1. 程式 github 連結：
   https://github.com/4107029020/TSP.git
2. 演算法說明
   a. Dynamic Programming
      運用 python 改寫課本的演算法

```
void travel (int n, const number W[][], index P[][],
             number& minlength)
{
  index i, j, k; number D[1..n][subset of V - {v1}];

  for (i = 2; i <= n; i++)
     D[i][∅] = W[i][1];
  for (k = 1; k <= n - 2; k++)
     for (all subsets A ⊆ V - {v1} containing k vertices)
        for (i such that i ≠ 1 and vi is not in A){
           D[i][A] = minimum (W[i][j] + D[j][A - {vj}]);
                     j:vj ∈ A
           P[i][A] = value of j that gave the minimum;
        }
  D[1][V - {v1}] = minimum (W[1][j] + D[j][V - {v1,vj}]);
                   2≤j≤n
  P[1][V - {v1}] = value of j that gave the minimum;
  minlength = D[1][V - {v1}];
}
```

```
27  #TSP
28  #Dynamic Programming
29  def travel(n, W, P):
30      D = {}
31      for i in range(n-1):
32          D[i+1, ()] = W[i+1][0]
33
34      temp_D1 = {}
35      for k in range(1, n-1):
36          comb = combinations(A, k)
37          for x in comb:
38              for i in range(1, n):
39                  if i not in list(x) :  #vi is in A
40                      for j in list(x):
41                          K = list(set(x)-{j})
42                          K.sort()
43                          temp_D1[i, tuple(x), j] = W[i][j] + D[j, tuple(K)]
44                      D[i, tuple(x)] = maxsize
45                      for j in list(x):   #get minimum
46                          if temp_D1[i, tuple(x), j] < D[i, tuple(x)]:
47                              D[i, tuple(x)] = temp_D1[i, tuple(x), j]
48                              P[i, tuple(x)] = {j}
49      D[0, tuple(A)] = maxsize
50      for j in range(1, n):
51          K = list(A-{j})
52          temp_D1[0, tuple(A), j] = W[0][j] + D[j, tuple(K)]
53      for j in range(1, n):   #get minimum
54          if temp_D1[0, tuple(A), j] < D[0, tuple(A)]:
55              D[0, tuple(A)] = temp_D1[0, tuple(A), j]
56              P[0, tuple(A)] = {j}
57      min_length = D[0, tuple(A)]
58      return D,P, min_length
```

我用 D 和 P 兩個字典來儲存資料，而 V 的子集合 A 是用 tuple 的方式儲存的，跟課本稍微不同的是，我點的編號是從 0 開始而不是 1。例如這個圖共有 4 個點，那這 4 個點分別就是 0, 1, 2, 3。在上圖叫做 travel 的 function 中，先把 A 為空集合（不經過任何點直接到 0）的路徑存入 D 中。接著就是主程式的部分，他就是在上圖第 35 到第 48 行中。其中 35 和 37 行的 for-k 迴圈和 for-x 迴圈是在把 V 的所有子集扣掉 0 這個點（V = set(range(0, num_vertex)) , A = V-{0}）存入 comb 中，而這兩個迴圈的時間複雜度為2^{n-1}，接著的 for-i 迴圈和兩個 for-j 迴圈的時間複雜度

約為 O(2n*n) = O($2n^2$)。但是因為 python 在集合中的元素超過 8 個之後就會因為編碼的關係不會按照順序排，所以在 41 到 43 中就先把 set 轉乘 list 再做 sort 後轉為 tuple 存入 D 中，而這個 sort 函數的時間複雜度為 O(nlogn)。接著第 45 行的 for-j 迴圈就是在取得路徑最短者。因此這個 travel function 的時間複雜度就是 $2^{n-1} \times 2n^2 \times nlogn = 2^{n-1} \times 2n^3 \times logn$跟網路上查到的$\Theta(2^n \times n^2)$是差不多的。

```
60    def getPath(n, P):
61        path = []
62        k = {0}
63        B = V
64        path.append(list(k)[0])
65        for i in range(n-1):
66            B = B - k
67            temp = list(B)
68            temp.sort()
69            k = P[list(k)[0], tuple(temp)]
70            path.append(list(k)[0])
71        path.append(0)
72        return path
```

接著我又做了一個 getPath 的 function，把剛剛 travel function 算出來的 P 丟進去後就可以跑出他的路徑。原理跟 travel 這個 function 差不多。

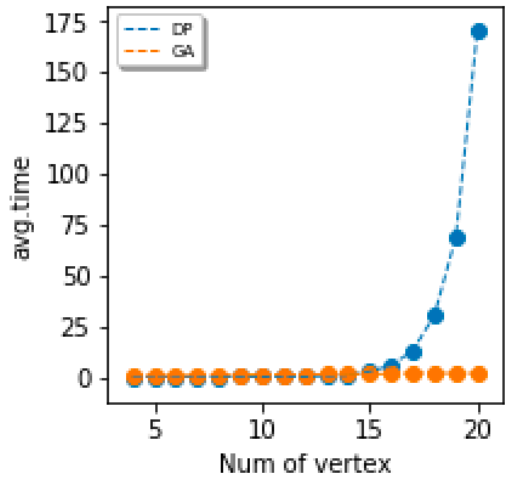
b. Genetic Algorithm

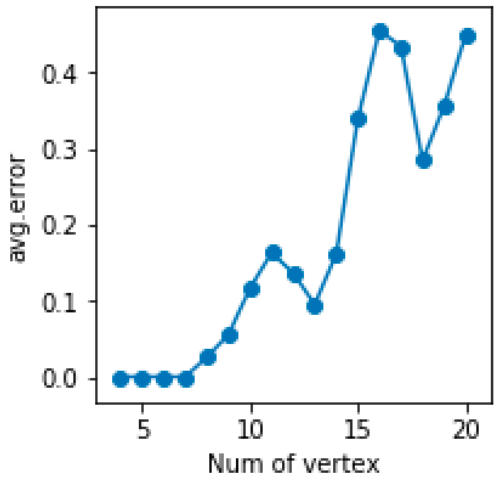
基因演算法求 TSP，程式碼改寫自這篇文章：

https://ithelp.ithome.com.tw/articles/10211706

用每個點代表不同的基因(Gene)，基因跟基因連起來就成了染色體(Chromosome)，不同的染色體就形成了一個族群(Population)。因此在 TSP 問題中，用每個點代表基因，路徑表示染色體，很多個路徑表示族群。而這些路徑的好壞取決於基因的交配(Crossover)和突變(Mutation)的策略。因此在設定時設定 populations=100 表示由 100 個不同路徑組成一個群體，mutate_percent=0.01 表示一個群體中有 1%是突變來的。elite_save_percent=0.1 表示設定最短的 10%路徑視為菁英，使這些好的基因可以流傳下去。

3. 兩演算法效能比較圖





[0.00018, 0.00028, 0.00052, 0.00162, 0.00408, 0.0099, 0.02681, 0.14304, 0.15806, 0.46045, 0.87549, 2.57575, 5.56241, 12.86247, 30.08654, 68.49355, 170.26966]
[0.70634, 0.63897, 0.69035, 0.82185, 0.89276, 0.9005, 1.05663, 1.05395, 1.19697, 1.28281, 1.45014, 1.34669, 1.53194, 1.76435, 1.7783, 1.66928, 1.77019]
[0.0, 0.0, 0.0, 0.0, 0.02579, 0.05537, 0.1162, 0.16289, 0.13645, 0.09494, 0.16162, 0.33867, 0.4548, 0.4337, 0.28412, 0.35584, 0.44977]

左上圖對應到第一和第二列資料，第一列資料是 Dynamic Programming(DP)在不同的頂點數下，執行的平均時間，第二列則是 Genetic Algorithm(GA)的。由圖中可以知道在頂點數小的情況下，DP 的執行時間是小於 GA 的，但大概在點數大於 15 後 DP 的執行速度驟增，在 20 個點時執行時間大概要近 3 分鐘。相反地，GA 的平均執行時間在 20 個點內都低於 2 秒。

接下來，右上圖對應到第三列資料，那就是 GA 算出來的最短路徑和實際比起來的 error 值。公式為：|計算出的 TSP(GA)的 Weight 和 - 最佳 TSP(DP)的 weight 和|/|最佳 TSP(DP)的 weight 和|。由圖中可以看出，當點數越多，計算出來的相對誤差就越高。