**Alpha-Beta Pruning Search with Time Limit**

**컴퓨터학과**

**2015410079**

**이준경**

1. **Main.py**

**파이썬 외부모듈 pygame을 이용하여 오목 게임의 화면을 출력해주고 입력을 받는 GUI와 게임의 흐름을 구성하였다.**

1. **State.py**

**바둑판의 상태와 바로 다음에 놓을 수 있는 돌의 위치(actions), 가장 좋은 수(bestAction)를 저장하는 클래스이다.**

**terminalTest()를 통해 게임이 끝나거나 정해진 가장 깊은 노드에 닿았을 때를 알려준다.**

1. **Alpha\_beta\_Search.py**

**alpha\_beta pruning 과 iterative deepening search를 구현한 모듈이다.**

1. **Update(state, I, j, p)**

**돌을 놓을 곳을 입력 받아 바둑판을 업데이트 시켜주며, 다음 노드로 향할 수 있는 (다음 돌을 놓을 수 있는 곳들) actions 를 업데이트 해준다. 바둑판의 모든 점에 대해 노드를 만들게 되면 실행시간이 비효율적으로 증가하므로 현재 놓인 돌에서 다섯 칸 이내의 곳들만 actions에 추가해준다.**

1. **alphaBetaSearch(state)**

**pygame의 타이머기능을 이용하였다.**

**d를 통해 점차 깊이를 더해가며 각 깊이마다 최적의 결과를 찾는다. 100초의 시간을 정해 놓고 그 안에 가장 깊은 d값으로 찾은 결과를 반환한다.**

1. **maxValue(state, a, b, d)**

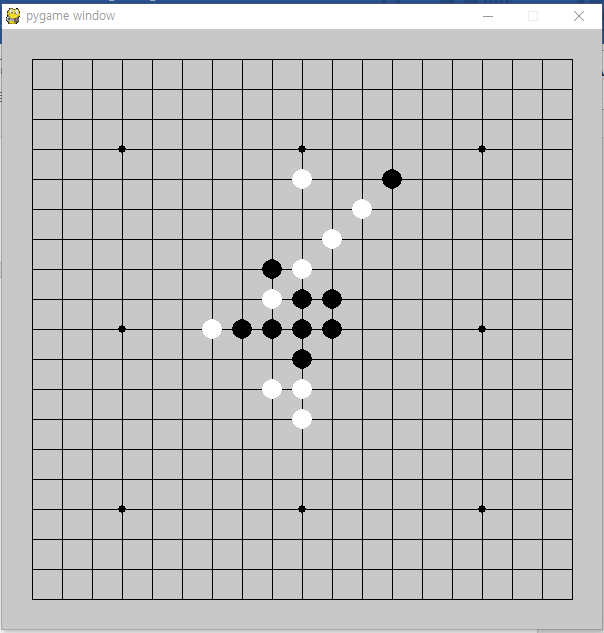
**state.actions를 통해 사용자의 입장에서 가능한 모든 수들을 두면서 다음의 컴퓨터 입장에서의 최소값들 중 최대값을 구하는 수를 찾는다. 만약 자식 노드에서 형제노드에서 얻은 값보다 큰 값을 발견하면 부모인 minValue에 의해 어차피 현재 노드가 선택되지 않을 것이므로 현재 노드의 탐색을 중단한다. 자식 노드의 값을 구하는 과정에서 d를 증가시켜 깊이를 더해준다. 각각의 actions에 대한 자식노드의 값을 얻어오기 위해 현재의 state의 사본을 만들어서 action의 값으로 update한 temp\_state에 대해 minValue를 실행한다.**

1. **minValue(state, a, b, d)**

**state.actions를 통해 컴퓨터 입장에서 가능한 모든 수들을 두면서 다음의 사람 입장에서의 최대값들 중 최소값을 구하는 수를 찾는다. 만약 자식 노드에서 형제노드에서 얻은 값보다 작은 값을 발견하면 부모인 maxValue에 의해 어차피 현재 노드가 선택되지 않을 것이므로 현재 노드의 탐색을 중단한다. 자식노드의 값을 구하는 과정에서 d를 증가시켜 깊이를 더해준다. 각각의 actions에 대한 자식노드의 값을 얻어오기 위해 현재의 state의 사본을 만들어서 action의 값으로 update한 temp\_state에 대해 maxValue를 실행한다.**

1. **Eval.py**

**바둑판 위의 돌들에 대해 정보가 주어지면 점수로 환산 시켜주는 모듈이다. (블랙보드에 김기혁님이 올려주신 c헤더파일을 파이썬 모듈로 변환해 사용하였습니다.)**



**실행화면**