Лабораторна робота №13.

Тема. Основи проектування та створення абстрактних типів даних для реалізації яких використовуються структури даних дерева.
Мета. Засвоїти послідовність дій по проектуванню та реалізації абстрактних типів даних на основі структур даних дерева.

Короткі теоретичні відомості.

Бінарне дерево, як і інші дерева можна представити (реалізувати) на основі зв’язаної структури. Варіантів реалізації є достатньо багато. Якщо загальні принципи використання зв’язаної структури є спільними чи надзвичайно подібними (порівняйте рис.1. та рис.2.) то вже конкретні реалізації досить сильно відрізняються (див. відповідні модулі в пакунках Trees та Trees_easy та порівняйте класи LinkedBinaryTree).

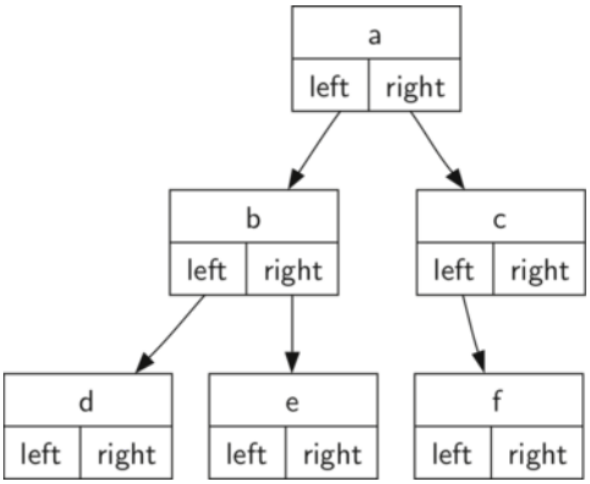


Рис.1.

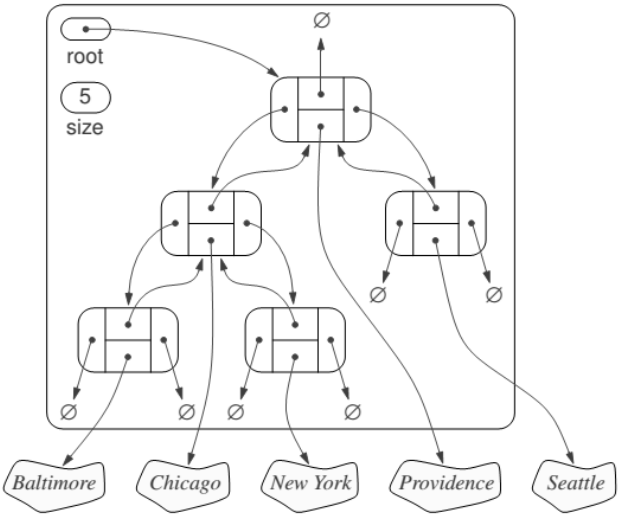


Рис2.

Методи в цих класах також відрізняються (див. Таблицю), хоча основні методи для роботи з деревом співпадають.

BinaryTree	LinkedBinaryTree (без врахування успадкованих)
`insert_left`	`_add_left`
`insert_right`	`_add_right`
`get_right_child`	`right`
`get_left_child`	`left`
`set_root_val`	`_add_root`
`get_root_val`	`root`
	`parent`
	`num_children`
	`_replace`
	`_delete`
	`_attach`
	`__len__`

Бінарні дерева широко застосовуються для вирішення різноманітних завдань. Наприклад, бінарне дерево дозволяє розробити програму, яка допоможе виграти в гру хрестики – нулики.

Для створення такої програми потрібно скористатися однією з реалізацій бінарного дерева, яку доведеться адаптувати або написати свій варіант реалізації бінарного дерева. Бінарне дерево буде використовуватися для побудови дерева гри і дозволить програмі знаходити найкращий поточний хід.

Розроблення програми передбачає реалізацію (адаптацію) структури даних бінарне дерево (клас BTree) для збереження об’єктів. Клас BTree повинен використовувати клас BTNode для представлення вершин (вузлів) бінарного дерева. Ще одним об’єктом для вирішення поставленого завдання буде клас Board, який буде використовуватися для збереження ігрового поля та іншої важливої інформації (див. рис.3.).

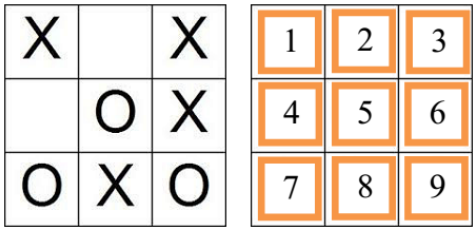
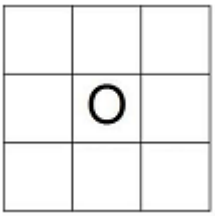


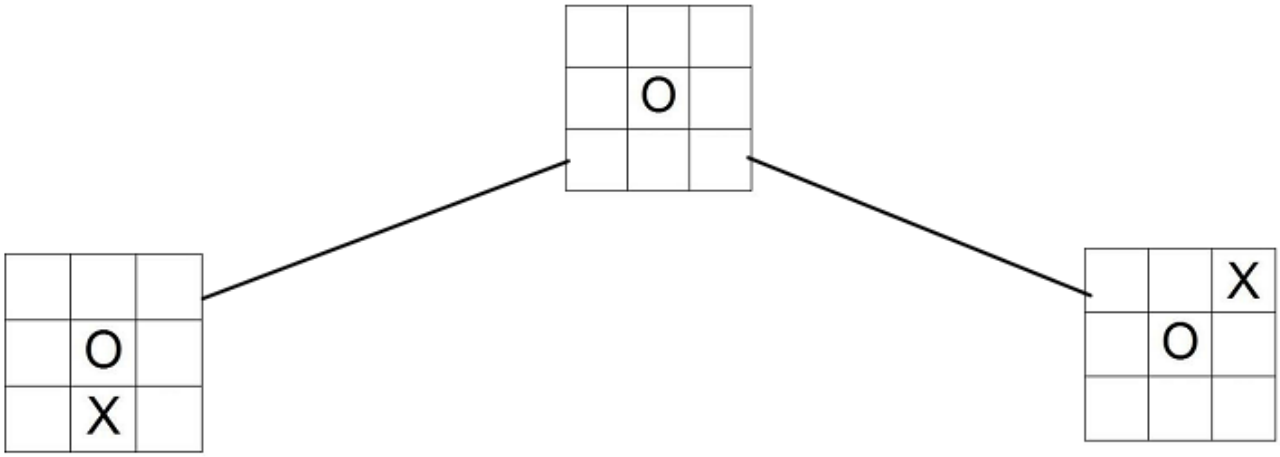
Рис.3.

Гра починається з пустого ігрового поля і один з гравців (наприклад, людина - користувач) робить перший хід. Перший хід це запис “0” в одну з комірок ігрового поля (комірки 1-9), наприклад в комірку 5. Наступний хід повинен зробити комп’ютер.

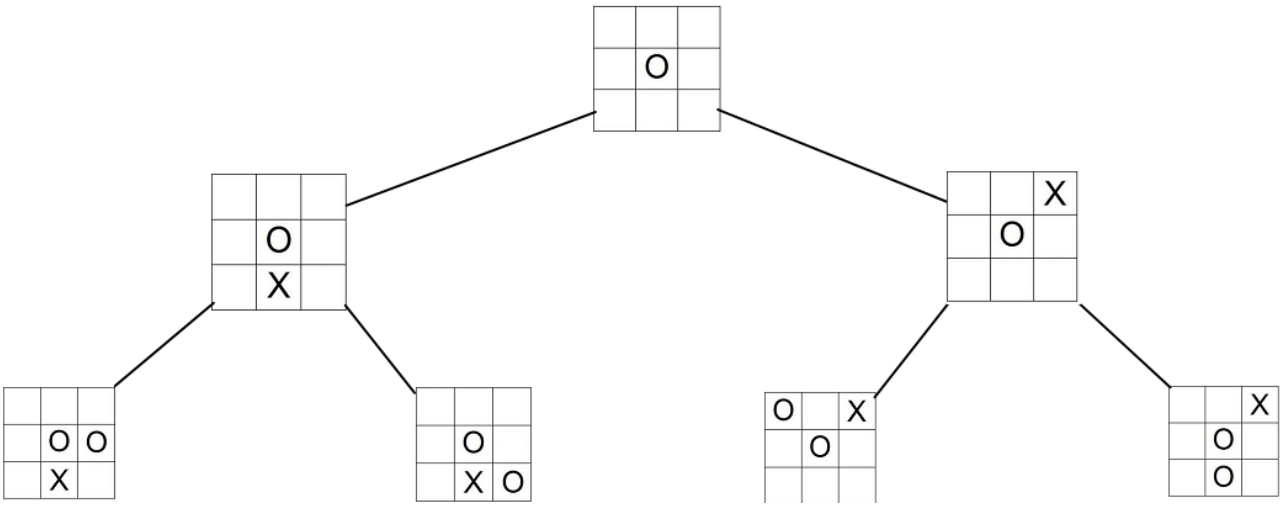
Комп’ютер повинен з використанням бінарного дерева вирішити проблему прийняття рішення про наступний хід. Вирішення проблеми прийняття рішення починається (ґрунтується) на поточному стані ігрового поля. Якщо комп’ютер повинен зробити другий хід то пошук рішення починається з поточного стану ігрового поля, який відображається на нульовому рівні дерева (level 0)



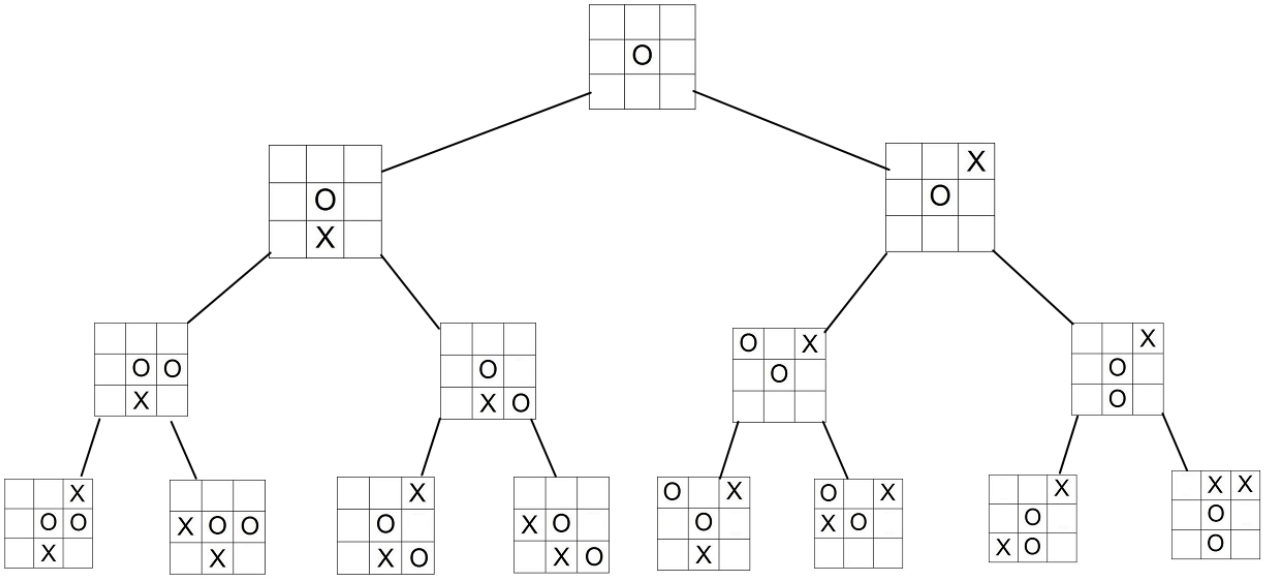
і комп’ютер випадковим чином вибирає дві комірки для запису “X” і вже два нові стани ігрового поля будуть відображені на наступному рівні бінарного дерева:



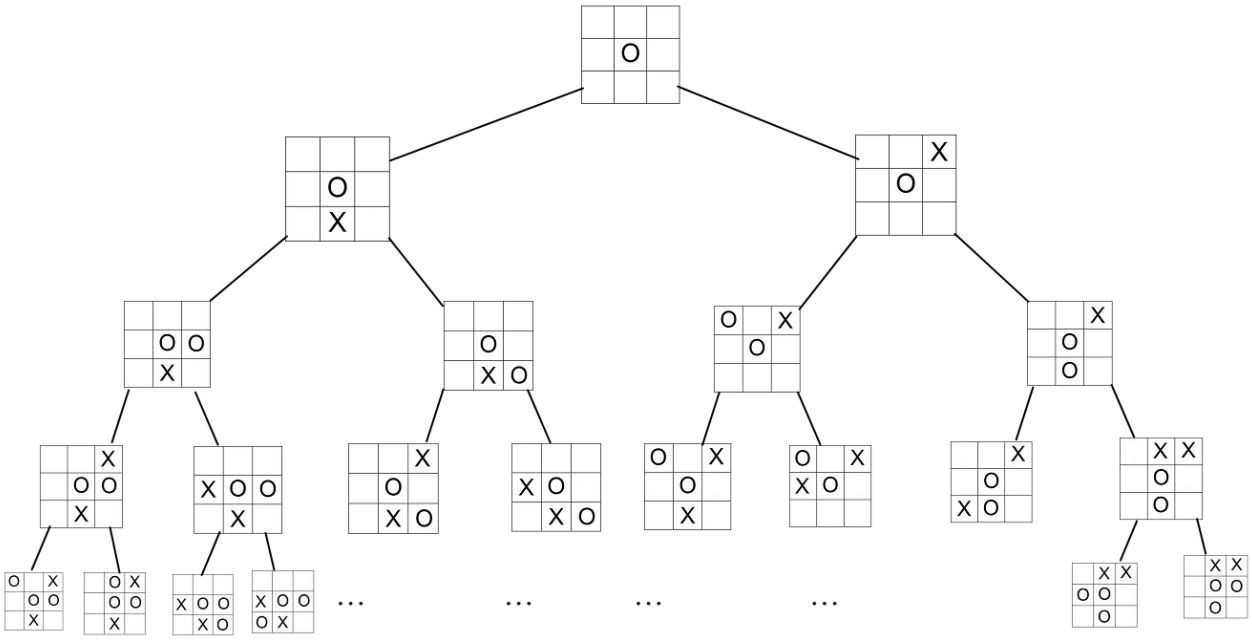
На основі кожного з двох станів ігрового поля (level 1 бінарного дерева) комп’ютер може створити ще два, записавши “0” у випадкові комірки.



Комп’ютер далі повторює цей процес і з кожної вершини (листа) другого рівня (level 2) отримає ще дві вершини із записаними “X” у випадкові комірки.



Дерево можна і далі розширювати (продовжувати будувати) записуючи “0” у дві випадкові але вільні комірки, що дозволить отримати четвертий рівень дерева на основі попереднього третього рівня:



Якщо автоматизувати (реалізувати програмно) цей процес розширення дерева то можна отримати повне бінарне дерево, яке буде представляти дерево гри. Бінарне дерево дозволяє комп’ютеру прийняти рішення про наступний хід. Якщо здійснити обхід (рекурсивно) правого і лівого піддерев та підрахувати кількість виграшних комбінацій на ігровому полі то наступним ходом буде хід

який зображений у позиці, яка знаходиться у корені піддерева з більшою кількістю виграшних комбінацій. Підрахунок виграшних комбінацій можна організувати наступним чином:

- якщо знайдена виграшна комбінація (виграшна комбінація на листі дерева) то повернути +1;
- якщо знайдена програшна комбінація (програшна комбінація на листі дерева) то повернути -1;
- якщо на ігровому полі нічия то повернути 0

Для дерева, яке зображено на рис.4. такою позицією буде позиція 3 на основі комбінації, яка є коренем правого піддерева.

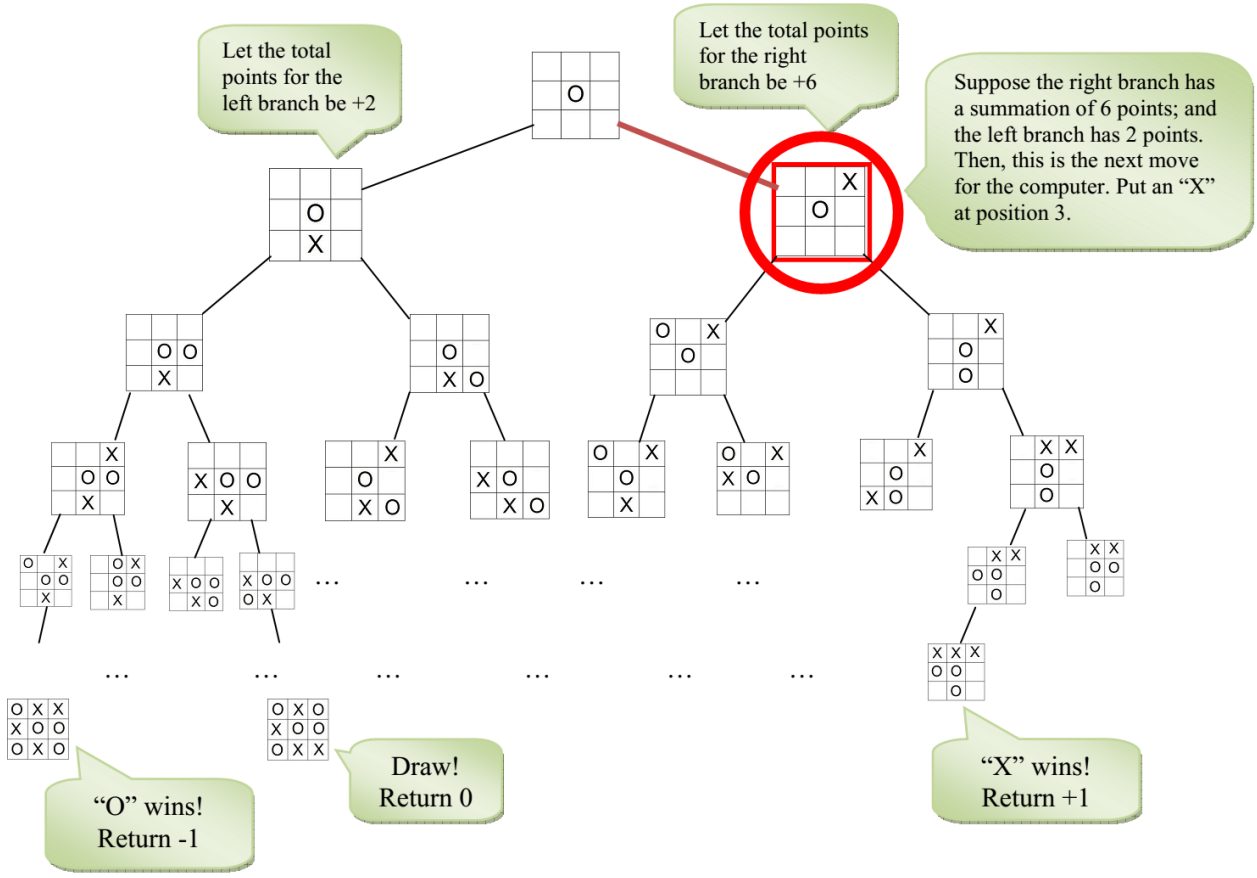
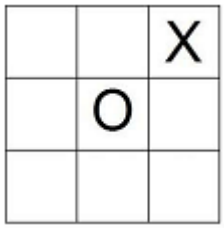


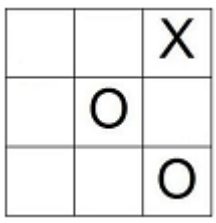
Рис. 4.

Якщо праве і ліве піддерева містять однакову кількість виграшних комбінацій то наступний хід вибирається випадковим чином з двох можливих.

Після ходу комп'ютера ігрове поле має наступний вигляд,



і якщо припустити що користувач зробив хід записавши "0" в дев'яту комірку



то ця комбінація буде для комп’ютера початковою для прийняття рішення про наступний хід.

Для того щоб комп’ютер зробив наступний хід поточне дерево можна знищити і побудувати нове де у корені буде нова початкова комбінація.

Якщо повторити вищеописаний процес певну кількість разів (скільки?) то можна отримати стан на ігровому полі коли переміг один з гравців або заповнені всі комірки (нічия).

Літератува:

1. Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, 2013 Chapter 8. Trees

2. Fundamentals of Python: Data Structures, Kenneth Lambert, 2013 Chapter 10. Trees