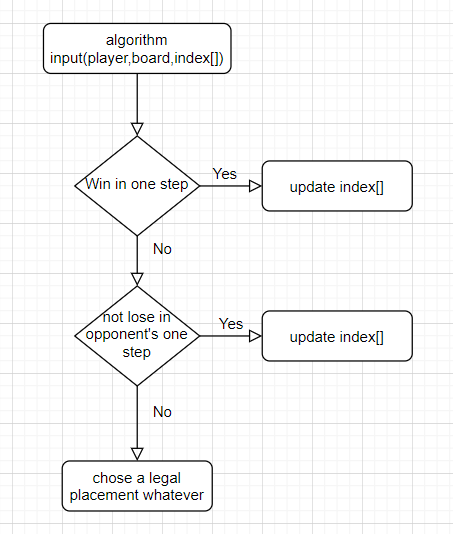
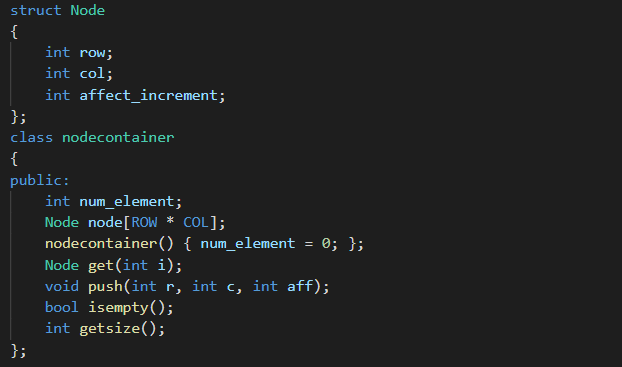
**Data structure final project “*chain\_reaction”***

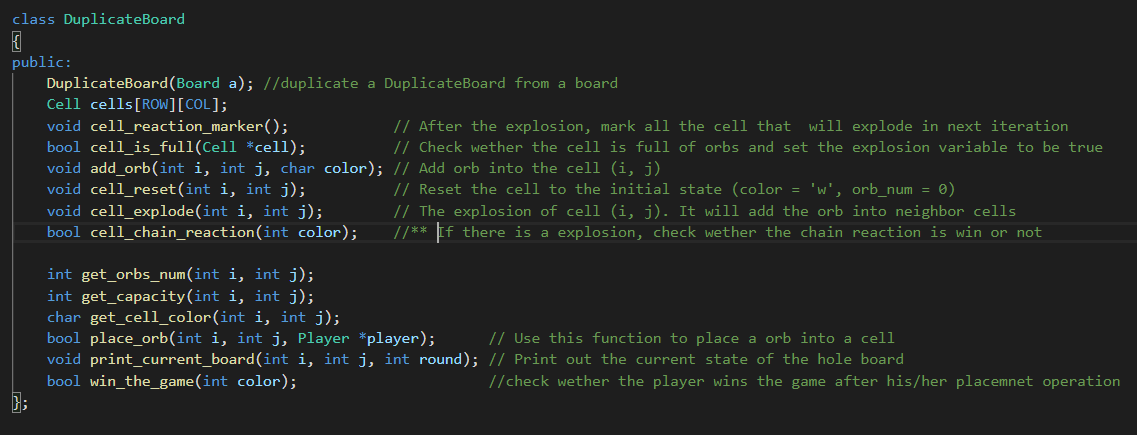
104072126齊安浩

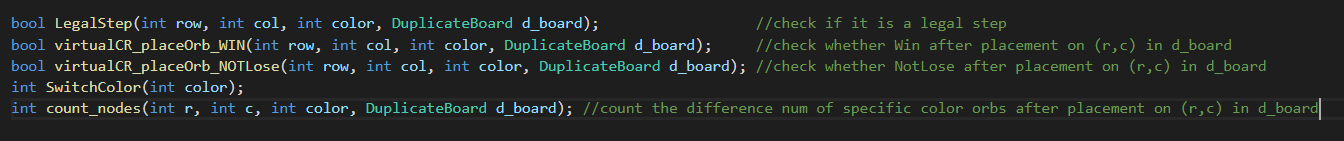
* Program flow chart



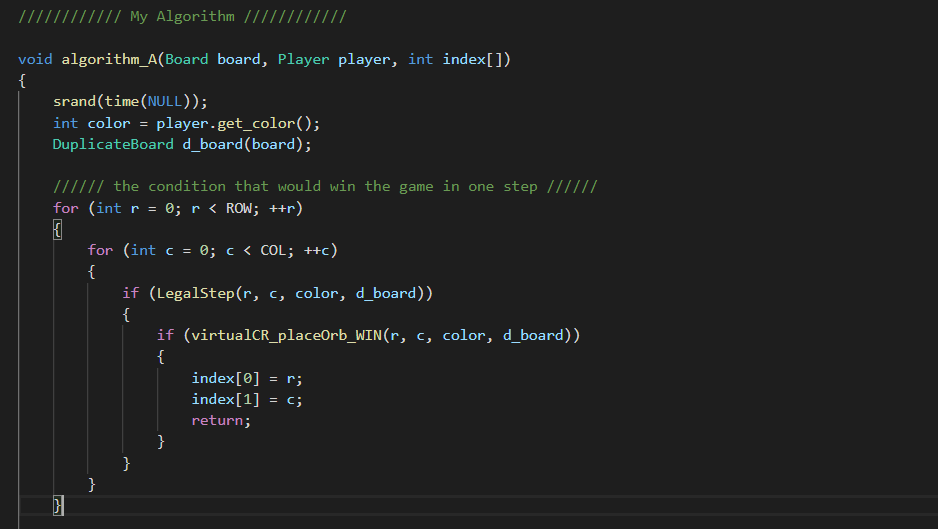
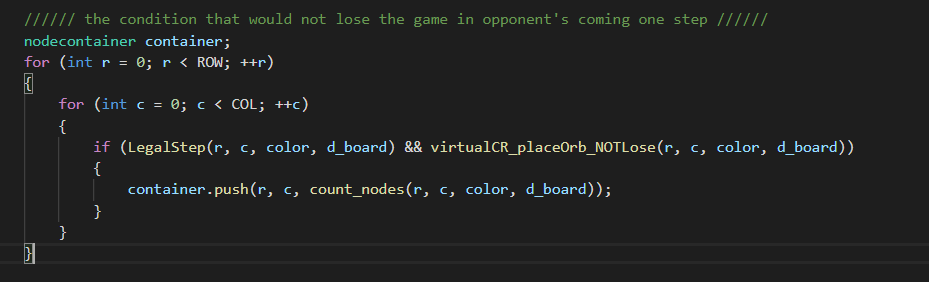
* Data structure used

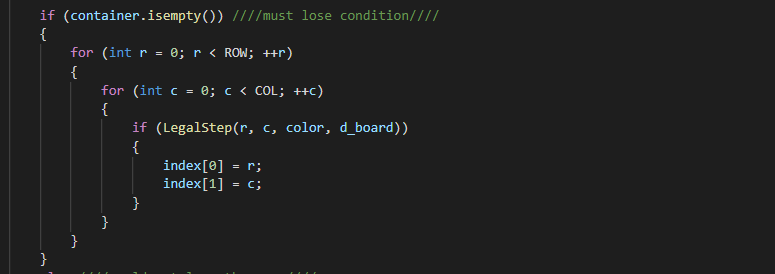
1. 宣告一個node 和container的class，用來在紀錄每回合預測之可行結果。
2. 宣告一個類似board的class: DuplicateBoard，並將權限設定為public，讓之後的操作能夠比較客製化。

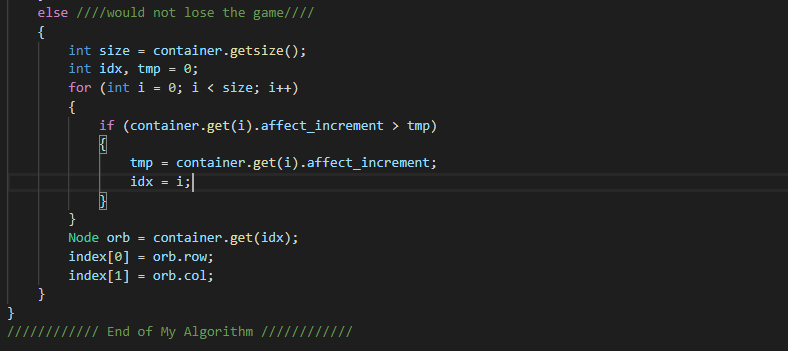


1. 自己宣告的五個function，之後會有詳細的解釋。

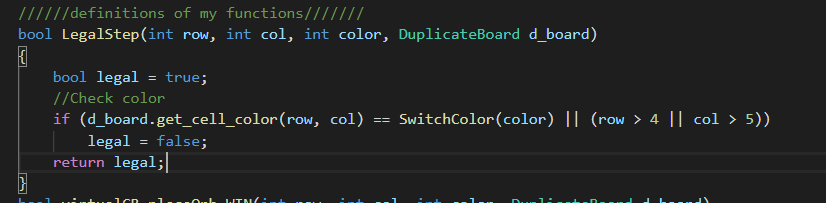
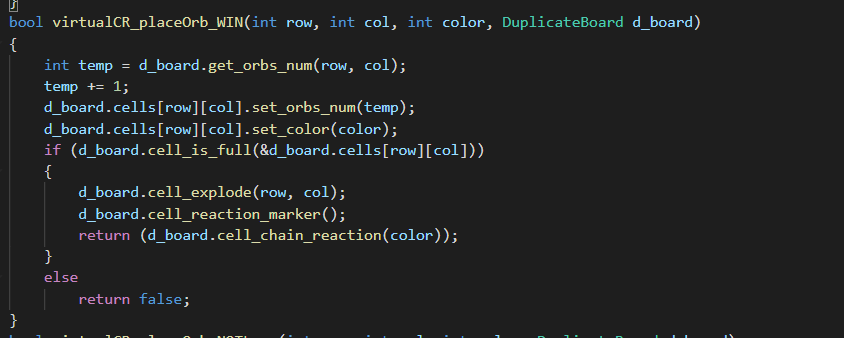
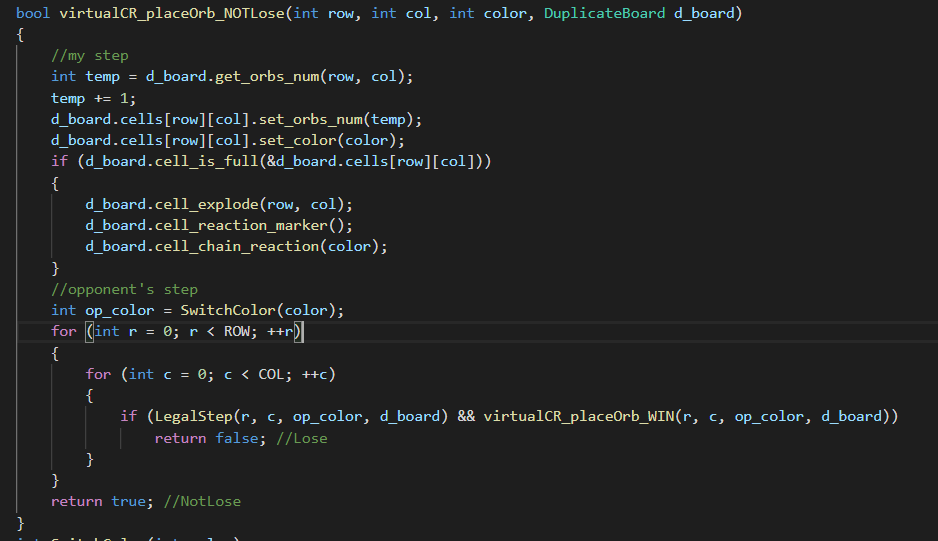
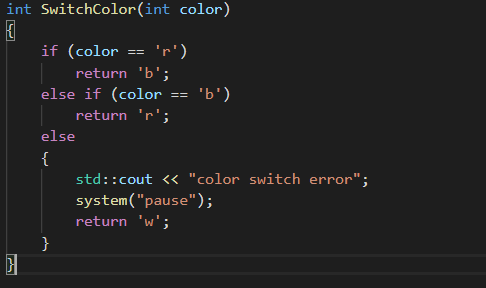
* Algorithm description

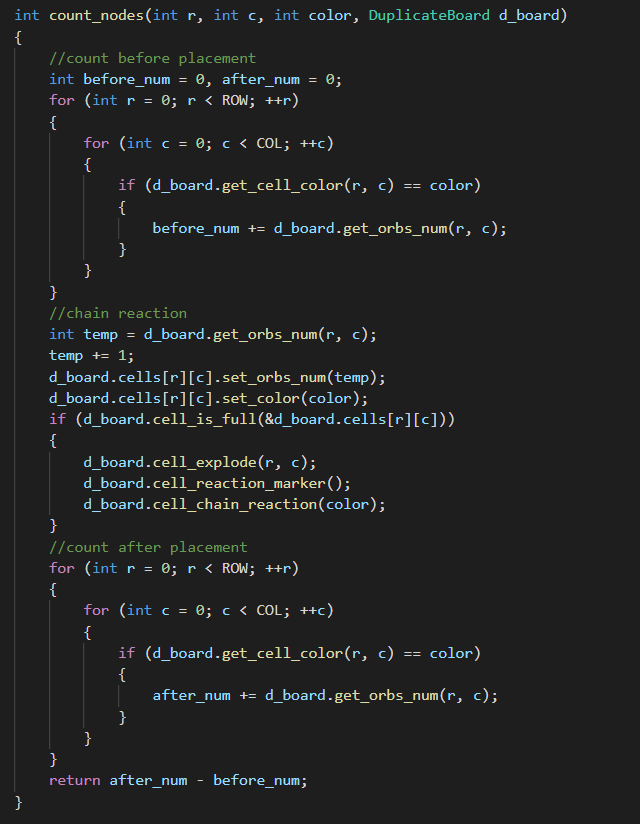
1. 先Duplicate一個Board，較d\_board，然後檢查board中每個點是否可以下了一步便贏，如果可以，便優先下這步，並return。
2. 假如都沒有一步便贏的狀況，那我們再判斷有沒有下了不會馬上輸的步點，如有，便將此點push進宣告的contain中。
3. 結束判斷後，假如container為空，則代表一定會輸，因此選擇一個可以下的點下就好。假如container步為空，則選cointainer中affect最多的點。





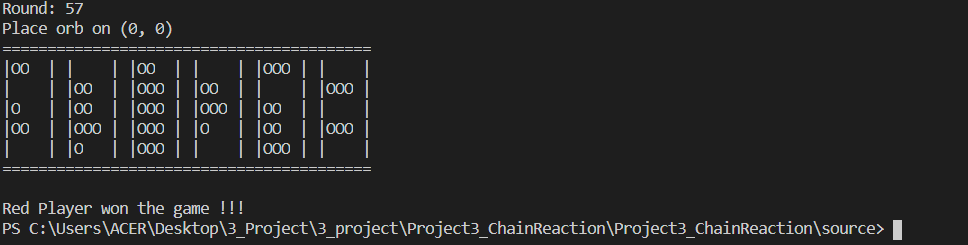
* detail description

1. LegalStep判斷該點是否可以下。
2. virtualCR\_placeOrb\_WIN判斷下了該點，經過chain reaction後，是否會勝利。
3. virtualCR\_placeOrb\_NOTLose判斷下了該點並經過chain reaction後，對手是否有辦法下了一步便能勝利。 
4. 轉換顏色Bj4><。
5. Count\_nodes計算在下了該點後，場上同顏色棋子增加的數量。此function協助algorithm在不會輸的可下棋子中，選擇能占最多地盤(棋盤中同顏色orb數)的落子。

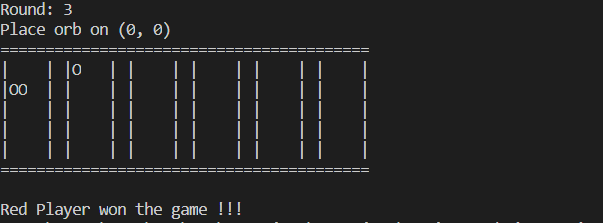


* Algorithm對抗分析

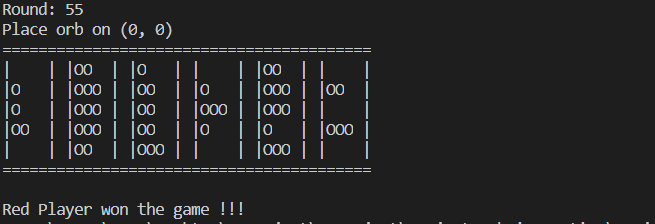
1. algorithmB



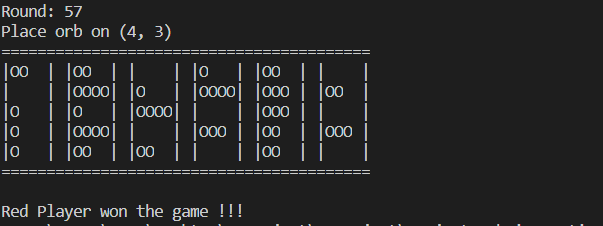
1. algorithmC



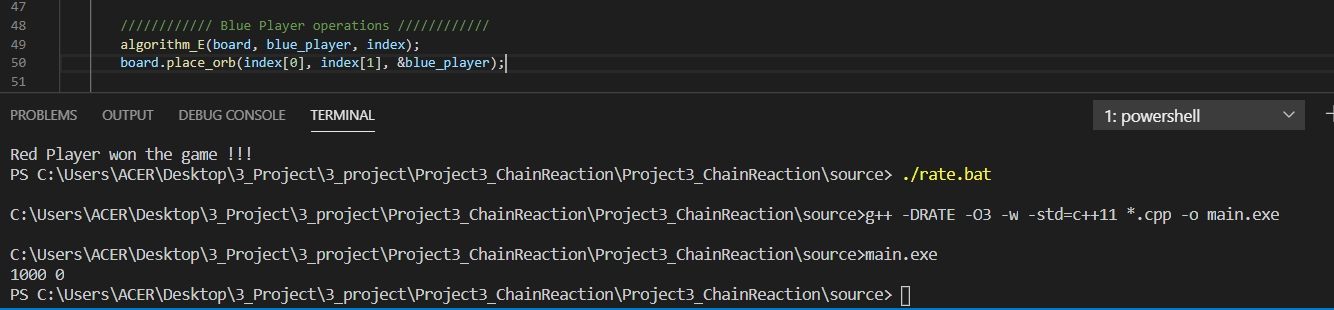
1. algorithmD



1. algorithmE



1. 總分析



用批次檔模擬了A跟E對戰1000次，結果是全勝。分析在助教的E算法中，雖用了game tree來增加預測性，但BoardEvaluator的計分方式卻不是以勝利來計分，是用占地盤來判斷，佔越多分數越高。而A算法中用簡單的方式預測落子後勝負的結果，如果會輸便不要下那點。在此邏輯下，再增加占地盤的判斷，勝率便更有效地提高了。