Homework 1 - One Max Problem

Exercise Problem：

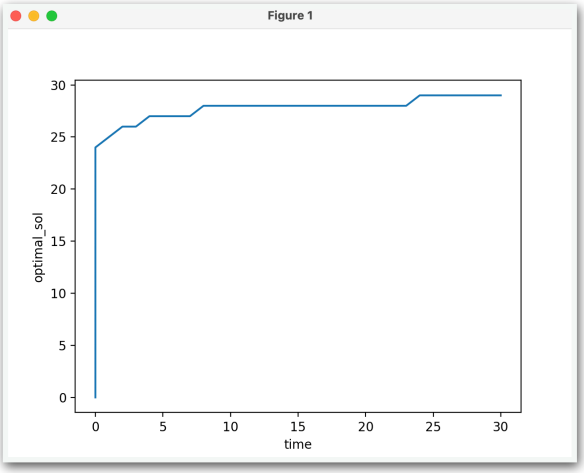
使用窮舉法(Exhaustive Search)觀察在半小時內one-max problem (100bits) 可以跑到幾個bits，程式流程及結果請使用txt檔說明,包含找到最佳解為何

Language : python

Result :

經30分後能找到的最佳解為29

```
---------------
Time = 28 分鐘
Optimal Solution = 29
---------------
Time = 29 分鐘
Optimal Solution = 29
---------------
Time = 29 分鐘
Optimal Solution = 29
---------------
Time = 30 分鐘
Optimal Solution = 29
---------------
```



Code Description :

① main

使用兩個thread（t1, t2），分別執行「定時執行檢查時間＆print資訊」、「執行笛卡爾積和計算」

使用THREAD在於希望「定時PRINT資訊」和「計算和」可同時進行，並不互相干擾

```
46  if __name__ == '__main__':
47      start = time.time() #開始算時間
48      print_now_optimal() #定時執行檢查時間＆print資訊
49
50      #以thread方式執行計算和（最佳解）
51      t2 = threading.Timer(0,cal_optimal)
52      t2.start()
53      t2.join()
54
55      #畫圖用
56      plt.xlabel('time')
57      plt.ylabel('optimal_sol')
58      plt.plot(history[0], history[1])
59      plt.show()
60
```

② cal_optimal()

執行笛卡爾積和計算(t1)。使用itertools求笛卡爾積，並依序求解

```
15    def cal_optimal():
16        global best_sol
17        global end
18        for cartesain_product in itertools.product([0,1], repeat=BIT_LEN):
19            if(not end): #觀察時間未到
20                temp_cnt = 0
21                for bit in cartesain_product: #計算笛卡爾積的和
22                    if bit != 0 : temp_cnt += 1
23                if temp_cnt > best_sol: best_sol = temp_cnt#取代最佳解
24            else:
25                break
```

③ print_now_optimal()

定時print資訊的函式。使用線程（此段可有可無，不影響程式主功能，為額外寫的）

```
27    def print_now_optimal():
28        global start
29        global end
30        passt = int(math.floor(time.time() - start)) #已經過時間
31        print("Time =",int(passt / 60),"分鐘\nOptimal Solution =",best_sol)
32        print('---------------')
33        #畫圖用
34        history[0].append(int(math.floor(time.time() - start) / 60))
35        history[1].append(best_sol)
36        #時間到→結束
37        if passt >= (SEARCH_TIME * 60) :
38            end = True
39            return
40        #定時執行檢查時間＆print資訊 (以thread方式)
41        t1 = threading.Timer((PRINT_TIME * 60),print_now_optimal)
42        t1.start()
```