人工智慧導論 hw1

F74056166 方嘉祥

1.

使用窮舉法，以 fopen 開啟檔案之後，由讀到的 x, y 範圍，從最小值(範例是-60, 30) 到最大值(範例是 60, 70)為止，一路呼叫 func()，並以一個變數儲存出現過的最小值，變數初始化是以一個非常巨大的值來避免誤判

2.

Hill climbing 使用 stepsize 為 1

	呼叫次數		呼叫次數		呼叫次數
暴力解	12221	[-60,18]	196	[25,54]	200
[33,63]	268	[-24,56]	204	[38,13]	88
[-58,-9]	232	[-31,29]	124	[-49,10]	120
[-53,41]	260	[19,-11]	92		

3.a.

執行次數

步數 / 位置	1	2	4	8	16
[33,63]	268	140	68	36	20
[-58,-9]	232	120	64	32	16
[-53,41]	260	132	68	36	20
[-60,18]	196	100	52	28	16
[-24,56]	204	104	56	28	16
[-31,29]	124	64	36	16	12
[19,-11]	92	48	24	16	8
[25,54]	200	104	52	28	20
[38,13]	88	44	28	12	8
[-49,10]	120	64	32	20	12

執行結果

步數 / 位置	1	2	4	8	16
[33,63]	-30.010	-29.424	-29.412	-29.412	-29.412
[-58,-9]	-20.010	-19.816	-19.056	-17.568	-17.568
[-53,41]	-20.010	-19.626	-19.608	-19.608	-19.608

[-60,18]	-20.010	-20.010	-20.010	-20.010	-20.010
[-24,56]	-30.010	-30.010	-28.845	-28.828	-28.828
[-31,29]	-20.010	-19.626	-19.614	-16.714	-16.714
[19,-11]	-10.010	-9.824	-9.812	-9.093	-9.093
[25,54]	-30.010	-29.714	-29.711	-29.711	-29.711
[38,13]	-10.010	-9.915	-9.527	-8.794	-8.794
[-49,10]	-20.010	-19.816	-19.808	-18.309	-18.309

3.b.

Hill climbing 雖然是比上方的暴力法還要來的有效率許多，但是其還是會遇到局部最低點的問題，比如像是[19,-11]和[38,13]，不論使用的 stepsize，他的結果都偏離了使用暴力解獲得的最小值，不過還是有像是[33,63]能夠接近最佳解的結果(stepsize 1 為最佳值)

不過就執行次數來看，兩個方法的差距非常巨大，12221 對上 268，就這點上，hill climbing 算是符合我們要求上的速度優化，要在此之上繼續優化準確度的話，我們可以嘗試看看諸如模擬退火演算法或是大洪水演算法。

4.

程式優化部分，我在撰寫暴力方法的時候，每一次 run 都要花費不短的時間，而 func 函式佔了絕大部分的時間，如果能使用類似於 multi thread 的方法，應該可以縮短運行的時間

另外，在我撰寫程式的過程中，我的 anaconda 內的 spyder 不知為何，每跑一次程式之後都需要 restart kernel，出現

SystemError: ..\Objects\moduleobject.c:449: bad argument to internal function 錯誤訊息，如果助教在改作業的過程中也遇到這種情況，可以試試 restart kernel