整體架構使用alpha beta purning

Getstep(): 我把整個code包在GetBestStep()裡面，而這整份code並沒有使用到gameStat的部分。

一張含有 文字 的圖片

自動產生的描述

GetLegalMovements(): 先用範例code本身就有的checkRemainMove來拿到有哪些點是還沒被occupied的，而這些點都能夠單獨成為一個legal move，所以先把它們加入list。然後對這些點只找1, 2, 3方向有沒有合法長度超過2的legal move，因為方向4-6的會有跟1-3一樣的movement，這樣就可以找到所有不重複的合法步數。

一張含有 文字 的圖片

自動產生的描述

UpdateMapStat(): 更新地圖

一張含有 文字 的圖片

自動產生的描述

GetBestStep(): 包含兩個小function，Min/MaxAgent，兩個Agent做的事情基本上相同，先看遊戲有沒有結束，結束了就直接回傳贏或輸，否則就去找所有合法步數然後初始化極大/極小值，然後對所有的movement做MinMax搜尋，直到層數足夠或是被alpha或beta剪枝掉，然後就會找到一個最佳的movement。

一張含有 文字 的圖片

自動產生的描述

這是GetBestStep的下半部分，處理掉一些極端值，如果合法步數只有一個的話就不特別計算，如果有太多合法步數的話會算不完，就隨機挑選一個，還有如果他在minmax的途中有出現錯誤也能夠透過random來稍微修正。

一張含有 文字 的圖片

自動產生的描述

GetScore(): 如果legal的起始點是0個就代表對方輸了，勝1代表對方贏了，因為給GetScore的地方都是放”給對手”的地圖，所有讓對方輸，自己可以拿到正分，而同時希望越少點數留在場上越好，越能夠推算出必勝的棋盤。

一張含有 文字 的圖片

自動產生的描述

isEnd(): 概念和GetScore的判斷依據類似，用剩下的點數來知道場上是不是已經沒有地方可以填了。

一張含有 文字 的圖片

自動產生的描述

Experiments: 透過和Sample\_2對打，不論先手或後手在10次的測試之下都能夠全勝，和朋友對打也有全勝和55開的紀錄，但我發現我的程式在先手的情況下勝率會略高一點，但也不排除是實驗誤差。

Experiences: 我在最一開始的時候只能夠算3層且15個legal move以下的深度，但在經由方向的限制和剪枝之後，可以算到4層且19個legal move的複雜程度，大大的增加了我的勝率。我也可以算5層13個move的難度，但因為很容易比其他人晚開始計算，所以會比我現在的參數還容易輸。