Họ tên: *Dương Mạnh Cường*

MSSV: *1760273*

Nhóm: *01*

MÔ TẢ THUẬT TOÁN MIN-MAX

Thuật toán Min-Max trong Trí tuệ nhân tạo

- Min-Max hay còn được gọi là Minimax là một thuật toán dựa trên kĩ thuật Đệ Quy và Backtracking ứng dụng trong việc đưa ra quyết định và lí thuyết trò chơi. Nó cung cấp từng bước đi, trạng thái tối ưu cho người chơi (giả định rằng đối thủ cũng chơi một cách tối ưu).

- Thuật toán Min-Max dựa vào việc vẽ ra toàn bộ cây trò chơi và dựa vào đây để tìm đáp án tối ưu.

- Thuật toán Min-Max thường được sử dụng phổ biến trong các game có ứng dụng AI (Trí tuệ nhân tạo). Chẳng hạn trong một số game như: Cờ vua, Cờ đam, Tic-tac-toe, Cờ vây,... và trong nhiều loại trò chơi 2 người mang tính luân phiên khác.

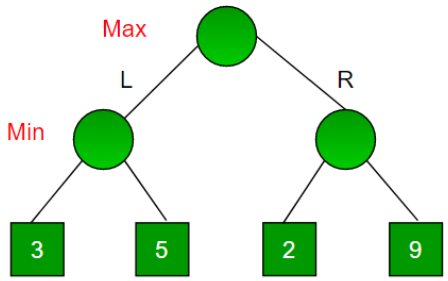
- Thuật toán này mô phỏng quá trình chơi của 2 người chơi, trong đó một người được gọi là MAX và người còn lại sẽ được gọi là MIN. Người MAX luôn muốn chiến thắng và có được điểm số cao nhất có thể và ngược lại người MIN sẽ cố gắng tối ưu hóa điểm số của người MAX xuống mức thấp nhất có thể.

- Thuật toán này tương tự như DFS, sẽ thực hiện tìm kiếm theo chiều sâu, qua đó từng bước vẽ ra từng trạng thái của trò chơi cho đến khi không thể vẽ được nữa, tức lúc này cây trò chơi đã được vẽ hoàn chỉnh.

- Min-Max vận hành theo cách từ một trạng thái hiện tại, sẽ đi đến các trạng thái tiếp theo (tức các trạng thái mà đối thủ sẽ đi được), lần lượt như thế đến một trạng thái đích nào đó (tức là node lá của cây) sẽ thực hiện quay lui (backtracking) lại, mỗi lần quay lui như vậy sẽ cho ra một đáp án, từ đó so sánh với các đáp án của các lần quay lui trước đó tại trạng thái hiện tại mà xác định đâu là đáp án tối ưu nhất cho trạng thái hiện tại.

CÁCH THUẬT TOÁN MIN-MAX THỰC HIỆN

Ví dụ: *Xem xét trò chơi dưới đây bao gồm 4 trạng thái cuối cùng (node lá của cây). Giả sử bạn là người chơi tối đa (tức MAX), được phép đi đầu tiên và một điều quan trọng là đối thủ của bạn là một hệ thống AI tiên tiến luôn luôn tìm ra cách đi để tối ưu hóa điểm số của bạn nhất có thể. Hãy cho biết điểm số cao nhất mà bạn đạt được sau khi trò chơi kết thúc.*

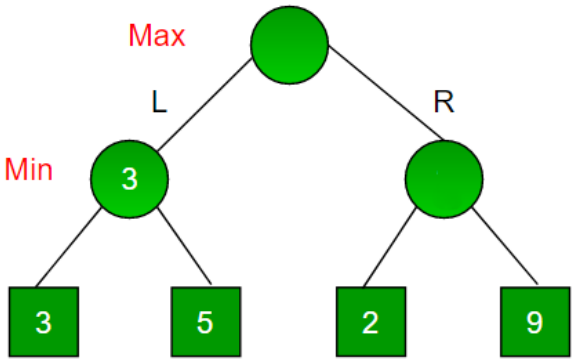


- **Step 1:**

Đây là trạng thái ban đầu, người MAX được phép đi trước, giả sử sẽ đi từ trái qua phải => MAX sang trái

- **Step 2:**

Đây là lượt đi của MIN, MIN có thể chọn 1 trong 2 trạng thái là 3 và 5, và vì luôn muốn cho MAX thấp điểm nhất có thể nên MIN chọn 3, sau đó quay lui lại trạng thái trước đó là lượt đi của MAX

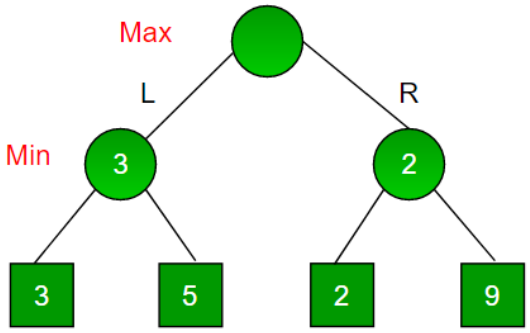


- **Step 3:**

Lúc này, MAX thấy các trạng thái bên phải chưa kiểm tra (tức cây con bên phải), MAX thực hiện tìm kiếm bên phải => Tới lượt đi của MIN

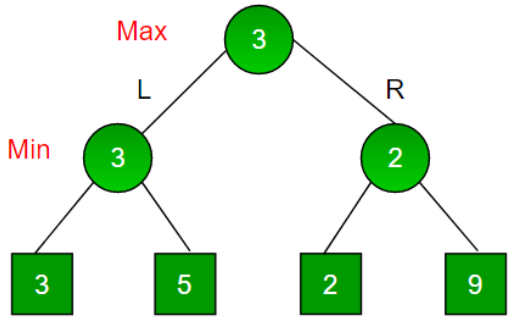
- **Step 4:**

Đây là lượt đi của MIN, có 2 trạng thái đích là 2 và 9, MIN sẽ chọn 2 sau đó quay lui lại lần trạng thái trước đó là lượt đi của MAX



- **Step 5:**

Lúc này MAX đã có hết đầy đủ dữ kiện (do cây trò chơi đã vẽ xong), vì luôn muốn có được điểm số cao nhất nên giữa 3 và 2, MAX chọn 3 và đây là kết quả cao nhất của bài toán này.



Code Python:

```python
import math
# Các tham số lần lượt là: độ sâu hiện tại, index của trạng thái này trong
# mảng scores ban đầu, lượt đi của ai (True là của MAX, False là của MIN),
# mảng chứa các trạng thái đích, và độ sâu của cây trò chơi hoàn chỉnh
def minimax (curDepth, nodeIndex,
             maxTurn, scores,
             targetDepth):

    # Trường hợp cơ sở: khi đã đi đến được một trạng thái đích => trả về điểm
    # số tại trạng thái này
    if (curDepth == targetDepth):
        return scores[nodeIndex]

    # Nếu là lượt đi của MAX => thực hiện tìm điểm tối đa của cây con trái
    # và cây con phải
    if (maxTurn):
        return max(minimax(curDepth + 1, nodeIndex * 2,
                   False, scores, targetDepth),
                   minimax(curDepth + 1, nodeIndex * 2 + 1,
                   False, scores, targetDepth))
    # Nếu là lượt đi của MIN => thực hiện tìm điểm tối thiểu của cây con trái
    # và cây con phải
    else:
        return min(minimax(curDepth + 1, nodeIndex * 2,
                     True, scores, targetDepth),
                   minimax(curDepth + 1, nodeIndex * 2 + 1,
                     True, scores, targetDepth))

# Các trạng thái cuối cùng
scores = [3, 5, 2, 9, 12, 5, 23, 23]

# Tính độ sâu của cây trò chơi hoàn chỉnh
treeDepth = math.log(len(scores), 2)

print("The optimal value is : ", end = "")
print(minimax(0, 0, True, scores, treeDepth))
```