Report

109550039 楊富翔

1. 實驗目的 ：
   透過實作 AI 演算法，找到贏下 nim game on the board 的最佳解。
2. 實驗方法 ：
   本次作業我實作的兩種方法，分別為 mcts 以及 minimax
3. 程式碼 ：
   A. Minimax :

```
if depth == 0:
    return random.choice(getLegalSteps(mapStat)), evaluate(mapStat)
if maximizingPlayer:
    bestScore = float('-inf')
    steplist = getLegalSteps(mapStat)
    bestStep = steplist[0]
    for step in steplist:
        mapStatCopy = copy.deepcopy(mapStat)
        mapStatCopy = applyStep(mapStatCopy, step)
        _, score = minimax(mapStatCopy, depth - 1, alpha, beta, False)
        if score > bestScore:
            bestScore = score
            bestStep = step
        alpha = max(alpha, bestScore)
        if alpha >= beta:
            break
    return bestStep, bestScore
else:
    bestScore = float('inf')
    steplist = getLegalSteps(mapStat)
    bestStep = steplist[0]
    for step in steplist:
        mapStatCopy = copy.deepcopy(mapStat)
        mapStatCopy = applyStep(mapStatCopy, step)
        _, score = minimax(mapStatCopy, depth - 1, alpha, beta, True)
        if score < bestScore:
            bestScore = score
            bestStep = step
        beta = min(beta, bestScore)
        if alpha >= beta:
            break
    return bestStep, bestScore
```

   在 minimax 中我使用 alpha-beta pruing 去減少樹的分支，但 minimax 的分支依然很龐大，我最多只能使用到 4 層

   B. MCTS :

```python
# The UCT algorithm balances exploration and exploitation in the tree search
def UCT(node):
    if node.n == 0:
        return float('inf')
    return node.w / node.n + 1.414 * np.sqrt(np.log(node.parent.n) / node.n)

class Node:
    def __init__(self,mapStat,player = 1,parent = None, step = None):
        self.mapStat = mapStat
        self.parent = parent
        self.player = player
        self.step = step
        self.children = []
        self.n = 0
        self.w = 0

    def game_over(self,mapStat):
        if len(getLegalSteps(mapStat))==0:
            return True
        else:
            return False

    def expand(self,mapStat):
        if self.game_over(mapStat):
            return
        steps = getLegalSteps(mapStat)
        for move in steps:
            mapStatCopy = copy.deepcopy(mapStat)
            child_mapStat = applyStep(mapStatCopy, move)
            chlid_node = Node(child_mapStat,3-self.player,self,move)
            self.children.append(chlid_node)
```

```python
def select(self):
    node = self
    while node.children:
        # Select the best child node based on the UCT formula
        node = max(node.children, key=UCT)
    # Otherwise, return the best child based on the UCT formula
    return node

def update(self, result):
    self.n += 1
    self.w += result

def rollout(self,mapStat,player):
    if self.game_over(mapStat):
        return
    mapStatCopy = copy.deepcopy(mapStat)
    steps = getLegalSteps(mapStatCopy)
    if len(getLegalSteps(mapStatCopy))==1:
        if player == 1:
            return False
        elif player == 2:
            return True
    step = random.choice(steps)
    mapStatCopy = applyStep(mapStatCopy,step)
    result = self.rollout(mapStatCopy,3-player)
    return result

def backpropagate(self,result):
    node = self
    node.update(result)
    if node.parent:
        node.parent.backpropagate(result)
```

MCTS 雖然可以跑得比較快但經過測試，皆會輸給 minimax 因為 MCST 只是利用機率去判斷，而 minimax 則是思考了所有的分支，可以說 minimax 是 MCTS 的最佳解。

4. 實驗結果 ：

   我最後採用的方法是，當 free_region>=16 的時候使用 MCTS，反之則使用 minimax，並且我將方向[1,7]改到剩方向[1,4]因為方向相加為 7 兩者是等價，這樣 minimax 的層數可以到 6 層，但需要注意的是 minimax 一開始的一步還是有可能會超時。