AI HW1

學號：F74056069 姓名：崔濟鵬

1. 由於本題限制x, y為int，故可以簡單的利用雙層loop來實作暴力解。

首先將x從給定範圍中的最小值開始，固定住此x值的同時，將y從給定範圍的最小值開始，依次的將x, y放入得出z，若z值小於目前儲存的最小值min，就替換掉min，固定x值走遍y值後，繼續下一個x+1，y又從最小開始走遍直至最大，最終可以得到此範圍中的最小值。

若x_min ~ x_max有m個，y_min ~ y_max有n個，則總時間複雜度為O(mn)，花費時間可觀。

2. 此部份利用助教給的範例測資來比較

測資	call func() 次數	結果
暴力解	11011	-30.010
hill climbing[33, 63]	268	-30.010
hill climbing[-58, -9]	232	-20.010
hill climbing[-53, 41]	260	-20.010
hill climbing[-60, 18]	195	-20.010
hill climbing[-24, 56]	204	-30.010
hill climbing[-31, 29]	124	-20.010
hill climbing[19, -11]	92	-10.010
hill climbing[25, 54]	200	-30.010
hill climbing[38, 13]	88	-10.010
hill climbing[-49, 10]	120	-20.010

3. 此部份利用助教給的範例測資來比較(次數/結果)

起始點	size=1	size=2	size=4	size=8	size=16
[33, 63]	268 / -30.010	140 / -29.424	68 / -29.412	35 / -29.412	19 / -29.412
[-58, -9]	232 / -20.010	120 / -19.816	63 / -19.056	31 / -17.568	15 / -12.753
[-53, 41]	260 / -20.010	132 / -19.626	68 / -19.608	35 / -19.608	19 / -19.608
[-60, 18]	195 / -20.010	99 / -20.010	51 / -20.010	27 / -20.010	15 / -7.245

[-24, 56]	204 / -30.010	104 / -30.010	56 / -28.845	28 / -28.828	15 / -11.704
[-31, 29]	124 / -20.010	64 / -19.626	36 / -19.614	16 / -16.714	12 / -14.411
[19, -11]	92 / -10.010	48 / -9.824	24 / -9.812	16 / -9.093	8 / -7.712
[25, 54]	200 / -30.010	104 / -29.714	52 / -29.711	28 / -29.711	20 / -10.567
[38, 13]	88 / -10.010	44 / -9.915	28 / -9.527	12 / -8.794	8 / -8.794
[-49, 10]	120 / -20.010	64 / -19.816	32 / -19.808	20 / -18.309	11 / -18.309

從上表可以明顯得觀察出，隨著step size的增加，call func的次數明顯的減少，但同時的，離真正的最小解也有遠離的趨勢。

可以驗證課堂上所說的，hill climbing符合local search的特性，alpha太小容易走太多步，但是可以得到較好的準確值，alpha太大雖然容易得到一個解，卻往往容易跨過那個最佳解。

而藉由不同起始點，有著不同趨勢的最小值來觀察，可以觀察出另一特性，hill climbing容易卡在local maxima，造成解非我們所想要的答案，或許就是快速之下的一些妥協。

至於要如何平衡速度跟準確度，這部份雖然我沒有實作，但是我相信可以利用如stochastic的方式，平時不全部要求最佳解來走向下一步，機率性的選擇較為普通的解，這樣容易就有機會跨過local maxima來達到真正的最佳解，也可以利用random-restart的方式，不要僅限某個起始點，透過隨機選擇起始點來進行演算，多次尋找較容易得到一個最佳解，或者可以利用退火演算法、local beam search的方式來找尋，且這些方法針對不同function或分佈的時候，可能都得經由多次實驗後，才能或取一個比較好的特化參數跟解答。

4. 針對以上的觀察，可以得出如果面對一個未知的function，要得出最佳解，或許暴力是一個辦法，但是在現在追求效率的社會下，並不是一個最好的解法，浪費時間跟精力，小題目可能還好，未來後面的作業就不可能用如此暴力解來處理。

故有時候訴諸一些較為聰明的方法，雖然不一定100%保證能得到一個最佳的答案，但是可以透過多次實驗，調整參數，更換演算法，直到比較能夠良好的處理此數據分佈，從中得到平均下最好的結果。也有些時候，理論上的某些演算法很棒，結果必定漂亮，但是取捨於目前硬體限制等，我們無法實作如此的演算法，那我們就只能選擇一個local maxima，就像hill climbing一樣，得到一個區域的最佳解，再進一步的尋找最佳解，不是追求一蹴可及，觀察數據想傳達給我們的話，或許就能夠在這些演算法中找到一個很不錯的平衡。

目前課堂為止提到的退火演算法，剛好在之前的作業研究課程中有做過相關的課題，那時也是觀察了參數跟結果的關聯性，hill climbing如果跨太大步，容易錯過點與點中間的較佳解，太過小的步數又容易浪費時間，而退火演算法還加上了初始溫度以及退火速率的參數，讓整體演算法更有可變性，更有可能得出一個更符合此function的參數值，來讓平均情況下皆能獲得最佳值，這就是演算法之間的差別跟抉擇。