Homework 2 - One Max Problem (Hill Climbing)

Exercise Problem：

使用爬山演算法解決100 bits one-max problem，transition請實作兩種方式：

1.使解往左走或往右走 0001->0010 or 0001->0000

2.隨機找其中一個bit，進行0->1, 1->0

※ 本次作業請畫出51 run的結果平均收斂圖(詳細請見公告)，並一併附於壓縮檔中，另程式流程及結果請使用.txt檔說明,包含兩個transition方法找到的最佳解為何，平均最佳解為何，並嘗試分析其原因。

Language : python

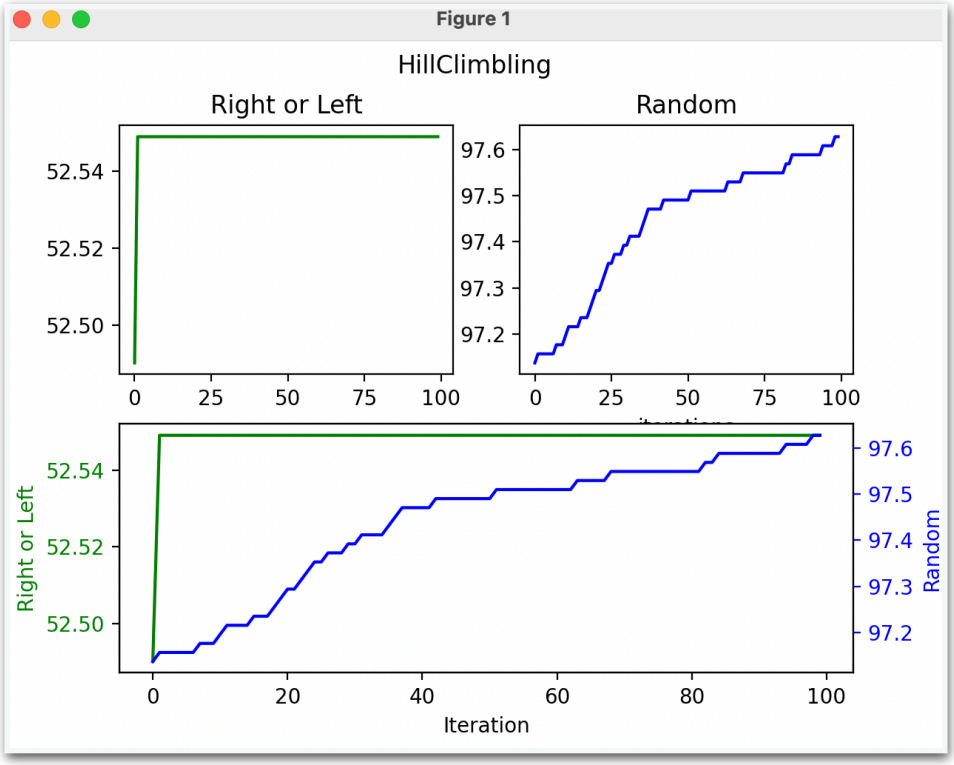
Execute : python3 **{xx.py} {algo}** {runs} {iteration} {bits len}　※ 紅字為必填，其他無填寫則為預設值

Eg. python3 main.py HC 51 100 100

Result :

因為『往左或往右』容易卡在區域最佳解（avg: 52）無法繼續比對其他解，比起『隨機翻轉』能更容易跳脫，所以得到的解較佳（avg: 97）

```
lana@lanadeMacBook-Pro 啟發式課程 % python3 main.py HC 51 100 100
————————————————————————————
【 往左/右走 】
在第 33 回有最佳解： 95 ，平均最佳解： 52.54
【 隨機左右移 】
在第 5 回有最佳解： 99 ，平均最佳解： 97.62
————————————————————————————
```



Code Description :

① 📁啟發式課程/main.py

讀取command指令，並執行相關程式

```
def load_para(algo, runs, iter, bits):
    varibles.ALGO = algo
    varibles.RUNS = int(runs)
    varibles.ITER = int(iter)
    varibles.BIT_LEN = int(bits)

if __name__ == '__main__':
    load_para(sys.argv[1], sys.argv[2], sys.argv[3], sys.argv[4])
    varibles.initialize()
    if(sys.argv[1] == 'HC'):
        HW2.HW2_main()
```

② 📁啟發式課程/HW2/HW.py — HW2_main()

for : 執行variable.RUNS次（回數）； while : 每次做variable.ITER 次迭代。

並儲存結果進行比較 → for 算平均＆取最佳

```
def HW2_main():
    global RL_list, Rand_list, rl_best_sol, rand_best_sol, history
    global rl_temp_history, rand_temp_history
    rl_total_best = 0 ; rand_total_best = 0 ; rl_best_run = 0 ; rand_best_run = 0
    rl_sum_list = rand_sum_list = [0] * varibles.BIT_LEN

    for i in range(varibles.RUNS):
        rl_temp_history = [] ; rand_temp_history = [] #清空
        # ------ random一組解 ------
        one_nums = random.randint(0, varibles.BIT_LEN)
        rl_best_sol = one_nums
        zero_nums = varibles.BIT_LEN - one_nums
        RL_list = [0] * zero_nums + [1] * one_nums
        random.shuffle(RL_list)
        Rand_list = RL_list.copy()

        j = 0
        while j < varibles.ITER:
            temp_best = Hill_Climbling()
            rand_temp_history.append(temp_best[1])
            rl_temp_history.append(temp_best[0])
            j += 1
        # ------ 取代＆做平均計算用 ------
        rl_sum_list = np.array(rl_sum_list) + np.array(rl_temp_history)
        rand_sum_list = np.array(rand_sum_list) + np.array(rand_temp_history)
        if temp_best[0] > rl_total_best:
            rl_total_best = temp_best[0]
            history[0] = rl_temp_history
            rl_best_run = i
        if temp_best[1] > rand_total_best:
            rand_total_best = temp_best[1]
            history[1] = rl_temp_history
            rand_best_run = i
    rl_sum_list = np.divide(rl_sum_list, varibles.RUNS)
    rand_sum_list = np.divide(rand_sum_list, varibles.RUNS)
```

③ 啟發式課程/HW2/HW.py — Hill_Climbling()

執行爬山演算法的主程式，分為『往左或往右』&『隨機翻轉』兩部分。

```python
def Hill_Climbling():
    global RL_list, Rand_list, rl_best_sol, rand_best_sol
    # ------ 往左或往右 ------
    direction = random.randint(0, 1) # 0:left; 1:right
    rl_temp_list = RL_list.copy()
    if direction: #+1
        temp_dec = functions.BinaryList_to_Dec(rl_temp_list) + 1
    else: #-1
        temp_dec = functions.BinaryList_to_Dec(rl_temp_list) - 1
    #計算並取代
    rl_temp_list = functions.Dec_to_BinaryList(temp_dec)
    rl_temp_sol = functions.Count_Sol(rl_temp_list)
    if rl_temp_sol > rl_best_sol:
        RL_list = rl_temp_list #取代成新list
        rl_best_sol = rl_temp_sol #取代新成解

    # ------ 隨機翻轉 ------
    rand_temp_list = Rand_list.copy()
    position = random.randint(0, varibles.BIT_LEN - 1) #隨機更改的位置
    rand_temp_list[position] = not rand_temp_list[position] #翻轉位元
    #計算解
    rand_temp_sol = functions.Count_Sol(rand_temp_list)
    if rand_temp_sol > rand_best_sol:
        Rand_list = rand_temp_list #取代成新list
        rand_best_sol = rand_temp_sol #取代新成解

    return [rl_best_sol, rand_best_sol]
```