AI-HW3 40647001S 郭奕成

如何執行：使用瀏覽器開啟 index.html 檔案即可。

硬體：

CPU：AMD R5-2600X
GPU：AMD RX580 4G
RAM：DDR4 8Gx2

軟體：

OS：Windows 10 home
Browers：Chrome 版本 83.0.4103.106 (正式版本) (64 位元)
Firefox 版本76.0.1 (64 位元)
Edge 版本 83.0.478.54 (官方組建) (64 位元)

開發軟體版本：

Languages：HTML, CSS, JavaScript ES9
Framworks and Libraries：Vuejs(APP), Vuetify(UI), lodash(deep copy)

為何選擇這樣的規格：一般家用PC的規格

連絡電話：0975-389-568

如何執行：使用瀏覽器開啟 index.html 之後即會看到棋盤遊戲，玩家為黑方，先手，電腦為紅方，後手，當輪到玩家下棋時會在可以移動的棋子上看到黃色圈圈的提示，當滑鼠移動到上方時會有點擊提示，用滑鼠點選一個可以移動的棋子後會再顯示這個棋子可以移動到的格子，也一樣有相關的操作提示，如果有連跳發生，則會在棋子移動過去之後才顯示下一步的選擇。當遊戲結束、勝負出來之後會顯示一個 Modal 顯示比賽結果。如果要再開啟新遊戲刷新頁面即可重新開始。

資料結構： 8x8 2D int array，0 is empty，1 is black pawn，2 is black king，-1 is red pawn，-2 is red king。

技術(操練要項)：使用alphabeta pruning search。heuristic function計算盤面的雙方各自的pawns, kings, back row (防止敵人可以變成king), middle box(在棋盤中間的棋子比較有操作彈性而且可以限制敵人的一部分行動), middle 2 rows but not box(理由跟上一個差不多), can be taken this turn(會被吃掉要扣分), are protected(因為在邊界或有友軍或敵方卡位而敵人沒辦法吃掉的)，這幾個規則來計算一個盤面的分數，分數越大則對黑方越有利、對紅方越不利，分數越小則對紅方越有利、對黑方越不利。

測試時的表現：展開的深度是設定在4，如果設到5就會明顯感覺到卡頓，電腦的效能來跑要花點時間，使用體驗不是很好，感覺設在4的效果就滿不錯的，在上半局面的表現都滿好的，會積極吃子、保護自己的棋子不被吃，身為玩家在一開始還滿難下的，可能因為審局函式的規則還是不夠豐富，到了後半局面，棋子數量都比較少的時候，攻勢就變弱了，玩家在這時候就比較好操作，如果撐過前半段AI的攻勢，後來就容易反敗為勝。

功能及優點：就是有方便又美觀的使用者介面，為了這個介面架構就想很久、實作也一直碰到困難，做的頭超痛QQ。

參考文獻及網站及參考了哪些部分用於實作中：
kevingregor/Checkers: Analysis of optimal heuristic evaluation function for Checkers AI：參考他實驗出來的heuristic evaluation function。
Alpha-beta pruning：參考alphabeta pruning的維基百科上面的psudo code。

碰到的一些狀況及困難：
從最一開始挑選要什麼棋類遊戲的時候，雖然是推薦做五子棋，但我對這個遊戲之前沒啥研究，稍微看了一下之後也沒啥感覺，對於架構的設計沒啥想法，所以就去查了一下看看還有什麼其他的棋類遊戲，又希望規則不要太複雜的，不然程式碼真正在寫adversay search的部分應該會很小，都是在寫遊戲規則相關的程式碼。後來看到這個英國跳棋感覺還滿不錯的，腦內也有浮現一個簡單的UI跟程式架構，這個時間點就是麻煩的開始....，我小看這個遊戲了，規則看起來少少的，每個又都短短的，但開始寫code之後才發現超麻煩，要寫的code比想像中的還要多很多，尤其是連跳最麻煩，各種情況要考慮，寫成程式碼就是又臭又長又雜，而且之前沒有寫過這樣份量的javascript，我對這個語言也沒有非常熟悉，重點是很久沒寫了，很多東西都忘的差不多了，要一直查文件、文章、別人的發問等等，開發的速度就有點慢。終於把核心類別的部分寫的差不多了，稍微測試一下、跑一下之後，開始要來弄使用者介面了，因為我已經有幾次開發網頁的經驗了，想說應該不難吧，可以很快就弄出來吧，結果也是又多又雜的，很煩，想了很多個架構，都各有各的麻煩地方，最後選擇使用Vuejs的指令v-for，用雙層的來產生出二維的棋盤跟按鈕，他們的class, style, click event則用js的變數來綁定，因為棋盤的擺放看似有規則，但要用for loop跟if else什麼寫成程式碼還是一樣很麻煩，所以乾脆就直接存物件的陣列的陣列，物件裡直接包資料，我又懶得手打跟複製貼上，所以用python寫了一支叫做tmp.py的程式來產生這個pieceData.js的檔案，寫一寫之後都會發現當初資料設計的少了什麼東西或多了什麼東西，這時候就直接改python的程式就好，只有第一次寫的時候麻煩一點但之後方便很多。再來就是發現我希望有點MVC架構的問題，view controller跟model操作的方法還有溝通的管道跟一開始架構想還要多很多啊，之前寫的根本就沒啥能方便操作跟溝通的，少一堆東西，又再補了一堆程式碼，又一邊產生bug跟debug，像是king在給玩家選擇的時候不會顯示提示，是因為資料的class的值沒考慮周全，這個就小問題，處理起來用點技巧的話就沒有到很麻煩，另一個問題是如果吃子後跳到後排而變成king的時候，那個應該被吃掉的棋子卻還在，就是之前寫的board class有問題，這個問題一開始還沒發現問題點，想說怎麼玩起來它的行為都不一定跟預期的一樣，是在測試了很多次之後才發現問題是這樣，回去看之前的程式碼，用腦袋稍微跑過一次程式碼之後發現有問題的片段，一段程式碼執行順序的問題。雖然遇到很多問題，開發的時候頭很痛，但也因此學了很多，現在程式寫完了之後來看，真的是滿滿的收穫跟小小的成就感。