

Game Project

Team ID : 13

Team member : 0710006 盧可瑜 0516316 呂爾軒 0516072 洪立宇

策略

- 以下走法都保證當回合不跳過重複的格子。
- 對每一顆己方棋子模擬所有能跳的方式，分為「走一步」及「跳多步」兩種，將能動的step放進一個vector。
 - 走一步
 - 分別對於四個方位嘗試能不能走，也就是說判斷到達的格子是否為空。
 - 跳多步
 - 採用dfs，分別判斷四個方向是否有棋子能跳過及到達的格子是否為空
 - 將此走法丟進vector後，繼續呼叫dfs。
- 對於每一個step得到的新棋盤都檢查敵方最多能吃掉幾顆己方旗子，也就是程式中的dfs2和dfs3。
- 採用「估值函數」排序所有合法的移動方式。
- 選取最佳的走法，並檢查是否導致直接遊戲結束且敵方失敗，回傳此step。

估值函數

- $f(\text{step}) = (\text{己方吃子數} - \text{敵方吃子數}) * \text{eatWeight} + (\text{抵達位置相鄰己方棋數}) * \text{friendWeight} + (\text{前進的水平距離}) * \text{moveWeight} + (\text{是否抵達終點區域}) * \text{endpointWeight}$
- 權重決定方式
 - eatWeight
 - 始終設為100，為最大權重。
 - 因為此權重能避免己方棋子被吃，增加己方的得分機會，並盡可能吃掉敵方棋子，減少對方的得分機會。

- moveWeight、friendWeight
 - 前7步：moveWeight為10、friendWeight為1
 - 因為要盡量往前佔據有利位子，避免被敵方逼進壓縮發展空間。
 - 7步後：隨機將(moveWeight, friendWeight)設為(12,4)或(5,5)。
 - moveWeight是使棋子逐漸向終點移動。
 - friendWeight是使棋子不要太分散，減少被吃的機率。

					2		
						2	
		1	1	2		2	
		1	1		2		
	1	1		2			
		2	2		2		

- 以下圖為例，明顯可看出2相對1分散，被吃的可能性較高。
- endpointWeight
 - 設為-10
 - 當棋子已抵達終點，在其他條件相近的情況下，應優先移動其他未抵達終點的旗子。

檢查遊戲結束

- 為了避免當局吃掉敵方棋子後，造成敵方剩餘棋子都在終點，且分數大於等於我方的情形。
- 判斷方式如下：
 - 分別統計己方及敵方到達終點的數量。
 - 若己方>敵方，則維持原先走法。
 - 否則，選一個不會吃掉敵方且離終點近的走法向終點移動。

1							2
				2		1	
		1		1			
		1	1	1			
					1		

1							2
				1			
						1	
		1					
		1	1	1			
					1		

說明：

左圖為初始狀態，右圖為移動後的狀態，己方為1敵方為2。因為eatWeight的權重較大，因此會選擇去吃掉敵方棋子，導致平手。

很明顯地，在左圖狀態中，我方可將離終點近的旗子移向終點，這樣明顯是較好的策略。

策略選擇分析

- 選擇策略時，我們其實有考慮做min-max tree，加上 α - β pruning優化。然而，我們考慮到離終點至多有200步，也就是有200層的深度，每個狀態多的話可以有幾百種走法（因為可能可以連續跳10幾步），在5秒內是不可能用有限的記憶體做到的。再加上由於我們的對手未必和我們一樣採用所謂的最佳策略，因此使用min-max tree不能保證我們能贏。權衡之下決定不採用min-max tree。
- 這時，我們開始考慮MCTS，因為它並沒有直接考慮所有可能性，所以速度上相對較快，然而考慮到我們沒有足夠多的對手去對戰以調整UCB的參數，似乎難以保證實際對戰時的狀況。
- 在考慮計算資源後，我們想到了一些greedy的方式。以下舉一些例子。

- 將棋子從旁邊逼近，能減少被吃的機率。
- 抵達棋盤的四個角之後是無法被吃的。
- 將棋子集中成2x2之後是可以有效率且安全的移動，具體方式如右圖。
 - 從A開始移動至A'，依序是B、C、D。
 - 假設前方無阻礙的話可順利前進。

	A	D	A'	D'			
	B	C	B'	C'			

- 仔細觀察上述greedy的策略，會發現其實無法單純直接透過greedy設計出完美的走法，畢竟遇到阻礙就需要額外的判定方式，不夠具一般性，畢竟以我們的計算資源來說，無法考慮完所有可能性，且具體實現各種case的判定及因應。因此，搜尋還是必要的，greedy得到的一些性質，像是群聚、盡量向前移動、對方旗子愈少愈好，這些都是估值函數的參考依據。

- 然而，估值函數的參數也沒有足夠的資源來調整，畢竟我們的對手是未知的，沒有棋譜之類的東西可以參照，所以我們也就只是將參數調到能表現出我們希望的趨勢，再去額外判定少數例外情況，加上一些隨機性去配置參數。