки з кількома варіантами відповідей;
– «Листя» – це ситуації до яких приведе обрана відповідь;

Визначення 1.4. Враховуючи гру G з початковою позицією S, дерево гри, пов'язане з (G, S), є напіввпорядкованим кореневим деревом, визначеним наступним чином [11]:

- 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.

Застосування алгоритму дерева рішень дозволяє:

- визначити шляхи досягнення мети з виконанням кількісної оцінки складності виникаючих завдань та оцінкою труднощів здійснення того чи іншого варіанту;

- поліпшувати якість рішень в умовах невизначеності.

Зображуючи графічно можливі рішення і їх результати, що в залежності від ступеня складності має різне число гілок. Рухаючись від вихідної точки уздовж гілок дерева і комбінуючи оцінки можна різними способами досягнути кінцевих точок. Таким чином можна оцінити варіанти шляху і обрати оптимальний з погляду результативності та ризику.

Переваги застосування алгоритму дерева рішень:

- одновимірна схема, яка наочно показує причинно-наслідкові зв'язки;
- є можливість оцінити різні шляхи і обрати найменш ризиковий.

Недоліки застосування дерева рішень:

- трудомісткий;
- можуть виникнути складні конструкції, які не дуже точно будуть відображати інформацію.

1.4.2 Жадібний алгоритм

Жадібний алгоритм є найпростішим з евристичних правил або алгоритмів [10]. Гравець, який використовує даний алгоритм забирає як можна більше фішок при кожному можливому ході. Гравець, який дотримується даного алгоритму, завжди вибирає хід який максимізує або мінімізує деяку величину пов'язану з ігровою позицією, після того, як цей хід зроблено.

Доцільно використовувати даний алгоритм при першій спробі зіграти в нову гру, особливо якщо суперник – експерт. Таким чином не буде витрачатися час на пошуки вдалих ходів. Варто зауважити, що даний алгоритм не завжди працює. Для таких ігор як «Так-Тікс» та «Нім» застосування даного алгоритму може призвести не до оптимального рішення, а навпаки видати найгірший з можливих розв'язків.

1.4.3 Симетричний алгоритм

Симетричний алгоритм [10] – інтуїтивно очевидна стратегія. Кожного разу коли суперник робить хід в одній частині дошки, гравець робить подібний хід в іншій частині дошки. Ключовим моментом є рішення про те, як дана імітація має відбутися. Для успішної гри не слід залишати для суперника відкритий хід, який дозволяє йому ліквідувати імітаційний хід.

Стратегія яка підтримує просту симетрію називається Tweedledum-Tweedledee або імітатор. Дана стратегія є ефективною для гри «Нім» з двома купками фішок. Якщо купки рівні, мають однакову кількість фішок, то краще надати можливість першого ходу супернику. Суперник забирає певну кількість фішок з купки, гравець забирає таку саму кількість фішок з іншої купки, залишаючи дві однакові купки. З іншого боку, якщо фішок в купках різна кількість, варто починати хід першим, для того, щоб зрівняти кількість фішок. Після цього застосувати стратегію Tweedledum-Tweedledee.

Висновки до розділу 1

В даному розділі було розглянуто основні поняття та загальні визначення комбінаторних ігор та математичної теорії комбінаторних ігор; наведено основні результати, отримані в загальній теорії комбінаторних ігор; також було розглянуто класифікація комбінаторних ігор та алгоритмів, за допомогою яких можна аналізувати комбінаторні ігри.

РОЗДІЛ 2
АНАЛІЗ ДЕЯКИХ 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. У класичному варіанті гри кількість купок дорівнює трьом. Окремий випадок, коли купка одна, але максимальне число предметів, які можна взяти за хід, обмежена, відома як гра Баше. В найбільш відомому варіанті гри 12 фішок розкладають в три ряди (рис. 2.1)

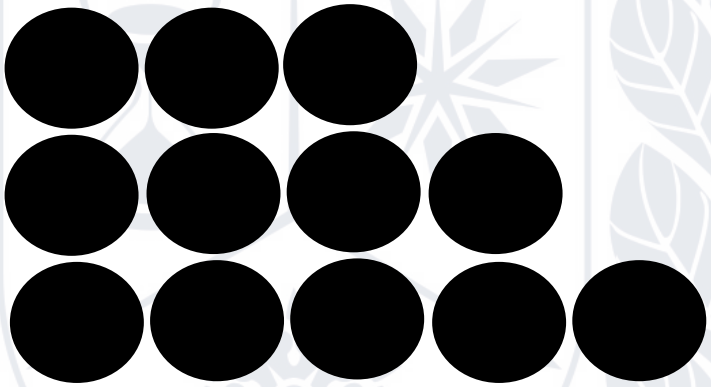


Рисунок 2.1 – Позиції фішок для гри по схемі 3-4-5

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 опублікував професор математики Гарвардського університету Чарльз Л. Баутон 1901 року [13]. Відкрита ним оптимальна стратегія заснована на двійковій системі числення. Позиції фішок Баутон визначив як небезпечну і безпечну.

Безпечна позиція – це позиція, яка утворилася після ходу гравця і гарантує перемогу.

Небезпечна позиція – це позиція, яка утворилася після ходу гравця і не гарантує перемогу.

Ч. Баутон 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 [10].

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.

Таким чином застосовуючи двійкову систему числення гравець може знаходити безпечні позиції та намагатися уникати небезпечних позицій.

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.

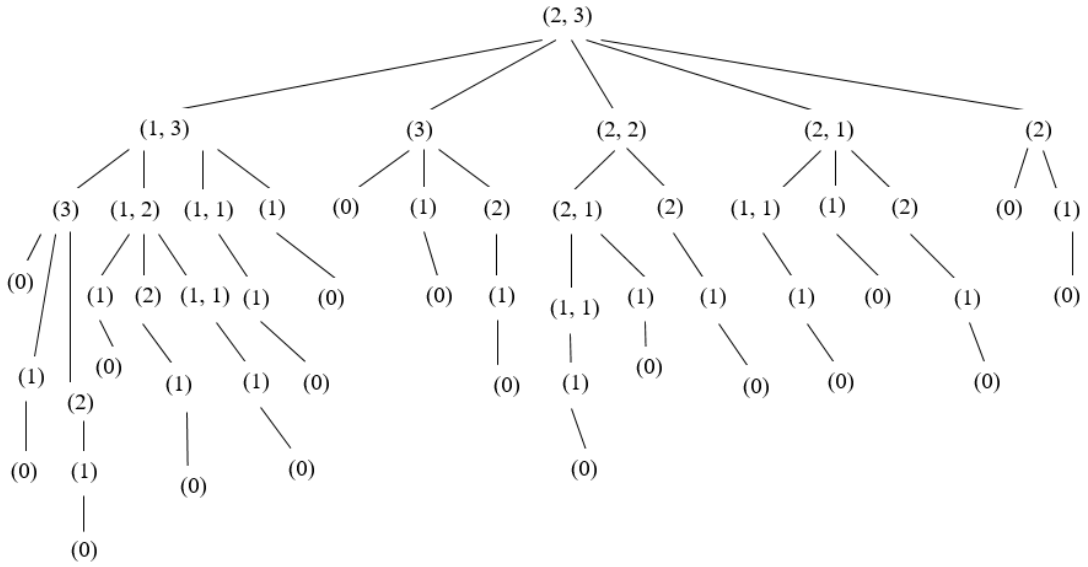


Рисунок 2.2 – Дерево рішень за схемою 2-3

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).

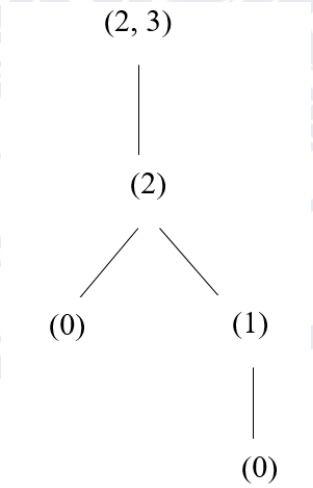


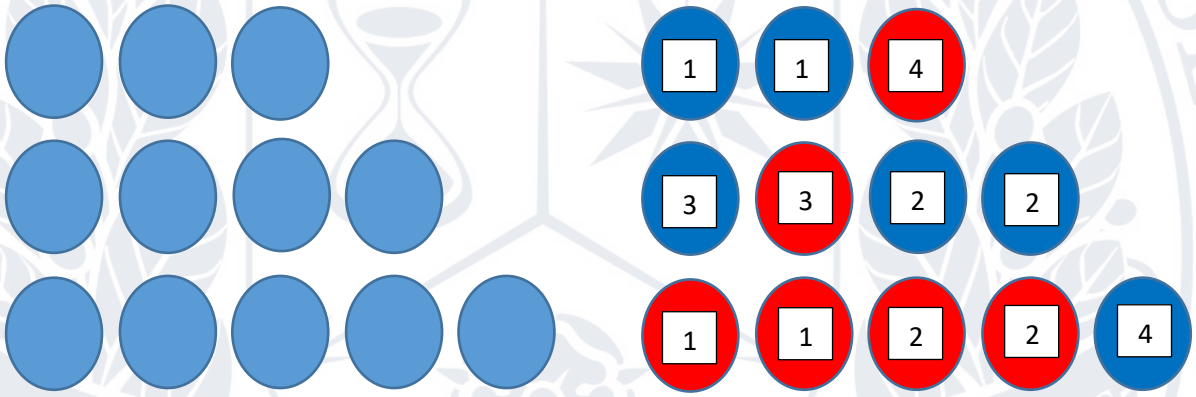
Рисунок 2.3 – Фрагмент дерева рішень для схеми 2-3

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» експонувався на виставці в Нью-Йорці, було зіграно 100 тис. партій, з яких машина виграла понад 90 тис.

Класична гра «Нім» має фундаментальне значення для теореми Шпрага-Гранді. Дана теорема стверджує, що звичайна гра в суму неупереджених ігор може прирівнюватися до гри в «Нім».

2.1.1 Функція Шпрага-Гранді

У комбінаторній теорії ігор функція Шпрага-Гранді стверджує, що кожна неупереджена гра за нормальних умов гри еквівалентна гри «Нім» з однією купою або нескінченному узагальненню «Нім» [18].

Для функції Шпрага-Гранді грою називається послідовна гра двох гравців з досконалою інформацією, яка задовольняє умову скінченності (всі ігри закінчуються; не існує нескінченних ходів) та умову нормальної гри (гравець, який не має змоги зробити хід, програє).

Позиції гравця у будь-який момент гри – це множина ходів, які дозволено зробити. Наприклад, нульова гра двох гравців – це гра де жоден з гравців немає дозволених ходів.

Одна з основних властивостей функції Шпрага-Гранді полягає в тому, що для всіх програшних позицій вона дорівнює нулю і позитивна для всіх виграшних позицій.

Функцією Шпрага-Гранді називається така функція G, певні для x і приймаюча невід'ємні значення, так що:

$$G(x) = min\{n \geq 0 : n \neq G(y), y \in F(x)\},$$

де:

n – будь-яке ціле невід'ємне число,

y – одна з позицій, в яку можна перейти безпосередньо з позиції x за один хід,

G(y) – значення Шпрага-Гранді позицій, в які можна перейти безпосередньо з позиції x за один хід,

F(x) – список позицій, в які можна перейти безпосередньо x позиції x за один хід.

Отже, G(x) – найменше невід'ємне ціле число, що знайдене серед значень Шпрага-Гранді для певних x. Функція F визначена на множині всіх позицій гри G наступним чином: F(P) = 0, якщо позиція P – програшна (гравець не може зробити жодного ходу), у випадку коли позиція P – виграшна:

$$F(P) = min\,(\Omega\{F(Q)|Q \in G(P)\}),$$

де Ω – безліч цілих невід'ємних чисел, G(P) – безліч допустимих ходів з позиції P.

2.1.2 Теореми Баутона

Основна мета Ч. Баутона в роботі [13] було довести, що якщо один гравець залишає певну комбінацію, і після цього грає без помилок, то суперник не має змоги виграти. Такі комбінації називаються безпечними. Вводяться наступні позначення:

*a, *b, …., *c – позначення купок з фішками;

*b` - позначення купки з фішками після здійснення ходу гравцем.

Перша теорема стверджує, що якщо Перший гравець залишає безпечну комбінацію, наприклад (*a; *b; *c), то Другий гравець не може перемістится в безпечну позицію за один хід. Наприклад, гравець за один хід змінює купку *c, *c ≠ *c`, а інші дві купки (*a, *b) не змінюються. Але варто зауважити, якщо (*a, *b, *c), безпечна позиція, то (*a, *b, *c`) не може бути також безпечною.

Теорема 2.1. (Перша теорема Ч. Баутона, узагальнена)

Якщо гравець залишає безпечну комбінацію (*a, *b, …, *c), то суперник не може перемістится в безпечну позицію за один хід.

Друга теорема стверджує , що якщо Другий гравець знаходиться в безпечній позиції, то незалежно від того як він рухається, Перший гравець може віднвити безпечну позицію своїм наступним ходом.

Теорема 2.2. (Друга теорема Ч. Баутона, узагальнена)

Якщо гравець опинився в безпечній позиції, то як би він не рухався, суперник своїм наступним ходом може повернути собі безпечну позицію.

2.1.3 Методи побудови виграшних стратегій для Нім-подібних ігор

Проаналізовані методи дослідження [16, 19, 20] висвітлюють загальні підходи до Нім-подібних ігор. В основу даних підходів покладені правила гри «Нім»: існує декілька рядів або купок з фішками; за один хід можна взяти будь-

яку кількість фішок з однієї купки. В «нормальній» грі виграє той, хто взяв останню фішку, в «мізер-грі» -програє той, хто взяв останню фішку.

Різні правила вилучення фішок з ігрового поля розбивають Нім-подібні ігри на кілька видів. Розглянемо наявні результати для Нім-подібних ігор, використовуючи [20].

Гра I. За кожен хід гравець може прибрати з ігрового поля від 1 до 3 фішок.

У даному типі гри, основна увага зосереджена на залишках чисел.

Теорема 2.3 Для гри I, гра є виграшною Другого гравця на кожному етапі, якщо число n фішок, що залишилися на ігровому полі на даному етапі кратне 4, і виграшною для Першого гравця в усіх інших випадках. Виграшна стратегія для Першого гравця полягає в тому, щоб вилучити з ігрового поля n mod 4 фішки (таку кількість фішок, яка з остачею ділиться на 4).

Гра II. Гравець може забрати будь-яку кількість фішок, меншу або більшу кількості фішок забраних суперником на попередньому ході. Перший гравець може прибрати будь-яку кількість фішок, окрім одразу всіх.

Аналіз виграшних стратегій для даного виду гри показує, що даний тип гри можна звести до аналізу двійкового розкладу чисел.

У даному різновиді Нім-подібних ігор, якщо гравець на певному етапі забирає 1 фішку, то при наступних ходах гравці можуть забрати лише по одній фішці. Оскільки за правилами «нормальної» гри, виграє той, хто забирає останню фішку, то гравець на етапі, коли кількість фішок, що залишилася на дошці стане непарною, може виграти гру, забравши 1 фішку.

Теорема 2.4 У грі II на етапі, коли на дошці залишається $2^p(2p+1)$ фішок, гра виграється першим гравцем тоді і лише тоді, коли правила гри дозволяють забрати на даному етапі 2^p фішок.

Приклад використання Теореми 2: партія за правилами Гри II, розпочата з 2^p фішок на дошці, завжди виграється другим гравцем, оскільки гравець, що розпочав гру, не може прибрати всі фішки. Партія за правилами Гри II

розпочата з іншою кількістю фішок, відмінною від 2^p, виграється гравцем, що розпочав гру.

Таким чином було визначено виграшні стратегії для Гри II. Однак можна отримати дані стратегії, використовуючи іншу ідею, яка може бути застосована до ігор з більш складними правилами.

Оскільки кількість кульок, як можна забрати з ігрового поля, постійно змінюється по ходу гри, визначається число $w(n)$, яке позначає найменшу кількість кульок, які потрібно забрати, щоб виграти на етапі, коли залишилося n фішок.

Теорема 2.5 Виграшна стратегія для гравця, який починає Гру II з n фішками на дошці, полягає у тому, щоб забрати таку кількість фішок, щоб та кількість фішок на полі m, яка стала після ходу першого гравця, у двійковому розкладі мала б нуль на місці крайньої правої одиниці у двійковому розкладі числа n.

Гра III. Гравець може забрати з ігрового поля кількість фішок, що менша або дорівнює подвоєній кількості фішок, забраних суперником за попередній хід. Перший гравець, може прибрати будь-яку кількість фішок, окрім одразу всіх.

Для даної гри важливу роль у пошуку виграшних стратегій відіграє послідовність чисел Фібоначчі.

Теорема 2.6 У грі III на етапі, коли кількість фішок, що залишилися на дошці, дорівнює n, гра виграється першим гравцем тоді і лише тоді, коли правила гри дозволяють забрати $l(n)$ фішок, де $l(n)$ - деяке число з послідовності Фібоначчі, що відповідає числу n (детальне пояснення в [20]).

2.2. Гра «Так-Тікс»

Гра «Так-Тікс» [6, 16, 21] 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. В книзі М. Гарднера [6] описана теорія для гри «Так-Тікс» та наведені деякі приклад гри, з поясненням для ходів гравців.

Існують виграшні стратегії для гравців («нормальна» гра):

- Для Першого гравця: якщо N непарне, потрібно взяти центральну фігуру і симетрично копіювати кожен хід суперника. У підсумку гравець забирає останню фішку, з ігрового поля, і виграє.
- Для Другого гравця: якщо N парне, потрібно симетрично копіювати ходи суперника. У підсумку гравець забирає останню фішку і виграє.

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

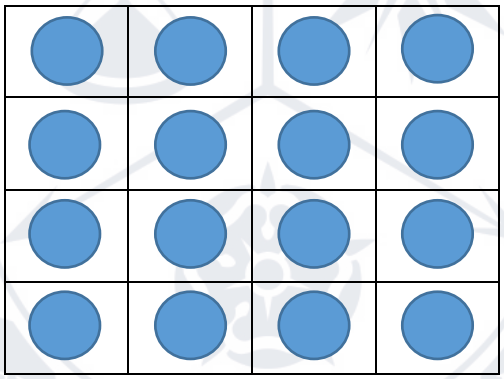


Рисунок 2.5 – Ігровий простір для гри «Так-Тікс»

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, 11-12].

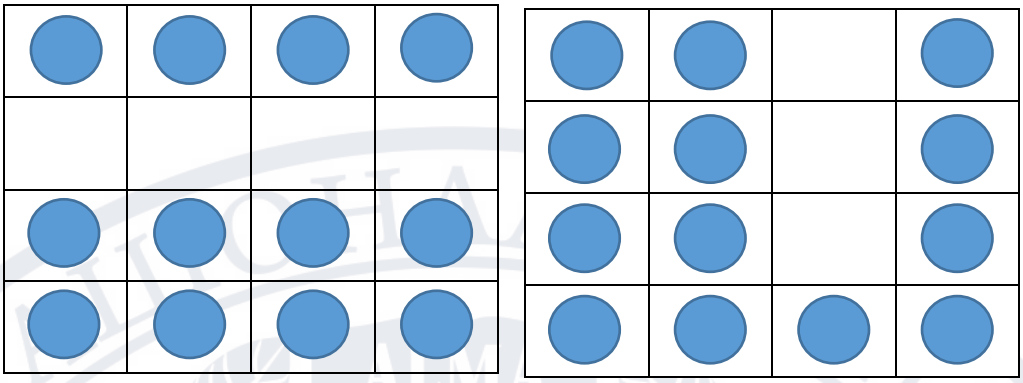


Рисунок 2.6 – Можливі ходи гравців в грі «Так-Тікс»

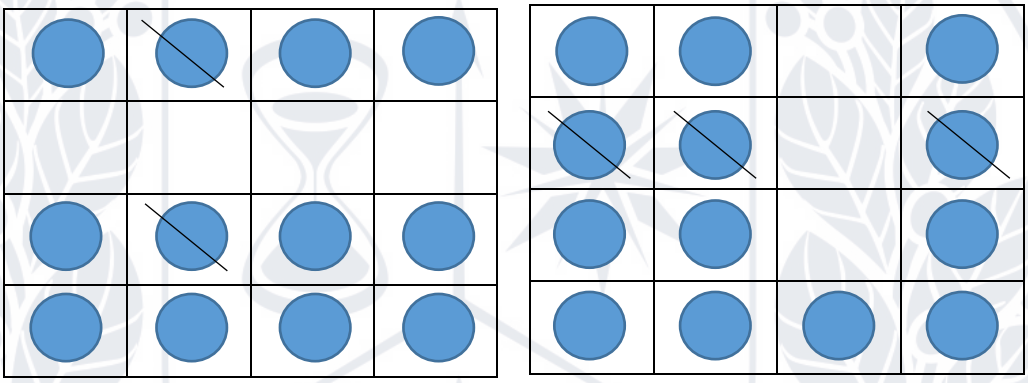


Рисунок 2.7 – Неможливі ходи для гравців в грі «Так-Тікс»

Гра «Так-Тікс» відноситься до неупереджених ігор, а також має властивість бути скінченною.

Неупереджена гра – це рівноправна комбінаторна гра з наступними властивостями: дія або хід, який доступний гравцеві залежить лише від стану гр, але не залежить від того, хто робить хід; гравець, який не може зробити хід – програє; гра завжди завершується. Умову завершення гри можна змінювати, для отримання гри у мізер – гравець, який зробив останній хід програє. Умова неупередженості робить не важливим, хто робить хід; кожна позиція має результат з урахуванням того, хто зробив хід [21].

Скінченна гра – це гра, яка починається з будь-якої позиції, а кінцевий результат завжди досягається після певної кількості ходів, яка залежить тільки

від початкової позиції [21]. Таким чином гра закінчується, коли фішок на ігровій дошці не залишилося. Кількість зіграних ходів дорівнює, як максимум кількості фішок на ігровому полі.

2.2.1 Аналіз алгоритмів

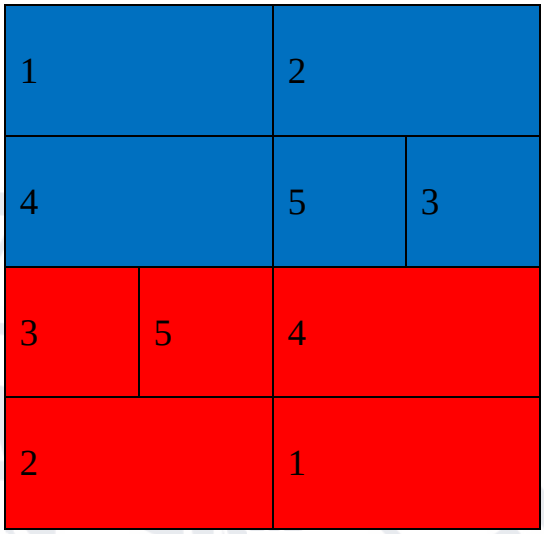
Проаналізуємо застосування симетричного алгоритму для гри. Даний алгоритм досить простий у реалізації, оскільки основна мета – віддзеркалення ходів суперника. Ходи першого гравця позначаються синім кольором, ходи другого гравця позначається червоним кольором. Другий гравець дотримується симетричного алгоритму відносно ходів першого гравця. Гравці грають у мізер-гру – програє той, хто робить останній хід. Для того, щоб виграти гру використовуючи даний алгоритм, потрібно не залишати супернику хід, який дозволить усунути можливість повторення ходу.

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)

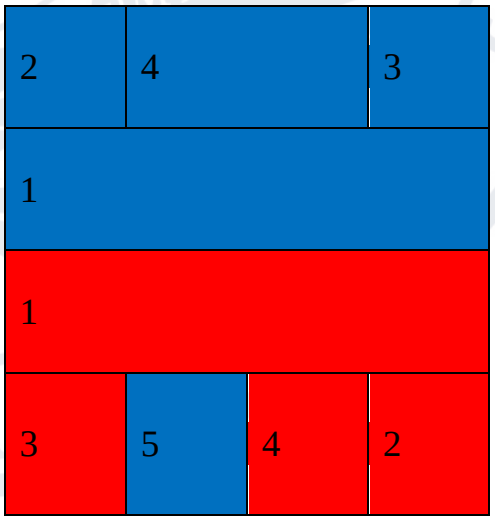
Рисунок 2.8 – Ігрове поле гри «Так-Тікс»: а) початкова ігрова дошка; б) ігрова дошка після гри (цифри – послідовність ходів гравців)

В даній таблиці наведено послідовність ходів гравців, другий гравець дотримувався симетричної стратегій, але на 4ому 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.

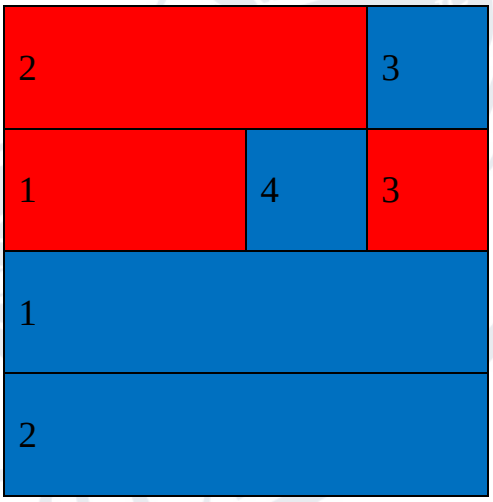
Жадібний алгоритм: за правилами максимальна кількість фішок, які можна вилучити з ігрового поля, рівна 4 по горизонталі або вертикалі. Перший гравець буде дотримуватися жадібного алгоритму, другий гравець не буде робити ходи за допомогою даного алгоритму.

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

2.2.2 Обчислення виграшних стратегій для гри Так-Тікс

Враховуючи, що гра «Так-Тікс» є різновидом гри Нім, аналіз наведених вище результатів дає підстави стверджувати, що вони залишаються справедливими і для «Так-Тікс». Однак, виграшні стратегії для «Так-Тікс» є значно простішими, ніж для «Нім». Тому було проведено аналіз різних типів гри (наведених у пункті 2.3.1), на основі якого було сформульовано виграшні стратегії для «Так-Тікс» та виконано програмну реалізацію отриманих стратегій.

Стандартна гра «Так-Тікс» проводиться на ігровому полі 4 х 4, заповненому фішками. Таким чином, гравець може забрати від 1 до 4 сусідніх фішок в одному рядку або стовпчику. Розглянемо варіацію правил - Гру I з попереднього пункту для гри «Так-Тікс».

Розглянемо, як визначаються виграшні стратегії в залежності від кількості фішок, що залишилися на ігровому полі. Етапи, в яких кількість фішок n = 1; 2; 3, є виграшними для першого гравця, оскільки гравець може забирати всі фішки, що залишилися на ігровому полі, один ряд або стовпчик за один раз, за умови, що ці фішки розташовані поруч одна з одною. Етап з n = 4 фішками, що залишилися, є виграшним для Другого гравця, оскільки будь-який хід Першого гравця призводить до виграшної ситуації для Другого гравця. На етапі n = 5 фішок, якщо перший гравець бере 2 або 3 фішки, то залишається 3 або 2 фішки і таким чином виграє суперник.

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 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.

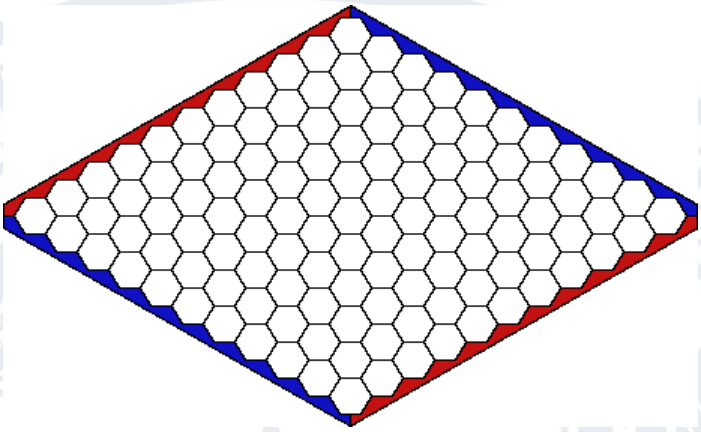


Рисунок 2.11 – Дошка для гри «Гекс», розмірності 11x11

Також популярними є дошки розмірності 14x14 та 19x19. Джон Неш вважав, що оптимальним розміром дошки є 14x14 [6]. 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.

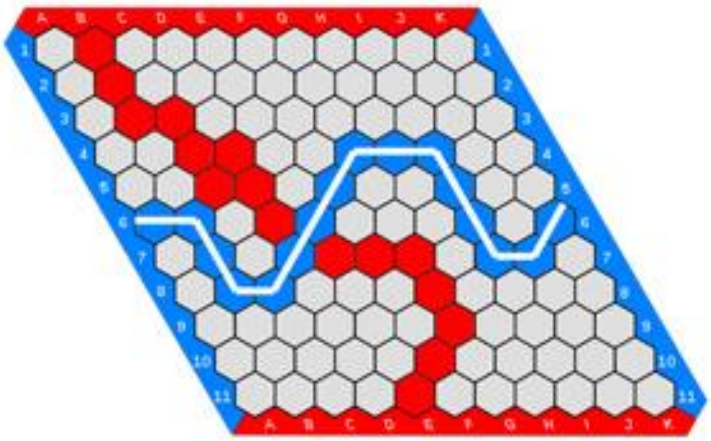


Рисунок 2.12 – Виграшна партія в «Гекс» для гравця, який грав синіми фішками

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 ігрову клітинку. 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.

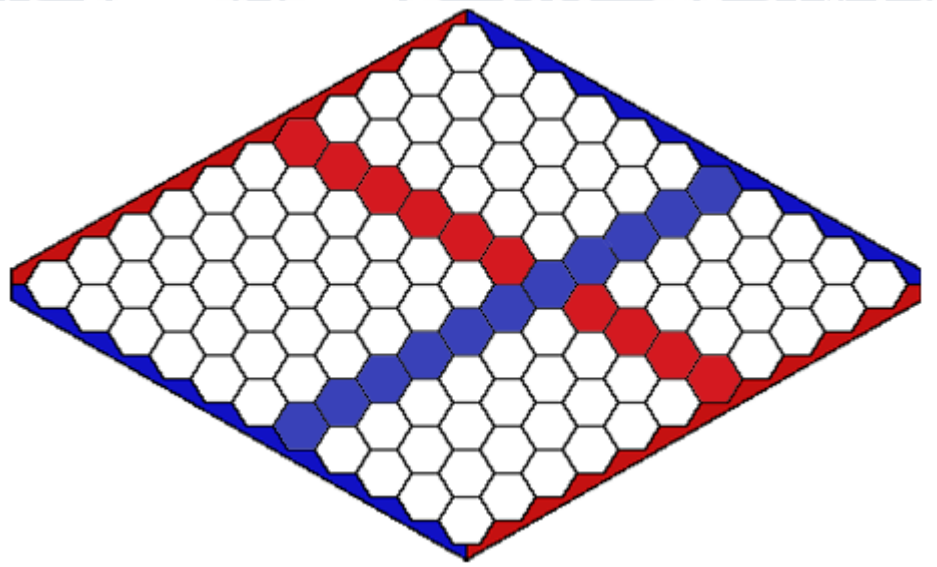


Рисунок 2.13 – Симетричний алгоритм для гри «Гекс»

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 [23].

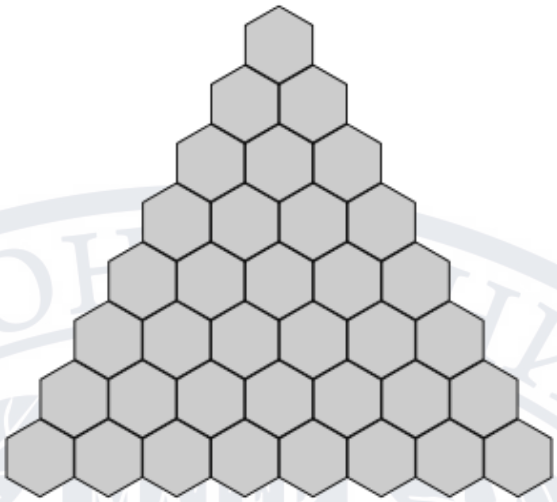


Рисунок 2.14 – Дошка для гри Y

– **Гавана:** винайдена К. Фрілінгом 1976 року, 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.

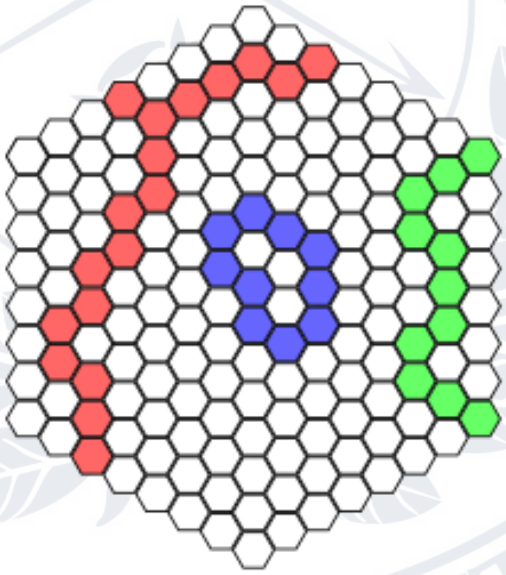


Рисунок 2.15 – Дошка для гри «Гавана» з виграшними позиціями

– 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 «Гекс» [24].

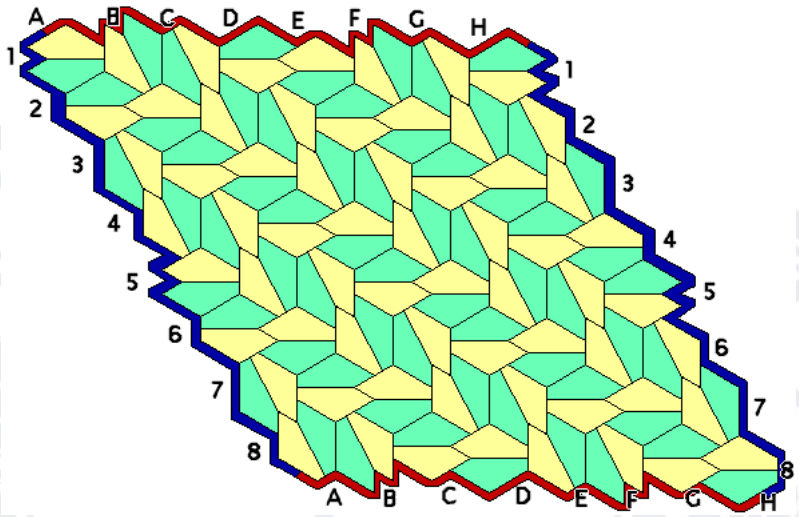


Рисунок 2.16 – Дошка для гри «Пекс»

– **Атолл:** 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 [25].

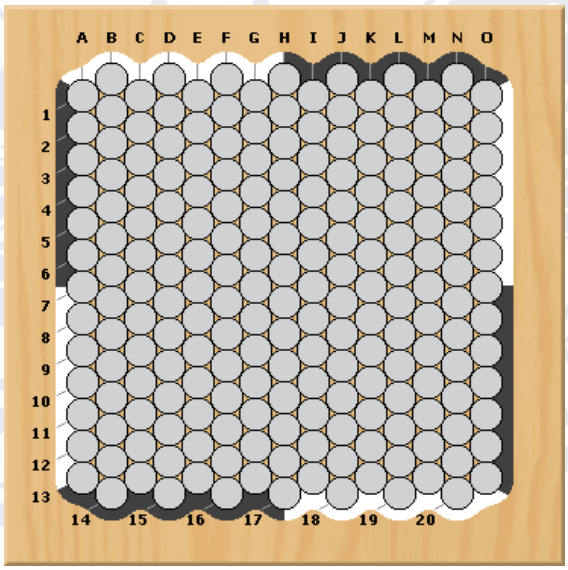
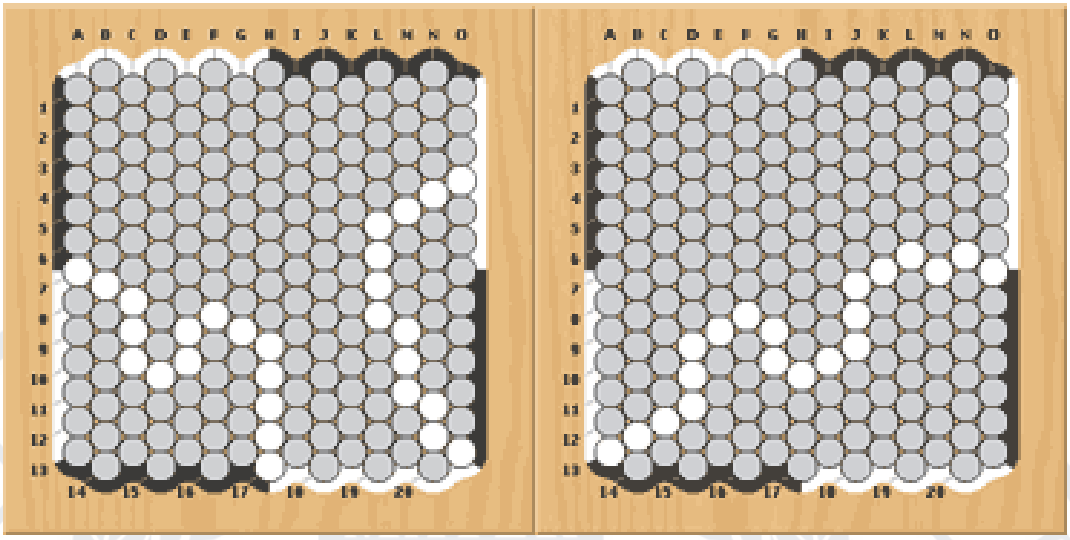


Рисунок 2.17 – Дошка для гри «Атолл»



а) б)

Рисунок 2.18 – Приклади виграшу у грі «Атолл»: 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 [26].

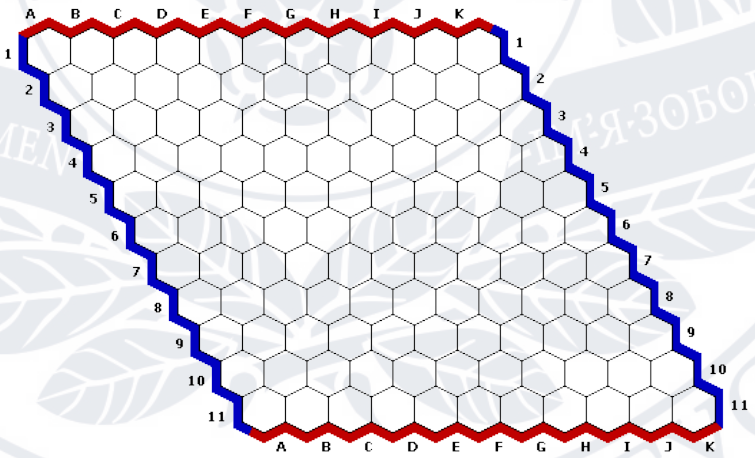


Рисунок 2.19 – Дошка для гри «Некс»

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 [27].

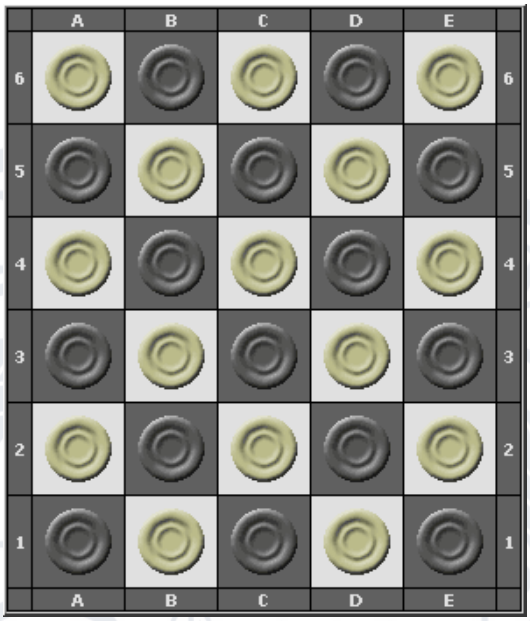


Рисунок 2.20 – Дошка для гри «Клоббер»

Це 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 клітинках. 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.

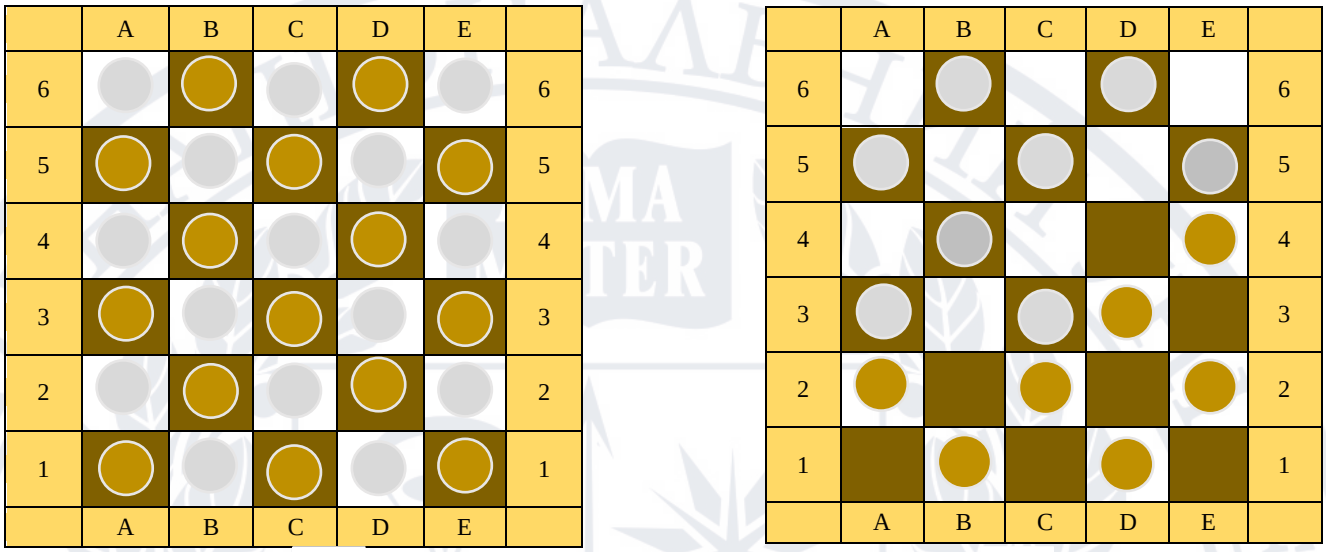
Проаналізувати гру «Клоббер» за допомогою дерева рішень складно, оскільки є велика кількість можливих комбінацій для ходів і дерево рішень є великим та складним.

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

Симетричний алгоритм є найдоцільнішим. В наведеному нижче прикладі 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 закінчення гри

2.5. Гра «Хакенбуш»

Гра «Хакенбуш» – це гра, в яку грають на конфігурації кольорових ліній (зазвичай червоних, синіх та зелених) [21]. Дані лінії з’єднуються із землею (великою горизонтальною лінією) або безпосередньо, торкаючись землі, або опосередковано, будучи з’єднаним з іншою лінією, яка з’єднана із землею. Гравці по черзі відтинають (стирають) відрізки. Наприклад, Перший гравець стирає червоні лінії, Другий гравець стирає сині лінії, обидва гравці можуть стерти зелені лінії. Коли лінія стерта, будь-які частини, що залишилися, які більше не з’єднані із землею, також видаляються. Перший гравець, який не може рухатися, програє, тобто виграє той, хто видаляє останню лінію.

Існує декілька варіантів гри Хакенбуш [9]:

• Оригінальний Хакенбуш: усі лінії мають однаковий колір і можуть бути відрізані будь-яким гравцем. Таким чином, можна зробити висновок, що

виграші є симетричними, і кожен гравець виконує однакові операції залежно від позиції на дошці (в даному випадку на малюнку);

• Синьо-Червоний Хакенбуш: відрізки пофарбовані у червоний або синій колір. Один гравець (зазвичай Перший або Лівий) відрізає лише сині лінії, інший гравець (Другий або Правий) відрізає лише червоні лінії;

• Синьо-Червоно-Зелений Хакенбуш: відрізки пофарбовані в червоний, синій та зелений кольори. Правила такі самі, як і для Синьо-Червоного Хакенбуша, з додотковим правилом, що будь-який гравець може відрізати зелені лінії.

Ігри типу «Хакенбуш» називаються упередженими (партизанськими/ партійними).

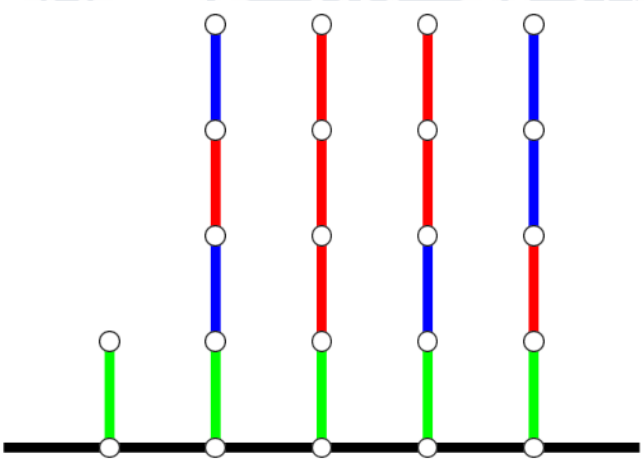


Рисунок 2.22 – Приклад гри «Хакенбуш»

Упереджені ігри – це ігри, в яких деякі ходи доступні одному гравцеві і не доступні іншому. Один гравець керує об’єктами одного класу, а інший – об’єктами іншого класу. Наприклад, гра в шахи – один гравець грає білими фігурами, а інший – чорними.

Гра «Хакенбуш» має чотири класи результатів (рис. 2.23):

1. Перший гравець може форсувати перемогу;

2. Другий гравець може форсувати перемогу;

3. Лівий гравець може форсувати перемогу, не важливо хто розпочав гру;

4. Правий гравець може форсувати перемогу, не важливо хто розпочав гру.

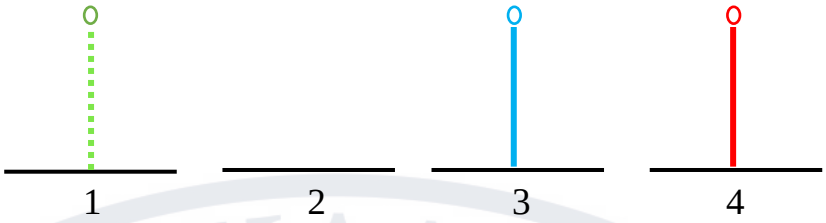


Рисунок 2.23 – Класи результатів гри «Хакенбуш»

Висновок до розділу 2

В даному розділі були розглянуті деякі з комбінаторних ігор, а саме «Нім», «Так-тікс», «Гекс», «Клоббер» та «Хакенбуш». Було розглянуто правила, особливості та варіації тієї чи іншої гри. Також було розглянуто застосування того чи іншого алгоритму для конкретних ігор.

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

3.1. Аналіз програми та коду

Код програми написаний на мові програмування Python. Програма має консольний інтерфейс. Гра ведеться проти комп'ютера. В програмі представлено наступні ігри: класична версія гри «Так-Тікс», гра «Так-Тікс» за правилами І гри Нім-подібних ігор, а також мізер-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" 4x4 (класична)
          2 - "Так-Тікс" 4x4 (модифікація)
          3 - "Нім" (мізер-гра)
""")
chosen_game = None
error_text = ""
```

```
while chosen_game != '1' and chosen_game != '2' and
chosen_game != '3':
    chosen_game = input(error_text)
    error_text = "Неправильний 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 рядок з *n* монет. Під час гри ігрове поле буде оновлюватися відповідно до зроблених ходів.

Для гри «Так-Тікс» поле представлене у вигляді списку рядків. В кожному рядку по чотири комірки, де будуть розташовуватися мітки гри, в кожної комірки є відповідні індекси. Щоб зробити хід, гравцеві потрібно з клавіатури ввести відповідний індекс комірки.

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[13]  + '  | ' + board[14]   + '
|   ' + board[15]  + '| ' + board[16])
        print ('    |     |     |     |')
        print ('-------------------------')
        print ('    |     |     |     |')
        print (' '  + board[9]  + '  | ' + board[10]   + '  |
' + board[11]  + '| ' + board[12])
        print ('    |     |     |     |')
        print ('-------------------------')
        print ('    |     |     |     |')
        print (' '  + board[5]  + '  | ' + board[6]   + '  |
' + board[7]  + '| ' + board[8])
        print ('    |     |     |     |')
        print ('-------------------------')
        print ('    |     |     |     |')
        print (' '  + board[1]  + '  | ' + board[2]   + '  |
' + board[3]  + '| ' + board[4])
        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 ігрового 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 whostartsfirst().

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). За умови, що гравці грають у мізер-гру.

Використана функція isWinner перевіряє, хто останній робив хід, та робить висновок про перемогу або поразку.

Якщо гравці грають у «нормальну» гру – перемагає той, хто зробив останній хід, функція перевіряє хто зробив останній хід і повідомляє про перемогу.

На даному етапі гра є завершеною і виникає питання чи буд гравець грати ще раз. Нічиєї 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. Нічиєї 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» оновлення 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. Якщо гравець введе індекс вже заповненої клітинки, або індекси клітинок, які не розташовані поруч, з’явиться попередження для гравця.

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) // 4 == (moves[0]-1) // 4 for m
in moves ) or all( m % 4 == moves[0] % 4 for m in moves)
        move = ' '
        possible_moves ='1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16' .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? (1-16)')
            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
вибранними')
                text_cells = 'клітинки'
                if 4 - move_index == 1:
                    text_cells = 'клітинку'
                print ('Ви можете обрати ще ' + str(4 -
move_index) + ' ' + text_cells + ' (1-16). Натисніть Enter
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 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 = [' ']*17
        playerLetter, computerLetter = inputPlayerLetter()
        turn = whoGoesFirst()
```

```
        print('   ' + turn + ' буде першим')
        gameIsPlaying = True
        while gameIsPlaying:
            if turn == 'player':
                drawBoard(theBoard)
                prev_moves = []
                for i in range(4):
                    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('Вітаю! Ви виграли!')
                        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, 3)
                for i in range(amount):
                    if column_or_row == 'row':
                        new_move = move + i+1
                        if (new_move-1) // 4 != (move-1) //
4:
                            break
                    else:
                        new_move = move + 4*(i+1)
                        print('nm: ' + str(new_move) + ' ' +
str(move))
                        if new_move < 0 or new_move > 16:
                            break
                    if isSpaceFree(theBoard, new_move):
                        makeMove(theBoard, computerLetter,
new_move)
                    else:
                        break

                if isBoardFull(theBoard):
                    drawBoard(theBoard)
                    print('Нажаль Ви програли:(')
```

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

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

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


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

        print("\nКінець...\nЗалишилася тільки одна
монета...")
        game_winner, game_loser = gameover(НімBoard,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 класичну гру «Так-Тікс».

Приклад гри «Так-Тікс»:

```
Виберіть гру яку бажаєте зіграти:
          1 - "Так-Тікс" 4x4 (класична)
          2 - "Так-Тікс" 4x4 (модифікація)
          3 - "Нім" (мізер-гра)

1

------------------------------------------------------------
                        ГРА "ТАК-ТІКС"
------------------------------------------------------------
ПРАВИЛА:
      1 - Гравець грає з комп'ютером
      2 - За один раз гравець може заповнити від 1 до 4
клітинок
      3 - Виграє той, хто робить останній хід
      ГАРНОЇ ГРИ!
------------------------------------------------------------

Яку літеру обираєте X чи O?
O
  computer буде першим
nm: 5 1
     |     |     |
     |     |     |
     |     |     |
--------------------
     |     |     |
     |     |     |
     |     |     |
```

```
---------------------
     |     |     |
 X   |     |     |
     |     |     |
---------------------
     |     |     |
 X   |     |     |
     |     |     |
Який наступний хід? (1-16)
9
Ви можете обрати ще 3 клітинки (1-16). Натисніть Enter щоб
більше не вибирати
10
Ви можете обрати ще 2 клітинки (1-16). Натисніть Enter  щоб
більше не вибирати
11
Ви можете обрати ще 1 клітинку (1-16). Натисніть Enter  щоб
більше не вибирати
     |     |     |
     |     |     |
     |     |     |
---------------------
     |     |     |
 O   | O   |    O|
     |     |     |
---------------------
     |     |     |
 X   |     |     |
     |     |     |
---------------------
     |     |     |
 X   | X   |    X| X
     |     |     |
Який наступний хід? (1-16)
15
Ви можете обрати ще 3 клітинки (1-16). Натисніть Enter  щоб
більше не вибирати
16
Ви можете обрати ще 2 клітинки (1-16). Натисніть Enter  щоб
більше не вибирати
nm: 10 6
     |     |     |
     |     |    O| O
     |     |     |
---------------------
     |     |     |
 O   | O   |    O|
     |     |     |
```

```
--------------------
    |     |     |
 X  | X   |     |
    |     |     |
--------------------
    |     |     |
 X  | X   |   X | X
    |     |     |
Який наступний хід? (1-16)
7
Ви можете обрати ще 3 клітинки (1-16). Натисніть Enter  щоб
більше не вибирати
8
Ви можете обрати ще 2 клітинки (1-16). Натисніть Enter  щоб
більше не вибирати
nm: 16 12
    |     |     |
    |     |   O | O
    |     |     |
--------------------
    |     |     |
 O  | O   |   O | X
    |     |     |
--------------------
    |     |     |
 X  | X   |   O | O
    |     |     |
--------------------
    |     |     |
 X  | X   |   X | X
    |     |     |
Який наступний хід? (1-16)
13
Ви можете обрати ще 3 клітинки (1-16). Натисніть Enter  щоб
більше не вибирати
14
    |     |     |
 O  |  O  |   O | O
    |     |     |
--------------------
    |     |     |
 O  | O   |   O | X
    |     |     |
--------------------
    |     |     |
 X  | X   |   O | O
    |     |     |
--------------------
```

```
    |     |     |
 X  |  X  |     X|  X
    |     |     |
Вітаю! Ви виграли!
Хочете 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
-----------------------------------------------------------


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 магістерській роботі було розглянуто алгоритмічну та програмну реалізацію комбінаторних ігор. Наведено приклади застосування теорії ігор у різних галузях життя. Описано визначення комбінаторних ігор, їх властивості. 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 скінченного 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.

Були проаналізовані методи побудови виграшних стратегій для Нім-подібних ігор. Ігри поділені на три типи в залежності від способу взяття фішок, основна увага зосереджена на залишкових числах, двійковому розкладі та числах Фібоначчі. Побудовані виграшні стратегій для гри «Так-Тікс».

Також розглянуті приклади конкретних комбінаторних ігор з використанням вищезазначених алгоритмів: «Нім», «Так-Тікс», «Гекс», «Клоббер» та «Хакенбуш».

Розроблена програмна реалізація ігор «Так-Тікс» та «Нім» мовою програмування Python з консольним інтерфейсом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Губко М. В., Новиков Д. А. Теория игр в управлении организационными системами. Москау: ИПУ, 2005. 138 с.
2. Деорнуа П. Комбинаторная теория игр. Москау: МЦНМО, 2017. 40 с.
3. Драч І. В., Зегельман М. М. Модифікація комбінаторної гри Баше – гра з «особливим» ходом. *Наука и абразование*: сборник трудов XII Междунар. науч. конференции, 1–9 июля 2018 г., Осло, Норвегия – Хмельницкий : ХНУ, 2018. С. 117-123.
4. Кріль С. О., Зегельман М. М. Комбінаторна гра – «Зв'язна незв'язність». *Математичне та комп'ютерне моделювання. Серія: Фізико-математичні науки*: збірник наукових праць. Кам'янець-Подільський: Кам'янець-Подільський Національний університет, 2018. Вип. 18. С. 100-105.
5. Левитин А. В. Глава 10. Ограничения мощи алгоритмов: Деревья принятия решения. *Алгоритмы. Введение в разработку и анализ*. Москау: Вильямс, 2006. С. 409-417.
6. Гарднер М. Математические головоломки и развлечения, 2-е издание. Москау: Мир, 1999. 447 с.
7. Фролов И. С. Введение в теорию комбинаторных игр: учебное пособие, Москау: Феникс, 2012. 202 с.
8. Фролов И. С. Введение в теорию комбинаторных игр. Простейшие комбинаторные игры. *Математическое образование*. 2012. № 3(63). С. 38-52.
9. Alabdala A., El-Seidy E. A group from Hackenbush game. *Journal of the Egyptian Mathematical Society*. Vol. 27, № 13, 2019.
10. Albert M. H., Nowakowski R. J., Wolfe D. Lessons in play: an introduction to combinatorial game theory, 2nd ed. Boca Raton: CRC Press, 2019. 344 p.
11. Aumann R. J. Lectures on Game Theory. San Francisco: Westview Press, 1989. 120 p.

12. Bhuiyan B. A. An overview of game theory and some applications. *Philosophy and Progress*. 2016. Vols. LIX-LX. P. 112-128.

13. Bouton C. L. Nim, a Game with a Complete Mathematical Theory. *The Annals of Mathematics*, 1901-1902, 2nd Ser., Vol. 3, № 1/4. P. 35-39.

14. Cherniichuk H. P., Krykun I. H. Notes about Winning Strategies for Some Combinatorial Games. *Journal of Mathematics Letters*. 2022. Vol. 1, № 1. P. 1–9.

15. Cherniichuk H. P., Krykun I. H. New algorithms and their software implementations for combinatorial games. *The 29th Conference on Applied and Industrial Mathematics dedicated to the memory of Academician Mitrofan M. Choban*: abstracts of conference, 25-28 August 2022, Chisinau, Moldova. P. 172-173.

16. El-Seidy E., Hussein S. E. S., Alabdala A. T. Models of Combinatorial Games and Some Applications: A Survey. *Journal of Game Theory*. 2016. Vol. 5(2). P. 27-41.

17. Siegel A. N. Combinatorial Game Theory. Providence: American Mathematical Society. 2013. 523 p.

18. Nivasch G. The Sprague–Grundy function of the game Euclid. *Discrete Mathematics*. 2006. Vol. 306. P. 2798–2800.

19. Oltean M. Evolving Winning Strategies for Nim-like Games, In *IFIP Student Forum*. 2004. P. 353–364.

20. Ooya T., Akiyama J. Impact of binary and Fibonacci expansions of numbers on winning strategies for Nim-like games. *International Journal of Mathematical Education in Science and Technology*. 2003. Vol. 34, № 1. P. 121-128.

21. Iacono J., Mazur K. Tactix on an S-shaped Board. *22nd Annual Fall Workshop on Computational Geometry*: proceedings of workshop, College Park MD, November 9-10, 2012. P. 17-18.

22. Nie P. Y., Matsuhisa T., Wang X. H., Zhang P. A. Game theory and applications in economics. *Journal of Applied Mathematics*. 2014. Vol. 2014, P. 1–2.

23. Y (game) – Wikipedia. URL:http://en.wikipedia.org/wiki/Y_ (game)

24. Пекс : idGameCenter URL: https://www.iggamecenter.com/ru/rules/pex

25. Атолл: idGameCenter URL: https://www.iggamecenter.com/ru/rules/atoll

26. Некс : idGameCenter URL: https://www.iggamecenter.com/ru/rules/nex

27. Клоббер : idGameCenter. URL: https://www.iggamecenter.com/ru/rules/clobber

28. Guy R. K. *Combinatorial games*. Columbus: American Mathematical Society Short Course Lecture Notes. 1990. 247 p.

Чернійчук Галина Петрівна

Прізвище, ім’я, по батькові

Інформаційних і прикладних технологій

Факультет

113 Прикладна математика

Шифр та назва спеціальності

Прикладна математика

Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему: «Аналіз алгоритмів та програмна реалізація комбінаторних ігор» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21—25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

(дата) (підпис здобувача освіти)