## 第一個大架構

當我開始要寫這份專題時，我先看了老師說的minimax以及alpha-beta pruning，感到他確實是一個可以挑戰的目標。確立目標後，我發現這個做法需要一個評分系統，並且這套系統不能只算玩家分數，這樣子才知道這一步究竟是好是壞。因此，我把評斷標準分為兩種，一種是必要的(例如會不會撞牆、撞玩家…)，另一種就是算評分及分數的。因此，當我隨時要加入新的判斷標準時可以輕易地加入。

由於要用alpha-beta pruning，我需要先知道跟我同一層的、走其他方向的結果以及上層告訴我不要再繼續執行的準則，因此我用DFS。最初，我有準備好幾個判斷標準，但是現在看來可能最多只能2個或3個了。

會這樣是因為我在解讀minimax的內容的時候解讀錯了，誤以為所有的分支都是指我自己走出來的，那些max跟min層所得到的都是我每一步最大的評分以及最小的評分。因此，無論跟誰對戰，我總是會看到奇怪的數據。

## 第二個大架構

當我終於釐清問題原來是那樣之後，時間剩下2天。

在這個過程中，我也有跟其他同學交流，發現BFS可以獲得一個不錯的成績。但是，我仔細想想，DFS憑藉著他可以走回頭路的特性，只要步伐數足夠就會比BFS好。而且在原本要跑錯誤的alpha-beta pruning的情況下我的電腦依然可以跑到我本身走了7步的情況。

因此，我針對前一個大架構的內容進行刪修，去除minimax的部分，變成只剩只考慮我的情況的DFS。在這個情況下，我可以探索到的步伐來到了11步。

DFS跟BFS最大的不同就是偵測他可以走幾步的難易度。BFS可以一直跑，直到發現時間快到了再停止就好；反觀DFS假使沒有先算就變成要去追蹤每個執行中的個體，然後再一個一個執行，而且還有執行完後回傳值要統合的問題，顯得麻煩許多。因此，我在一切開始之前有先benchmark看他每跨一步要花多少時間。

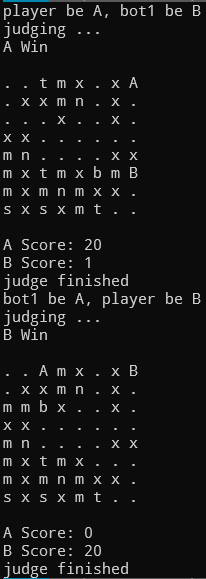
除了這個，我的程式中有對於物件的遠距而有差異。一個物件每較遠一步，他的分數就會被乘以0.8。然而，這個數字目前看來可能不是一個很好的數字。從跟助教的機器人對戰的結果可以發現我到後期會一直在原地踱步。但是人眼一看明明就還有可以吃完繼續加分的蘑菇，甚至是星星。

不過很有趣的是，當我把這個0.8改為0.82後，他「近視」的問題就被解決了，他就不會在遠方還有好多寶物的情況下還在原地踱步了。

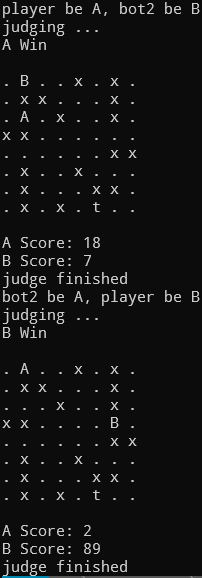
## 與助教機器人對戰的結果

3支機器人皆有戰勝

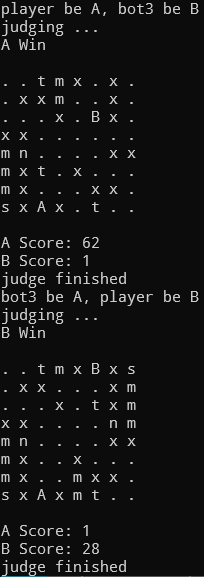
### 機器人1



### 機器人2



### 機器人3



## 假如仍有時間希望加入的內容

第一個就是多執行緒的部分。畢竟在這個算法中，4個方向各自如何事完全沒有關係的，是最後4個方向都知道後的整合才是重點。因此，這個方法可以加速不少。

第二個則是要考慮我會不會被關在死巷以及能不能把對方關在死巷的問題。畢竟，無論分數多高，只要有這樣的情況發生，勝負就非常明顯了。

第三個則是弄出真正的minimax、alpha-beta pruning出來。但是，這個難度真的會不知道要往上幾層樓呀！

## 其他心得、建議

這次的專題實在是很有趣也很實用，畢竟日常生活中很多的東西都要我們評斷在一個情況下，我們做出甚麼動作所得到的益處最大。老師介紹的minimax、alpha-beta pruning雖然在這次僅有1秒鐘而且沒有GPU的幫助下很難達成，但是我感覺得出來這以後很可能就是我發展中的一項先備知識。感謝老師介紹給我們！

另外，在這次專案中我也看到在撰寫程式之前，設計一個良好的架構的重要性。這次萬一我沒有設計好，相信我一定很難可以在短時間內轉換到我第二個架構。