INDRODUCTION TO ARTIFICIAL INTELLIGENCE
Homework #1

F54051201

林郁翔

一、 暴力搜尋法

1. 指定一個變數 z_min 存放最小值，並將其設為 9999
2. 以兩層迴圈，分別遍歷 x, y 的範圍
3. 範圍內的每一個 x,y pair 都會呼叫一次 func(x,y)
4. 若是此次的 func(x,y) 比 z_min 小，則將其替換為 z_min
5. 結束迴圈後，返回最小值 z_min，並以 round 函數做四捨五入

二、 暴力解與 Hill Climbing 的執行次數比較

x = -60~60, y = -30~70

暴力解次數	Hill Climbing 平均次數
12000	117.7

三、 Hill Climbing 不同 step size 的結果

step = 1					
init point	執行次數	result	init point	執行次數	result
(33,63)	137	-30.010	(-31,29)	81	-20.010
(-58,-9)	157	-20.010	(19,-11)	89	-10.010
(-53,41)	137	-20.010	(25,54)	105	-30.010
(-60,18)	164	-20.010	(38,13)	77	-10.010
(-24,56)	109	-30.010	(-49,10)	121	-20.010
step = 2					
init point	執行次數	result	init point	執行次數	result
(33,63)	73	-29.424	(-31,29)	41	-19.626
(-58,-9)	81	-19.816	(19,-11)	49	-9.824
(-53,41)	73	-19.626	(25,54)	57	-29.714
(-60,18)	84	-20.010	(38,13)	41	-9.915
(-24,56)	57	-30.010	(-49,10)	65	-19.816
step = 4					
init point	執行次數	result	init point	執行次數	result
(33,63)	37	-29.412	(-31,29)	25	-19.614

(-58,-9)	44	-19.056	(19,-11)	25	-9.812
(-53,41)	37	-19.608	(25,54)	29	-29.711
(-60,18)	44	-20.010	(38,13)	25	-9.527
(-24,56)	33	-28.845	(-49,10)	33	-19.808
step =8					
init point	執行次數	result	init point	執行次數	result
(33,63)	20	-29.412	(-31,29)	17	-16.714
(-58,-9)	24	-17.568	(19,-11)	17	-9.093
(-53,41)	20	-19.608	(25,54)	17	-29.711
(-60,18)	24	-20.010	(38,13)	13	-8.794
(-24,56)	17	-28.828	(-49,10)	21	-18.309
step = 16					
init point	執行次數	result	init point	執行次數	result
(33,63)	12	-29.412	(-31,29)	9	-14.411
(-58,-9)	12	-12.753	(19,-11)	9	-7.712
(-53,41)	12	-19.608	(25,54)	13	-10.567
(-60,18)	16	-7.245	(38,13)	9	-8.794
(-24,56)	12	-11.704	(-49,10)	12	-18.309

由以上結果可以發現 step 並不是越大越好，越大的 step 雖然執行次數越少，但是結果距離最小值卻都有一定的偏差。

而且較大的 step 到達最小值時，所需要用的「距離」(step * 執行次數) 並不會比較小的 step 所用距離小，猜測這是因為走過頭後，發現另外一個方向會有更好的選擇，造成多餘的步數。

四、 觀察心得

使用暴力解時，由於需要呼叫 func() 12000 次，所以需要耗費許多時間，因此若是將過程拆成多個 thread，應該可以節省不少時間。

使用 Hill Climbing 時，由於幾乎每走一步都會呼叫 func() 判斷每個方向是否都會有更好的解，因此推測若是一開始就決定好要往哪個方向 (上下擇一、左右擇一)，並且接下來就只往那兩個方向行走，到最後發現再繼續走，值會變大時，再看看其他方向有沒有更好的解 (一開始是往上、右，這時就查看下、左)，這樣應該可以節省中間許多不必要的呼叫。