Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота № 3**

з дисципліни «Програмування інтелектуальних інформаційних систем »

Тема: «Дерева прийняття рішень 102»

Виконав: Перевірив:

студент групи ІТ-04 вик. кафедри ІПІ

Коновальчук Андрій Баришич Лука Маріянович

Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом \_\_\_\_\_\_\_\_\_\_\_

Київ 2022

# Лабораторна №3

*Тема:*

Дерева прийняття рішень 102

*Мета:*

Розробити програмне рішення дерев прийняття рішень для гри в Шахи

*Завдання:*

1. Реалізувати алгоритм NegaMax

1. Реалізувати алгоритм NegaScout
2. Реалізувати алгоритм PVS

# Опис програмного коду

Програмний код на GitHub: <https://github.com/KonovalchukA-IT04/PiisLabs/tree/master/lab3>

Програмний код реалізований на базі бібліотеки python-chess, де наперед підготована гра, візуалізація у вигляді вимальовування дошки у форматі SVG, а також API-інтерфейс, який дозволяє рухати фігурами та слідкувати за станом дошки. Також була використана бібліотека-надбудова python-chess-board, створена на pygame, яка дозволяє візуалізовувати дошку та слідкувати за грою. Тож завдання лабораторної роботи зводиться до написання алгоритмів NegaMax, NegaScout та PVS, функції для визначення евристики. Алгоритми протестовані і візуалізовані на грі двох ботів (чорні та білі) один проти одного.

1. *Евристика*

Еврестична функція (рис. 1.1) взята з відкритих джерел (статті на програмування ботів для шах) і модифікована під нашу лабораторну роботу.

Принцип простий: умови виходу/кінця гри; набір оцінок для кожної фігури; обрахунок кількості фігур; обрахунок евристики за формулою за кількістю фігур; обрахунок евристики за формолою для кожної фігури окремо; сума всіх евристик; зміна знаку в залежності від граючої сторони.

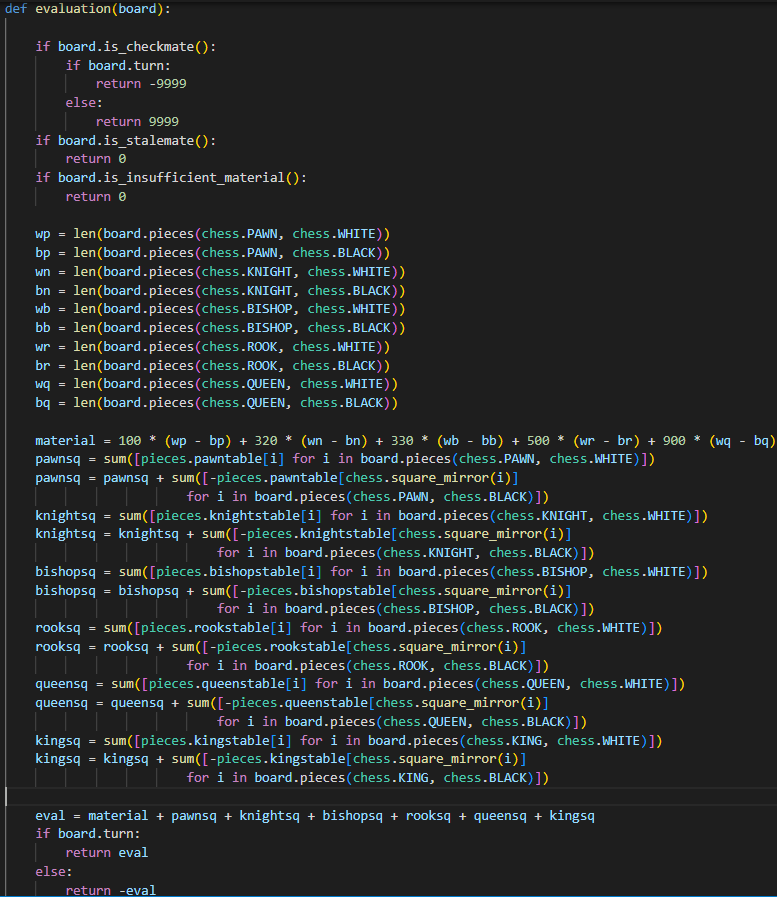


Рисунок 1.1 – Код еврестичної функції

1. *NegaMax*

Алгоритм Negamax – це варіант мінімаксного пошуку, який спирається на властивість нульової суми гри для двох гравців.Цей алгоритм грунтується на тому, що max(a,b)=-min(-a,-b) для спрощення реалізації мінімаксного алгоритму.

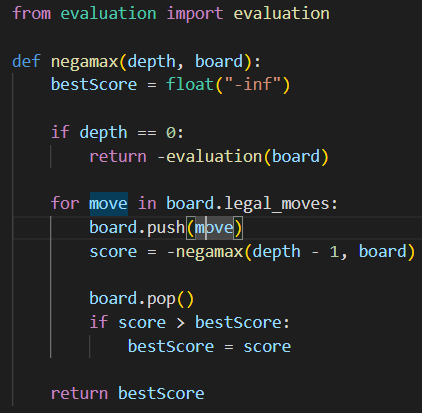


Рисунок 2.1 – Код алгоритму

Усі подальші функції будуть повертати саме еврестичну оцінку. А вибір кращого ходу відбувається далі, після виклику функції з алгоритмом. Кращі ходи записують в список, і список очищується, якщо знаходять ходи з кращою оцінкою. Якщо ходи мають однакову оцінку, то вони вибираються зі списку випадково.

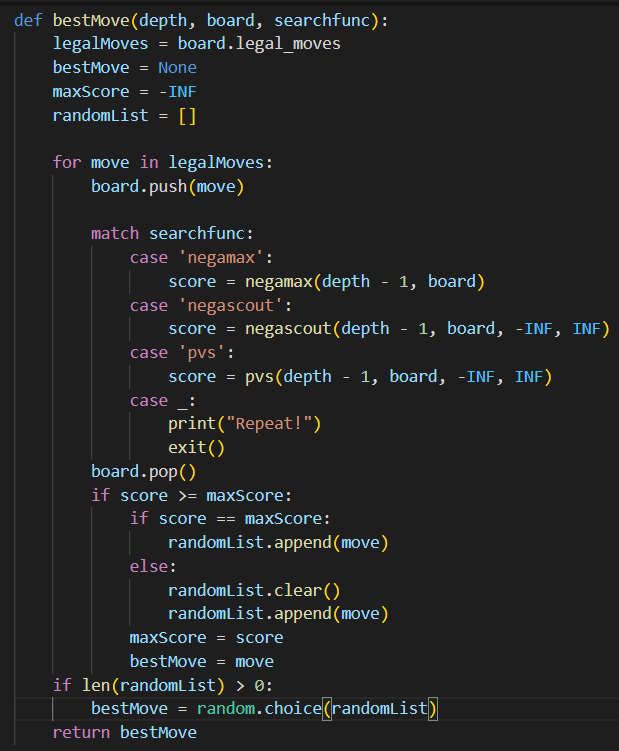


Рисунок 2.2 – Вибір накращого кроку

1. *NegaScout*

NegaScout є альфа-бета вдосконаленням алгоритму Negamax. Удосконалення прибирає і деякі помилки, що стосуються двох останніх шарів, які не потребують повторного пошуку. Дуже схожий з PVS, і тому про нього більше в наступному заголовку.

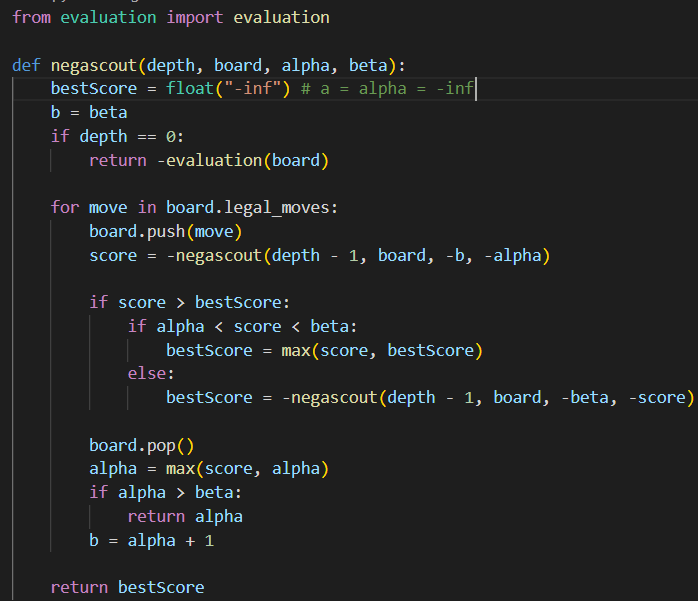


Рисунок 3.1 – Код алгоритму

1. *PVS*

PVS іноді ототожнюють з NegaScout, і як видно з коду, вони дійсно відрізняються всього лише одним рядком. Він є швидшим за alpha-beta pruning і домінує над ним тим, що він ніколи не досліджуватиме вузол, який можна відрізати альфа-бета-версією; однак він покладається на точне впорядкування вузлів, щоб скористатися цією перевагою.

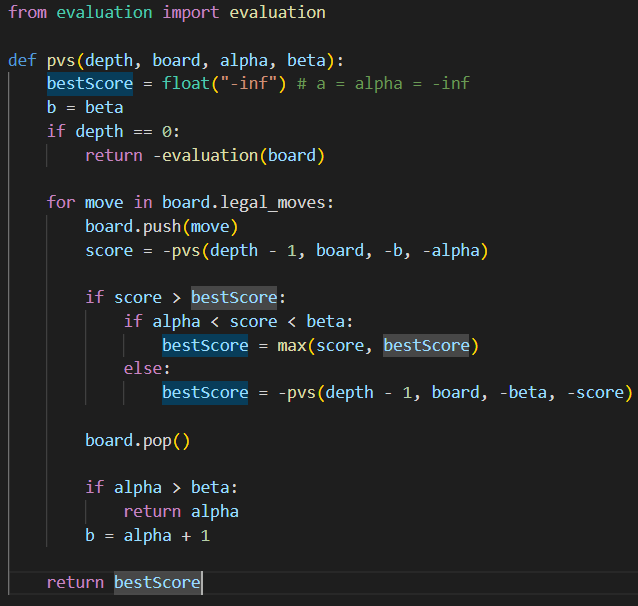


Рисунок 4.1 – Код алгоритму

# Висновок

В ході даної лабораторної роботи було створено програмне рішення дерев прийняття рішення для найоптимальніших рухів фігур на шаховій дошці.

Було реалізовано наступні алгоритми: NegaMax, NegaScout, PVS; також реалізовано еврестичну функцію для цих алгоритмів.

Лабораторна робота виконана на основі бібліотеки python-chess.

В звіті наявні описи алгоритмів, а також скріншоти з кодом.