AIHW1: 미니맥스 틱택토

2016104142 이광원

차례

프로그램 조건

- 사용자가 MAX Player가 되고 AI가 MIN Player가 되도록 대국하는 틱택토 프로그램이다.
- AI는 탐색한 노드 총 개수를 프린트 한다.
- Python으로 구현한다.

프로그램 흐름

- 초기 보드 출력
- 사용자(MAX Player)가 행동
- 결과 출력
- AI(MIN Player)가 행동
- 결과 출력
- 위 4단계 반복
- 게임 종료(비김) / 승자 발생 시 메시지 출력
- 프로그램 종료

전역 변수 선언

```
#user의 보드, ai의 보드, 탐색 카운터 기록을 list로 선언
user, ai, searches, search = [], [], [], -1
#초기 게임보드 선언
board = list(range(1,10))
#user를 MAX Player(첫 차례)로 설정
c_Player = True
```

- 게임 보드, 사용자가 차지한 칸, AI가 차지한 칸, 노드 탐색 기록 모두 리스트로 구현을 한다.
- 노드 탐색 횟수는 메인 프로그램에서 0으로 초기화 된 후 탐색 시 1씩 증가 하며, 탐색 완료 시 기록 리스트에 저장 된다.
- 사용자는 MAX Player로 지정 된다.

메인 프로그램 구현

```
#메인 프로그램
print_board() #초기 게임보드 출력
while 1:
    search = 0 #탐색 횟수 초기화
    if len(board) == 0: #승자 없이 게임 종료
        print("Result: Draw")
        break
    #Player 차례 실행, 승자가 있는지 확인
    if turn(c_Player):
        if c_Player:    #user 승리
            print("Result: You Win")
            break
        else:           #ai 승리
            print("Result: AI Win")
            break
    #승자가 없으면 다음 player로 전환
    else:
        if c_Player:    #Max → Min 전환
            c_Player = False
        else:           #Min → Max 전환
            searches.append(search) #탐색횟수 기록
            c_Player = True
#프로세스 내 총 탐색 횟수 출력
print("Total number of nodes searched:", sum(searches))
input() #종료 대기
```

- 게임의 진행 여부를 결정 한다.
- 게임 결과에 따른 적절한 메시지를 출력한다.
- 게임이 끝날 때 까지 사용자와 AI에게 번갈아 가며 차례를 준다.
- 마지막에 AI의 총합 노드 탐색 횟수를 출력 한다.

Player 차례 구현

```python
#각 Player의 차례 실행, 결과 확인 및 출력
def turn(player):
    if player : #user의 차례
        n = int(input("Make a move:")) #input으로 Box 선택
        move(board, user, n)
        print_board()
        return check_winner(user)
    else:        #ai의 차례
        n, x = minimax(False) #미니맥스 알고리즘으로 Box 선택
        #탐색한 노드 수 출력
        print("AI made a move. Number of nodes searched:", search)
        move(board, ai, n)
        print_board()
        return check_winner(ai)
```

승리 조건 확인 함수

```python
#player가 승리 조건을 가지고 있는지 확인
def check_winner(player):
    #틱택토 승리조건
    winCases = [[1,2,3], [1,4,7], [1,5,9], [2,5,8],
        [3,5,7], [3,6,9], [4,5,6], [7,8,9]]
    for case in winCases:
        check = 0
        #리스트 비교
        for i in range(3):
            if case[i] in player:
                check += 1
        if check == 3:  #승리조건에 부합하는 경우
            return True
    return False
```

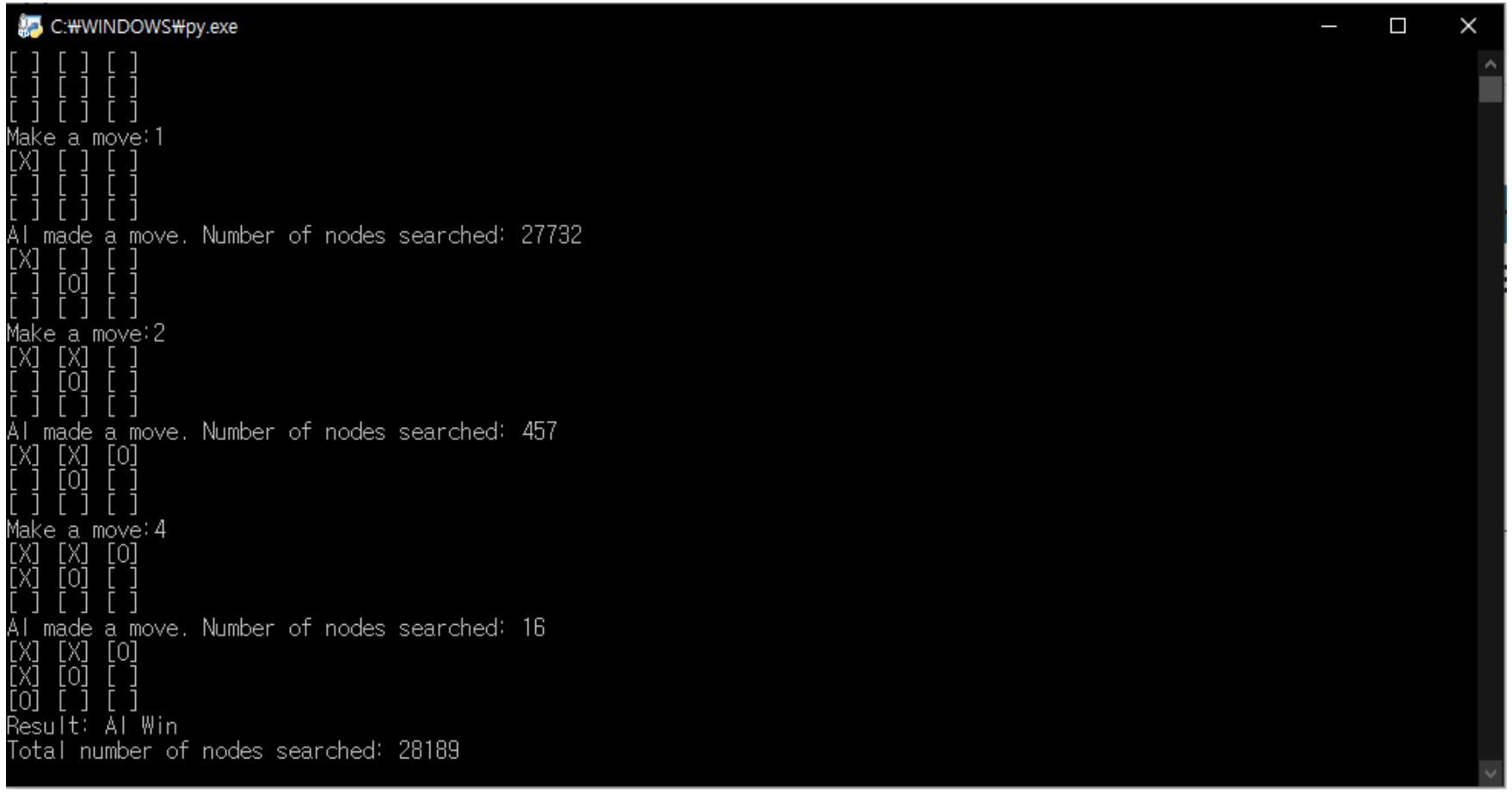
미니맥스 알고리즘

```python
#틱택노 게임의 미니맥스 알고리즘
def minimax(player):
    pos = -1
    #게임이 종료된 경우
    if len(board) == 0 or check_winner(user) or check_winner(ai):
        return -1, evaluation()
    if player:  #Max Player
        value = -100    #음의 무한대로 시작
        for box in board:
            move(board, user, box)
            x, score = minimax(False)   #Min Player로 전환
            move(user, board, box)
            if score > value:   #value와 위치 결정
                value = score
                pos = box
    else:       #Min Player
        value = +100    #양의 무한대로 시작
        for box in board:
            move(board, ai, box)
            x, score = minimax(True)    #Max Player로 전환
            move(ai, board, box)
            if score < value:   #value와 위치 결정
                value = score
                pos = box
    return pos, value
```

게임 플레이 – 진행 중


C:₩WINDOWS₩py.exe
[ ] [ ] [ ]
[ ] [ ] [ ]
[ ] [ ] [ ]
Make a move:1
[X] [ ] [ ]
[ ] [ ] [ ]
[ ] [ ] [ ]
AI made a move. Number of nodes searched: 27732
[X] [ ] [ ]
[ ] [O] [ ]
[ ] [ ] [ ]
Make a move:

게임 플레이 – AI 승리 시


C:₩WINDOWS₩py.exe
[ ] [ ] [ ]
[ ] [ ] [ ]
[ ] [ ] [ ]
Make a move:1
[X] [ ] [ ]
[ ] [ ] [ ]
[ ] [ ] [ ]
AI made a move. Number of nodes searched: 27732
[X] [ ] [ ]
[ ] [O] [ ]
[ ] [ ] [ ]
Make a move:2
[X] [X] [ ]
[ ] [O] [ ]
[ ] [ ] [ ]
AI made a move. Number of nodes searched: 457
[X] [X] [O]
[ ] [O] [ ]
[ ] [ ] [ ]
Make a move:4
[X] [X] [O]
[X] [O] [ ]
[ ] [ ] [ ]
AI made a move. Number of nodes searched: 16
[X] [X] [O]
[X] [O] [ ]
[O] [ ] [ ]
Result: AI Win
Total number of nodes searched: 28189

게임 플레이 – 비긴 경우


C:₩WINDOWS₩py.exe
[O] [X] [ ]
[ ] [X] [ ]
[ ] [ ] [ ]
AI made a move. Number of nodes searched: 473
[O] [X] [ ]
[ ] [X] [ ]
[ ] [O] [ ]
Make a move:7
[O] [X] [ ]
[ ] [X] [ ]
[X] [O] [ ]
AI made a move. Number of nodes searched: 21
[O] [X] [O]
[ ] [X] [ ]
[X] [O] [ ]
Make a move:4
[O] [X] [O]
[X] [X] [ ]
[X] [O] [ ]
AI made a move. Number of nodes searched: 2
[O] [X] [O]
[X] [X] [O]
[X] [O] [ ]
Make a move:9
[O] [X] [O]
[X] [X] [O]
[X] [O] [X]
Result: Draw
Total number of nodes searched: 26368

결론

- AI는 미니맥스 알고리즘으로 최선의 수를 두기 때문에 사용자에게는 게임을 비기는 것이 최대 기대 값이다.
- AI의 노드 탐색 횟수는 사용자의 플레이에 따라 달라지지만, 상하좌우 대칭적으로는 같은 값을 가진다.
- 미니맥스는 깊이 우선 탐색이고, 시간 복잡도가 높은 편이다.