Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота № 2**

з дисципліни «Програмування інтелектуальних інформаційних систем »

Тема: «Дерева прийняття рішень 101»

Виконав: Перевірив:

студент групи ІТ-04 вик. кафедри ІПІ

Зубрей Андрій Баришич Лука Маріянович

Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_

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

Київ 2022

# Лабораторна №2

Тема:

Дерева прийняття рішень 101

Мета:

Розробити програмне рішення дерев прийняття рішень для гри Pacman

Завдання:

1. Додати генерацію ворогів з поведінкою. При генерації ви вказуєте кількість згенерованих ворогів одного з двох типів. Перший тип шукає дорогу до гравця (оптимальність цього пошуку на вашій совісті). Другий тип рухається випадково
2. Ви маєте реалізувати алгоритми перемоги за вашою грою - minimax з alpha-beta pruning та expectimax. Функція оцінки має оцінювати “силу” поточної позиції - чим більше число, тим краща позиція.

# Опис програмного коду

Програмний код на GitHub: <https://github.com/AndriiZubrey/PIIS/tree/main/Lab2>

Програмний код реалізований на базі проекту Berkeley C188 Pacman, де наперед підготована гра, візуалізація, а також архітектура, яка дозволяє розширити і доповнити код. Тож завдання лабораторної роботи зводилось до написання алгоритмів MiniMax, Alpha-Beta pruning та Expectimax, функції для визначення евристики. Весь код в рамках даної лабораторної написаний у файлі multiAgents.py.

1. RandomGhost, DirectionalGhost

Дані методи були передбачені курсом Berkeley, і не потребують нашого редагування. Методи передбачають поведінку привидів з випадкового руку та переслідування Pacman. Для виклику відповідної поведінки потрібно вказати відповідний параметр (назва методу) з прапорцем -g при запуску гри.

1. Дерева прийняття рішень

2.1. MiniMax

MiniMax – правило прийняття рішень, що використовується в теорії ігор, теорії прийняття рішень, дослідженні операцій, статистиці і філософії для мінімізації можливих втрат з тих, які особа, яка приймає рішення не може уникнути при розвитку подій за найгіршим для неї сценарієм. У даній лабораторній роботі ми провели максимізацію дій Pacman та мінімізацію дій Ghost.

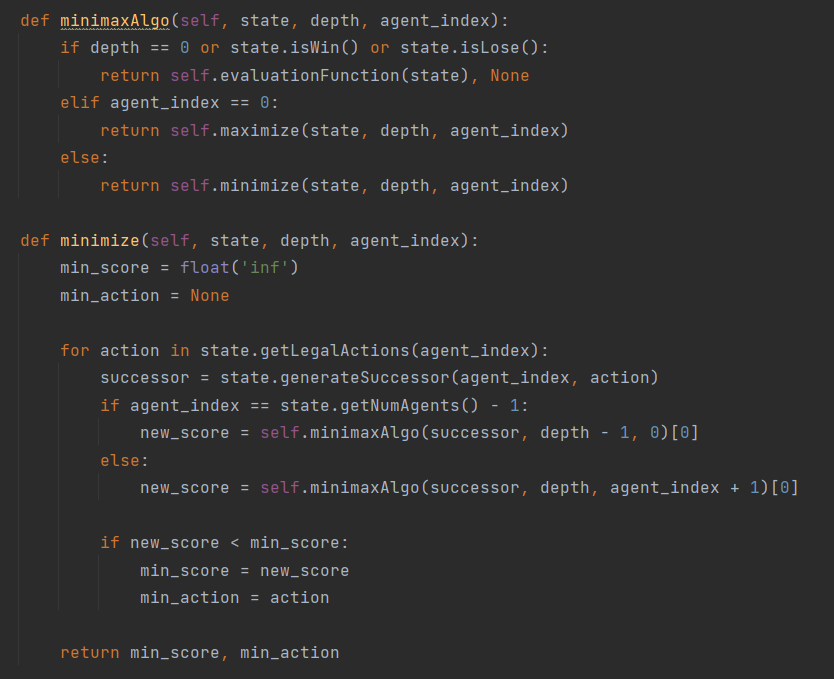


Рисунок 2.1.1 – Код алгоритму(частина 1)

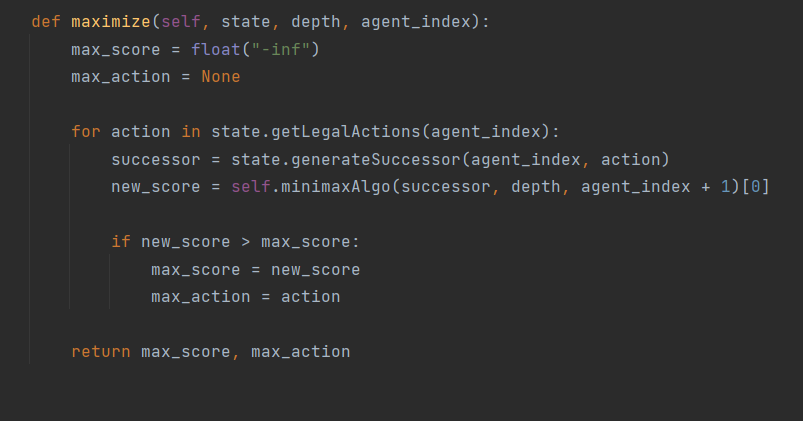


Рисунок 2.1.2 – Код алгоритму(частина 2)

Функцію не є монолітною, вона розбивається на максимізацію, мінімізацію і логіка через індексацію агентів agentIndex (0 – Pacman, >1 – Ghost), яка необхідна також для отримання відповідних дій (actions) із стану гри (State).Повертається значення еврестичної функції (evaluationFunction, котра буде розглянута пізніше), яка визначає в кінці виклику методу getActions агенту найкращі дії для Pacman. Розгляд глубини працює від 0 до вказаного у виклику програми максимуму (depth).

2.2. Alpha-betapruning

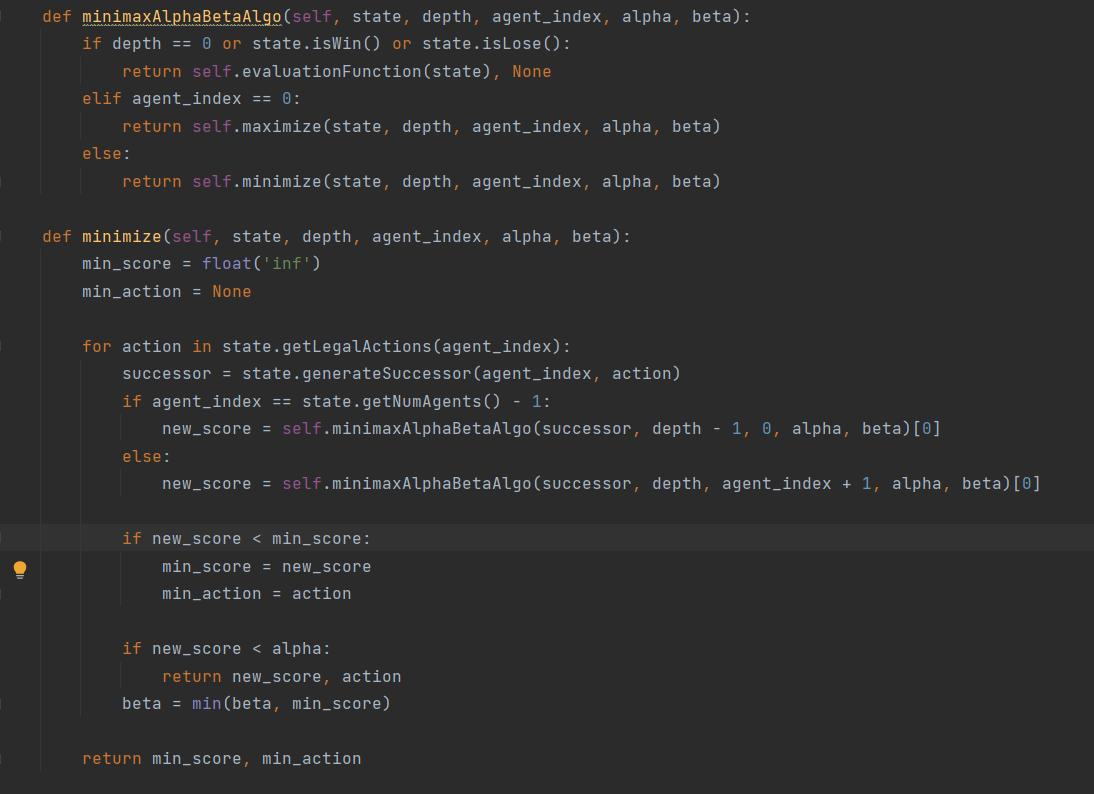
Алгоритм пошуку, що зменшує кількість вузлів, які необхідно оцінити в дереві пошуку мінімаксного алгоритму і при цьому дозволяє отримати ідентичний результат. 

Рисунок 2.2.1 – Код алгоритму(частина 1)

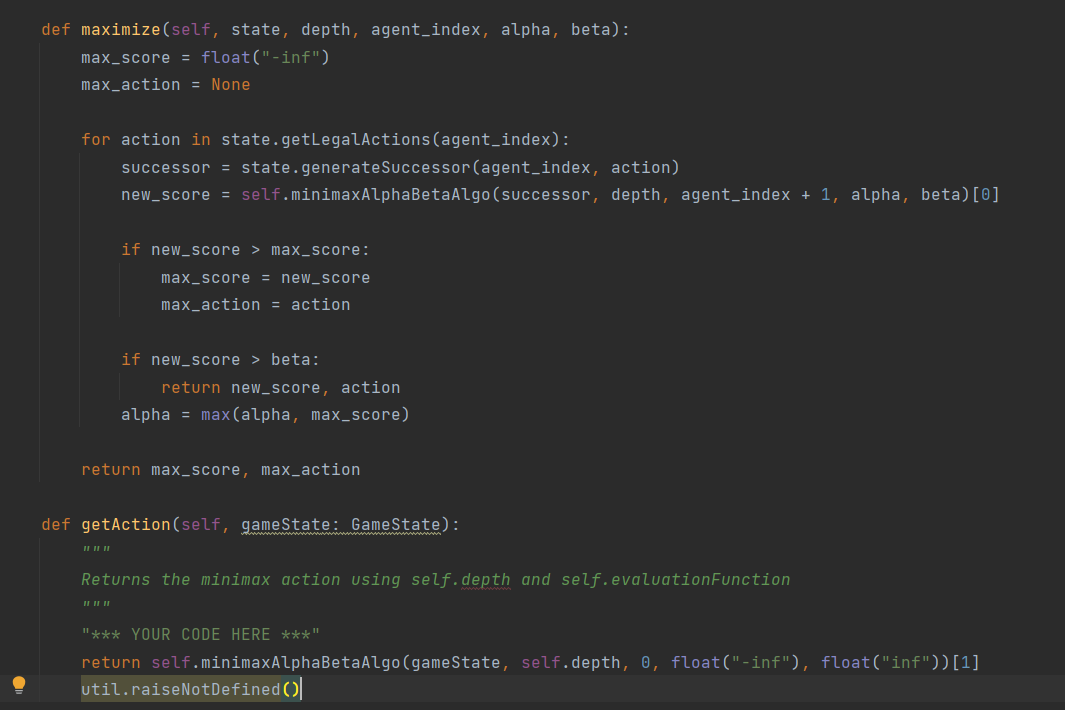


Рисунок 2.2.2 – Код алгоритму(частина 2)

Код алгоритму майже аналогічний попередньому. Відрізняється лише додатковою логікою “відсікань”, а також додатковими змінним (alpha, beta), необхідними для роботи цього алгоритму.

2.3 Expectimax

Алгоритм Expectiminimax є різновидом алгоритму MiniMax для використання в системах штучного інтелекту, які грають у ігри з нульовою сумою для двох гравців, такі як нарди, в яких результат залежить від комбінації навичок гравця та елементів випадковості, таких як кидання кубиків. На додаток до вузлів “min” і “max” традиційного мінімаксного дерева, цей варіант має вузли “chance” (“move by nature”), які приймають очікуване значення випадкової події, що відбувається.

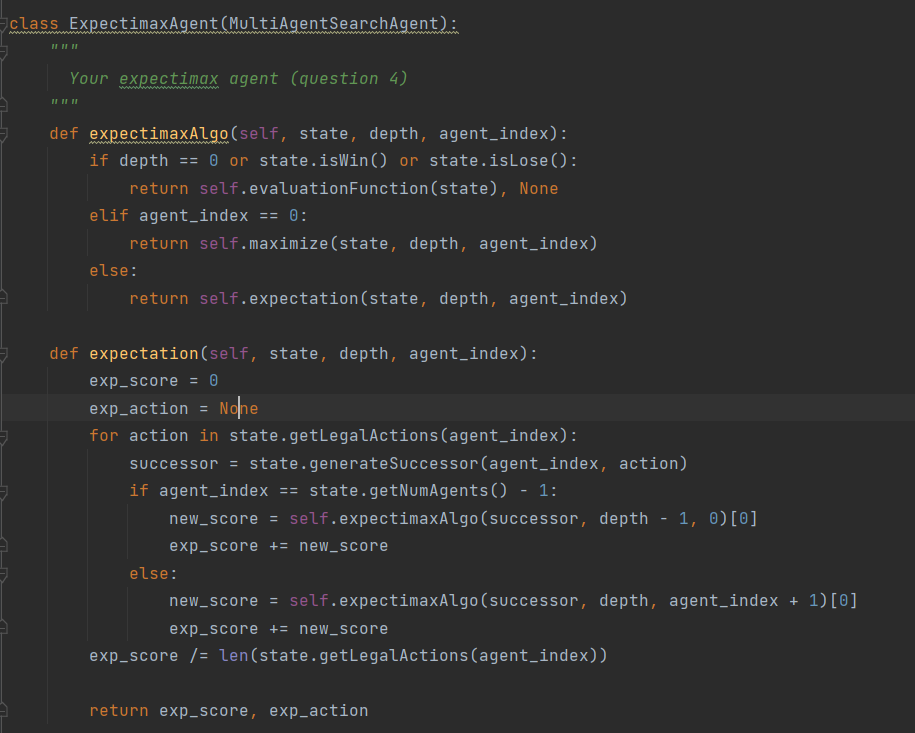


Рисунок 2.3.1 – Код алгоритму(частина 1)

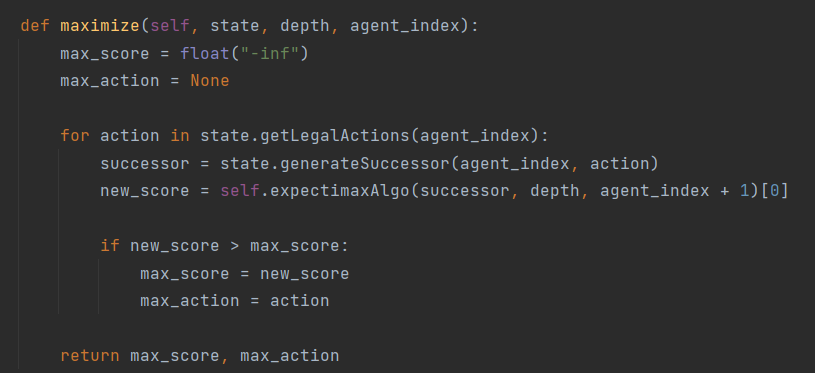
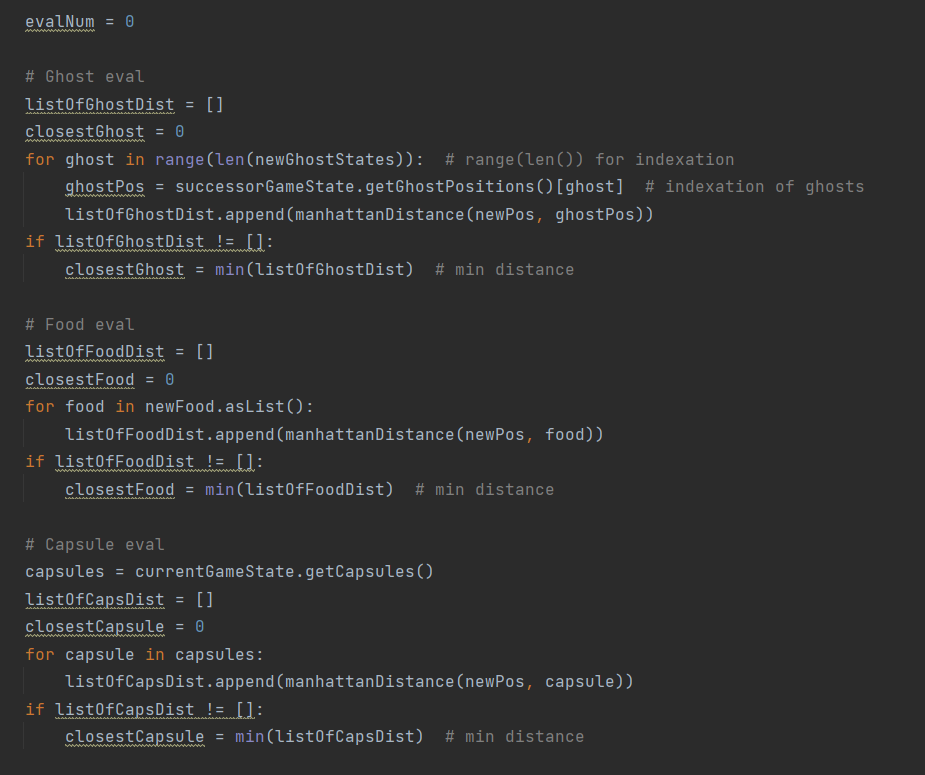


Рисунок 2.3.2 – Код алгоритму(частина 2)

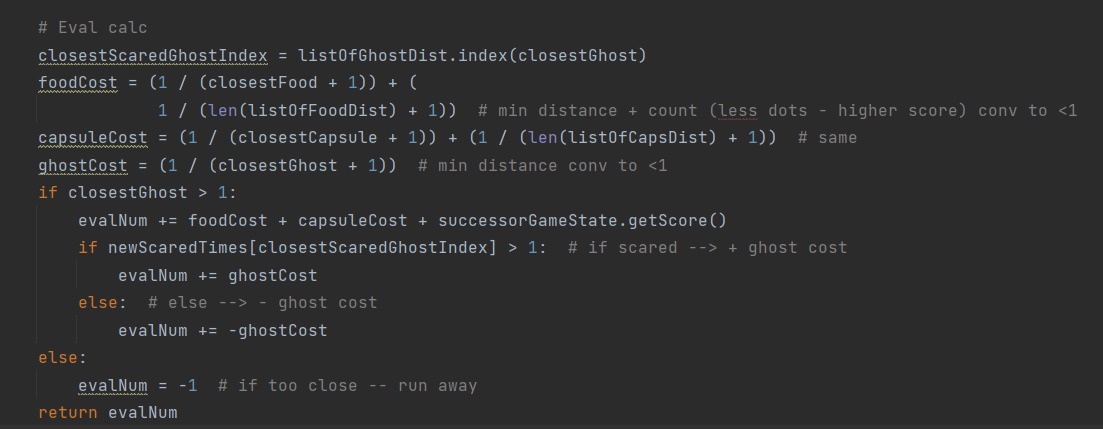
Код майже аналогічний до коду MiniMax. Під час мінімізації ми вираховуємо середнє арифметичне значення (сумуємо value і ділимо на довжину контейнеру actions). Та порівнюємо знайдені значення для кожного actions з “кращими значенням” на попередніх ітераціях.

2.4 Евристика

Для наших попередніх функцій треба було прописати підходящу евристику для Pacman.



Рисунко 2.4.1 – Код еврестичної функції(частина 1)



Рисунко 2.4.1 – Код еврестичної функції(частина 2)

В якості евристики ми вираховуємо обернені значення відстані до найближчої їжі, кількості їжі, відстані до найближчої капсули, кількості капсул, відстані до найближчого Ghost. Якщо Ghost наляканий, то приплюсовуємо додатнє значення, якщо не наляканий, то віднімаємо. Якщо привид занадто близько (відстань <1), то повертаємо від’ємне значення, як поганий вибір для наступного руху.

**Висновок**

В ході даної лабораторної роботи було створено програмне рішення дерев прийняття рішення для найоптимальнішого руху Pacman в лабіринті.

Було реалізовано наступні алгоритми: MiniMax, Alpha-Beta pruning, Expectimax; також реалізовано еврестичну функцію для цих алгоритмів.

Лабораторна робота виконана на основі проекту Berkeley C188 Pacman. Розібрано структуру та архітектуру проекту, і доповнені наступні файли: multiAgents.py.

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