**Bài 18: Gaming - Phần 1 - Lập trình AI bằng Python**

Trò chơi được chơi với một chiến lược. Mỗi người chơi hoặc đội sẽ đưa ra chiến lược trước khi bắt đầu trò chơi và họ phải thay đổi hoặc xây dựng chiến lược mới tùy theo (các) tình huống hiện tại trong trò chơi.

**1. Thuật toán Search :**

Bạn sẽ phải xem xét trò chơi máy tính cũng với chiến lược tương tự như trên. Lưu ý rằng Thuật toán tìm kiếm là thuật toán tìm ra chiến lược trong trò chơi máy tính.

**Tổng quan :**

Mục tiêu của các thuật toán tìm kiếm là tìm ra tập hợp các nước đi tối ưu để chúng có thể đến đích cuối cùng và giành chiến thắng. Các thuật toán này sử dụng tập hợp các điều kiện chiến thắng, khác nhau cho mọi trò chơi, để tìm ra nước đi tốt nhất.

Hình dung một trò chơi máy tính như một cái cây. Chúng ta biết rằng cây có các nút. Bắt đầu từ gốc, chúng ta có thể đi đến nút chiến thắng cuối cùng, nhưng với những nước đi tối ưu. Đó là công việc của các thuật toán tìm kiếm. Mỗi nút trong cây như vậy đại diện cho một trạng thái trong tương lai. Các thuật toán tìm kiếm tìm kiếm thông qua cây này để đưa ra quyết định ở mỗi bước hoặc nút của trò chơi.

**2. Combinational Search**

Nhược điểm chính của việc sử dụng các thuật toán tìm kiếm là chúng có bản chất là toàn diện, đó là lý do tại sao chúng khám phá toàn bộ không gian tìm kiếm để tìm ra giải pháp dẫn đến lãng phí tài nguyên. Sẽ cồng kềnh hơn nếu các thuật toán này cần tìm kiếm toàn bộ không gian tìm kiếm để tìm ra giải pháp cuối cùng.

Để loại bỏ loại vấn đề như vậy, chúng ta có thể sử dụng tìm kiếm tổ hợp sử dụng phương pháp khám phá không gian tìm kiếm và giảm kích thước của nó bằng cách loại bỏ các chuyển động sai có thể xảy ra. Do đó, các thuật toán như vậy có thể tiết kiệm tài nguyên. Một số thuật toán sử dụng heuristic để tìm kiếm không gian và tiết kiệm tài nguyên sẽ được trình bày sau đây

**3. Thuật toán Minimax**

Đây là chiến lược được sử dụng bởi tìm kiếm tổ hợp sử dụng phương pháp heuristic để tăng tốc chiến lược tìm kiếm. Khái niệm chiến lược Minimax có thể được hiểu với ví dụ về hai trò chơi người chơi, trong đó mỗi người chơi cố gắng dự đoán động thái tiếp theo của đối thủ và cố gắng giảm thiểu chức năng đó. Ngoài ra, để giành chiến thắng, người chơi luôn cố gắng phát huy tối đa chức năng của bản thân dựa trên tình hình hiện tại.

Heuristic đóng một vai trò quan trọng trong các loại chiến lược như Minimax. Mỗi nút của cây sẽ có một hàm heuristic được liên kết với nó. Dựa trên kinh nghiệm đó, người ta sẽ đưa ra quyết định di chuyển về phía nút có lợi nhất cho họ.

**4. Alpha-Beta Pruning**

Một vấn đề lớn với thuật toán Minimax là nó có thể khám phá những phần không liên quan của cây, dẫn đến lãng phí tài nguyên. Do đó phải có một chiến lược để quyết định phần nào của cây có liên quan và phần nào không liên quan và để phần không liên quan chưa được khám phá. Cắt tỉa Alpha-Beta là một trong những loại chiến lược như vậy.

Mục tiêu chính của thuật toán cắt tỉa Alpha-Beta là để tránh việc tìm kiếm những phần của cây mà không có bất kỳ giải pháp nào. Khái niệm chính của việc cắt tỉa Alpha-Beta là sử dụng hai giới hạn có tên là Alpha, giới hạn dưới tối đa và Beta, giới hạn trên tối thiểu. Hai tham số này là các giá trị hạn chế tập hợp các giải pháp khả thi. Nó so sánh giá trị của nút hiện tại với giá trị của các tham số alpha và beta, để nó có thể di chuyển đến phần của cây có lời giải và loại bỏ phần còn lại.

**5. Thuật toán Negamax**

Thuật toán này không khác với thuật toán Minimax, nhưng nó có cách triển khai thanh lịch hơn. Nhược điểm chính của việc sử dụng thuật toán Minimax là chúng ta cần xác định hai hàm heuristic khác nhau. Mối liên hệ giữa những khám phá này là ở chỗ, trạng thái của trò chơi càng tốt cho một người chơi, thì người chơi kia càng tệ hơn. Trong thuật toán Negamax, công việc giống nhau của hai hàm heuristic được thực hiện với sự trợ giúp của một hàm heuristic.

**6. Xây dựng bot để chơi game :**

Để xây dựng các bot để chơi hai trò chơi người chơi trong AI, chúng ta cần cài đặt thư viện easyAI. Nó là một khung trí tuệ nhân tạo cung cấp tất cả các chức năng để xây dựng trò chơi hai người chơi. Bạn có thể tải xuống như sau:

**pip** install easyAI

Trong trò chơi này, sẽ có một đống tiền xu. Mỗi người chơi phải lấy một số xu từ đống đó. Mục tiêu của trò chơi là tránh lấy đồng xu cuối cùng trong đống. Tôi sẽ sử dụng lớp LastCoinStanding được kế thừa từ lớp TwoPlayersGame của thư viện easyAI :

from easyAI **import** TwoPlayersGame, id\_solve, Human\_Player, AI\_Player

from easyAI.AI **import** TT

Bây giờ, kế thừa lớp từ lớp TwoPlayerGame để xử lý tất cả các hoạt động của trò chơi -

**class** **LastCoin\_game**(TwoPlayersGame):

**def** **\_\_init\_\_**(self, players):

Xác định người chơi và người chơi sẽ bắt đầu trò chơi.

self.players = players

self.nplayer = 1

Xác định số lượng xu trong trò chơi, ở đây chúng tôi đang sử dụng 15 đồng xu cho trò chơi.

self.num\_coins = 15

Xác định số xu tối đa mà người chơi có thể nhận được trong một nước đi.

self.max\_coins = 4

có một số điều nhất định cần xác định như được hiển thị trong đoạn mã sau. Xác định các bước di chuyển có thể.

**def** **possible\_moves**(**self**):

**return** [str(a) **for** a **in** range(1, **self**.max\_coins + 1)]

Xác định việc loại bỏ các đồng tiền

**def** **make\_move**(**self**, move):

**self**.num\_coins -= int(move)

Xác định ai đã lấy đồng xu cuối cùng.

**def** **win\_game**(**self**):

**return** **self**.num\_coins <= 0

Xác định thời điểm dừng trò chơi, đó là khi ai đó thắng.

**def** **is\_over**(**self**):

**return** **self**.win()

Xác định cách tính điểm.

**def** **score**(**self**):

**return** 100 **if** **self**.win\_game() **else** 0

Xác định số xu còn lại trong đống.

**def** **show**(**self**):

print(**self**.num\_coins, 'coins left in the pile')

**if** \_\_name\_\_ == "\_\_main\_\_":

tt = TT()

LastCoin\_game.ttentry = lambda self: **self**.num\_coins

Giải quyết trò chơi với đoạn code sau:

r, d, m = id\_solve(LastCoin\_game,

**range**(2, 20), win\_score=100, tt=tt)

print(r, d, m)

Quyết định ai sẽ bắt đầu trò chơi

**game** = LastCoin\_game([AI\_Player(tt), Human\_Player()])

game.play()

Bạn có thể tìm thấy kết quả sau và cách chơi đơn giản của trò chơi này :

d:2, a:0, m:1

d:3, a:0, m:1

d:4, a:0, m:1

d:5, a:0, m:1

d:6, a:100, m:4

1 6 4

15 coins left in the pile

Move #1: player 1 plays 4 :

11 coins left in the pile

Player 2 what do you play ? 2

Move #2: player 2 plays 2 :

9 coins left in the pile

Move #3: player 1 plays 3 :

6 coins left in the pile

Player 2 what do you play ? 1

Move #4: player 2 plays 1 :

5 coins left in the pile

Move #5: player 1 plays 4 :

1 coins left in the pile

Player 2 what do you play ? 1

Move #6: player 2 plays 1 :

0 coins left in the pile