Homework 2 F74109016 大四 葉惟欣

Main.py

我有多寫一個 main.py，裡面可以創建 巡迴演唱的地點數為整數個點，且巡迴演唱的距離為整數 0~60 的地圖，但要設定參數多少個整數個點。同時 main.py 裡面也可以用來讀取地圖矩陣的文字檔，並將地圖轉換為 List(List(int))，再呼叫 BF.py 裡面的 BF 函數傳遞地圖作為輸入參數，其中實做的方式為 DFS，此外也有用排列組合的 test 函數用來做驗證。SA.py 裡面的 SA 函數的輸入參數，同時可再 main.py 裡面設定調整 Simulated Annealing 過程的參數。

Section 1. Coding problem 1. Brute Force (暴力法) :

在第一個方法中我用了 DFS 深度優先搜尋。將所有路徑都探索了一遍。由於深度優先搜尋不會改變起點。因此我用了一個迴圈，去將所有的城市做為起點都跑了一次。題目要求要回到起點，在每一次拜訪所有的城市後，我都會加入最終拜訪的城市折返回起點的距離作為總旅程距離。當所有的路徑排列組合都求得總距離後，得到最短總旅程距離，並回傳最短路徑。

驗證方法 ： 同時因為此為拜訪所有節點，所以其實可以透過排列組合而得到，我將所有城市排列組合後的路徑都去算過總旅程距離。最後得到最短總旅程距離，並回傳最短路徑。作為驗證程序。

Coding problem 2. Simulated Annealing (模擬退火法) :

程式會輸入矩陣，而後，矩陣會先隨機生成一個經過每個城市的路徑，然後固定起點後。再透過交換路徑的中兩個城市的經過順序後，不斷的更新經過所有程式後折返的總距離，來比較是否有比最好的路徑短，如果有則直接跟新，沒有的話則透過機率來看是否需要更新。機率的算法為 exponential (原本的距離 -更新後的總距離 /t) ，而這裡更新的總距離一定會比原先的原本的距離還大，因為如果比較小的話就會直接更新。如果一定會比原本的距離大，如果大越多，exponential 就越往無限小，這樣由這條路徑更新後續的路徑機率更小，反之，如果大一點點，那 expoential 就會接近 0，這樣往後由這條路徑更新後續的機率會接近 1。這樣可以避免陷入 local optimal。

而溫度的變化是不斷降溫的概念。溫度變比較小，exponential(原本的距離 -更新後的總距離(小於 0 的值) /t) 就會較小，因為原本的距離-更新後的總距離一定是負的值，負的值除上很小的正數會比起除上較大的正數更遠離 0。而這樣的概念代表當 t 很大時，也就是除上較大的正數會相對更接近 0，這樣子更新的機率值就會接近 1。

這個溫度變化的精神就是在模擬退火的過程，溫度較高的時候，更新的機率會比較大，就如同分子在高溫的時候會頻繁運動，且運動速度會比較快，以不穩定的方式活動。而當分子在低溫的時候就會比較溫度，也不會頻繁的去改變當前的最優解。這讓找尋最優路徑的過程不會一開始就掉入區域最優解，而是有很高的機率離開區域最優解，去尋找可能的全域最優解。

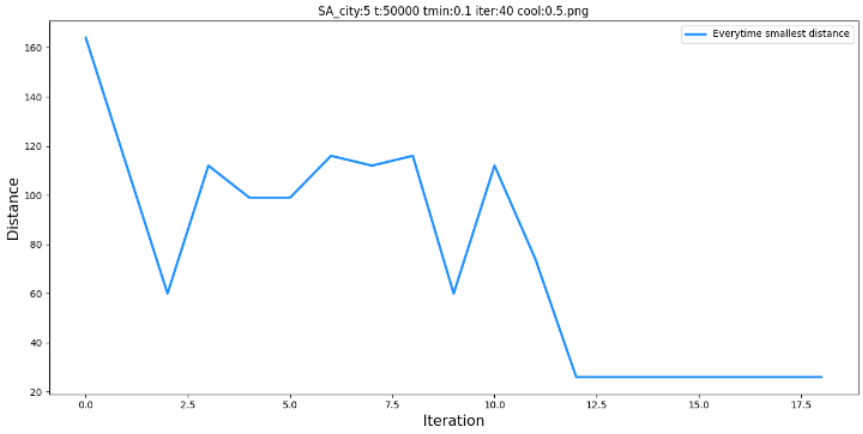
而 SA.py 裡面也有可以視覺化的程式，能夠把迭代的次數以及每次迭代的最優路程距離化成折線圖。同時也會畫出附圖，將拜訪路程畫出來。

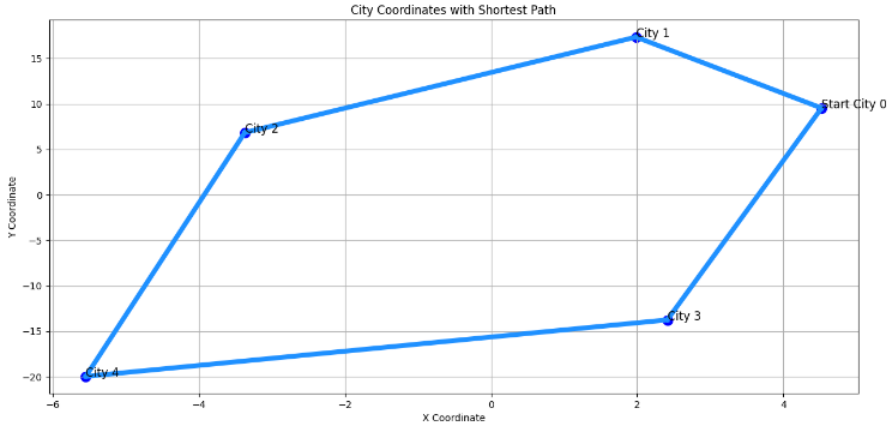
Section 3.

實作過程中觀察到什麼？如改變什麼參數可以獲得什麼結果，可以視覺化結果。

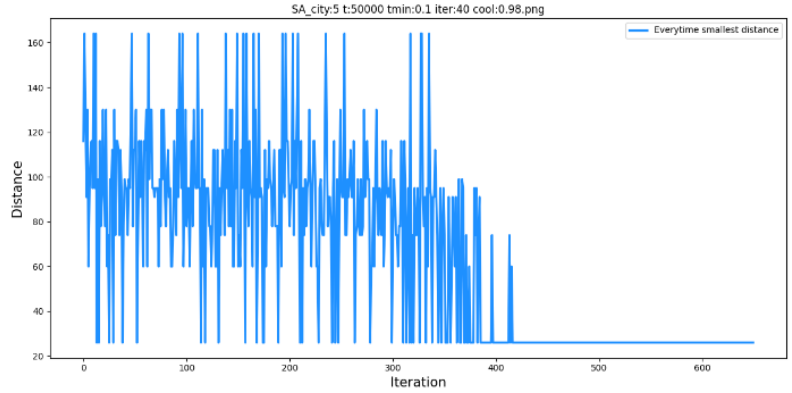
我觀察到參數 Temperature 很高的話就是有很高的機率會換當前的路徑，所以可以看到曲線圖都是波折比較多的，也相當於比較容易跳出區域最優解。而 coolrate 的改變可以決定要多快收斂起來。可能有時候嘗試 100 次就夠了，那 coolrate 就可以設較接近 0，如果要嘗試很多次才足以，那就要設接近 1。

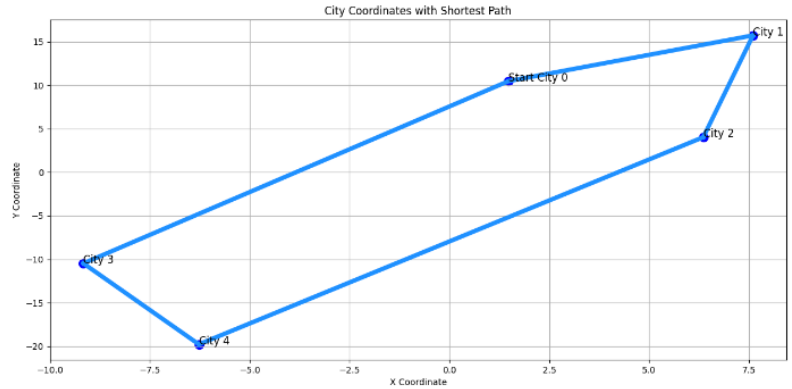
Initial Temperature : 50000 , temperature min : 0.1 ,Iteration : 40 ,coolrate = 0.5



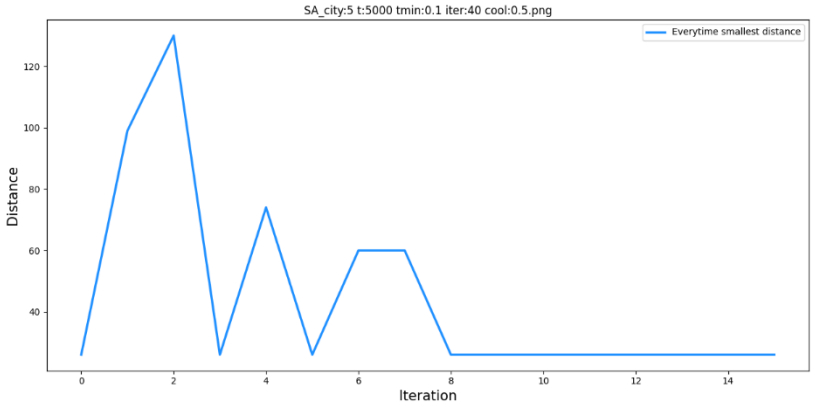


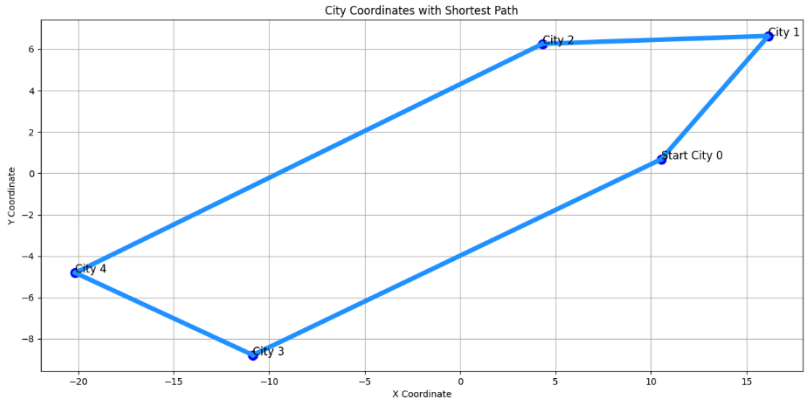
Initial Temperature : 50000 , temperature min : 0.1 ,Iteration : 40 ,coolrate = 0.98



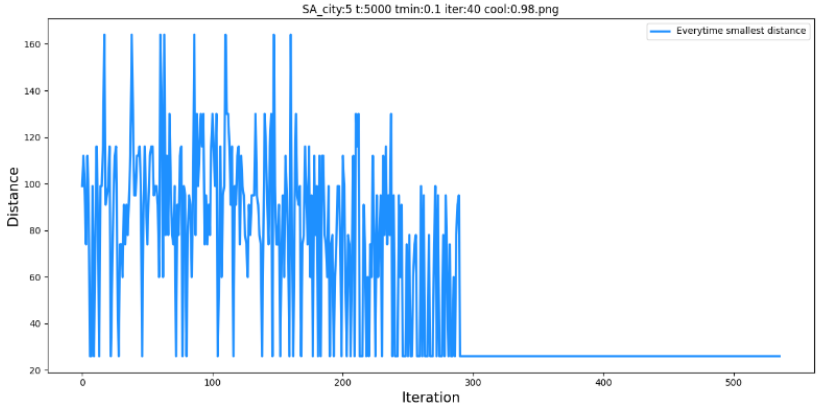


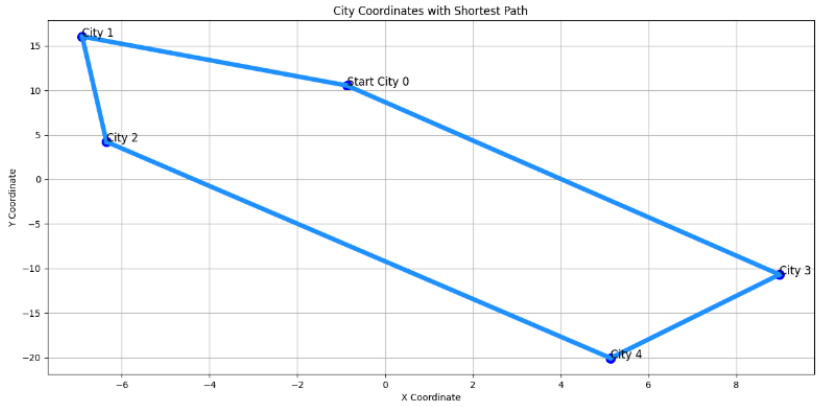
Initial Temperature : 5000 , temperature min : 0.1 ,Iteration : 40 ,coolrate = 0.5



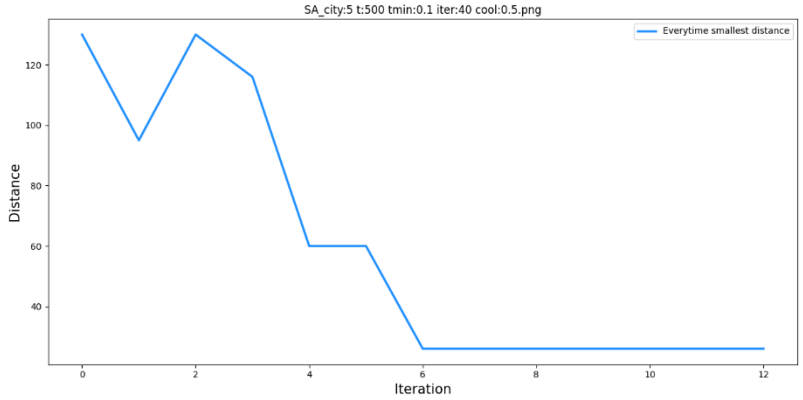


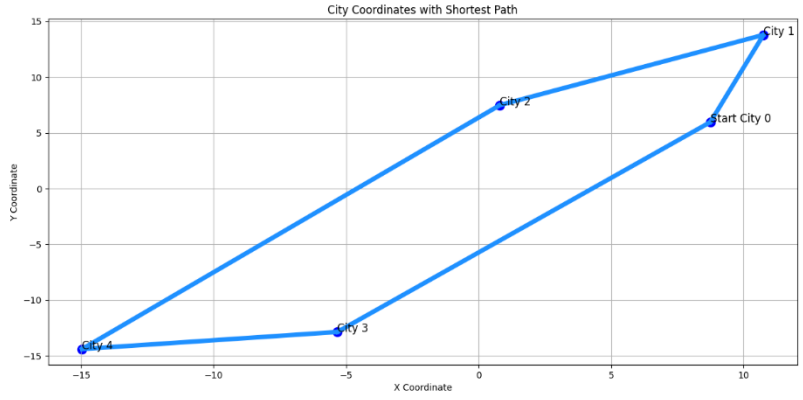
Initial Temperature : 5000 , temperature min : 0.1 ,Iteration : 40 ,coolrate = 0.98



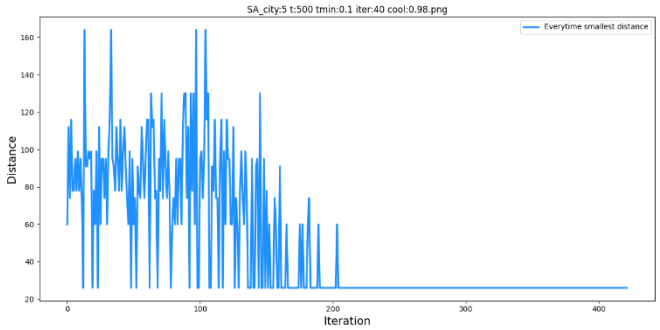


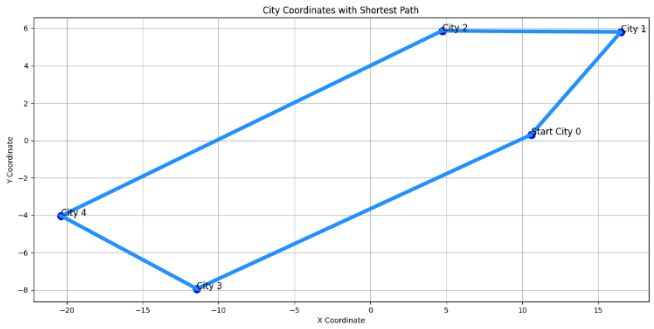
Initial Temperature : 500 , temperature min : 0.1 ,Iteration : 40 ,coolrate = 0.5





Initial Temperature : 500 , temperature min : 0.1 ,Iteration : 40 ,coolrate = 0.98





Section 4. 比較兩者優缺點和實作的心得。

當我嘗試比較兩者優缺點時，我試著用比較大的地圖去跑跑看，但後來發現用 Brute Force 都跑不出來，因為在深度優先演算法的過程需要許多函數以及 stack 去處理儲存的知訊，此外也是不斷的遞迴迭代。導致程式跑到當機。
然而用退火演算法就可以很快地得到答案，即便是算 1000 個城市的最短經過的距離也是可以在一秒多的時間內求得。雖然沒有用任何資料結構，就是單純的一直算數學，也可以算出答案並得到還算優的解。
以下為退火演算法跑各城市求得的時間。

```
(venv) E:\NCKU\4A\AI\HW2>python __main__.py
######################### Simulated Annealing:
city num: 50
best_path:  1716
best_dist:  [28, 39, 36, 1, 15, 18, 24, 42, 31, 8, 47, 35, 34, 12, 20, 33, 45, 38, 9, 44, 30, 6, 43, 19, 11, 49, 27, 2, 26, 29, 40, 3, 7, 17,
5, 32, 37, 22, 16, 10, 48, 25, 4, 46, 21, 0, 23, 41, 14, 13, 28]
elapsed time:  0.09269046783447266
```

```
(venv) E:\NCKU\4A\AI\HW2>python __main__.py
######################### Simulated Annealing:
city num: 20
best_path:  653
best_dist:  [15, 7, 16, 8, 18, 13, 6, 2, 3, 10, 19, 14, 11, 4, 1, 17, 5, 0, 12, 9, 15]
elapsed time:  0.06128811836242676
```

```
(venv) E:\NCKU\4A\AI\HW2>python __main__.py
######################### Simulated Annealing:
city num: 100
best_path:  4198
best_dist:  [49, 1, 73, 52, 96, 42, 92, 93, 38, 88, 28, 4, 87, 61, 56, 67, 71, 91, 79, 53, 74, 6, 70, 26, 90, 82, 54, 97, 55, 18, 27, 77, 84,
7, 14, 21, 72, 22, 47, 30, 86, 59, 50, 13, 15, 48, 41, 98, 2, 83, 66, 5, 10, 94, 45, 60, 89, 31, 11, 76, 12, 16, 25, 63, 43, 81, 34, 78, 46, 9
, 57, 75, 85, 58, 40, 99, 0, 32, 35, 36, 44, 64, 29, 62, 20, 69, 24, 68, 23, 51, 80, 19, 33, 8, 3, 17, 39, 65, 95, 37, 49]
elapsed time:  0.1426231861114502
```

在做這個作業的時候，我覺得我對 AI 的原理有一點初步的認識。以前訓練模型的時候我都常常納悶為什麼不用傳統方法算就好了，而深度學習的模型幾乎都是依據機率去求得最正確的答案。現在 AI 又為顯學，為什麼這種機率的方式，

還可以得到很好的結果，並且讓人類相信他。退火演算法中用機率的概念去判斷要不要更新。讓尋求答案的過程不會落入區域最憂解的想法其實很直覺，也真實出現現實生活中，剛開始的過程就是要不斷嘗試，而溫度比較大，嘗試的次數也會較頻繁，當嘗試多了就會漸漸開始收斂的概念，也很符合人類的行為會漸漸穩定下來。除了這種很符合現實的實作方式，也有計算的比傳統的方法快的優點，不需要花費太多的記憶體資源，即便答案無法保證是全局最憂解，但卻可以解決這種 TSP 旅行商人 NP-hard 的問題。

```
######################### Simulated Annealing:
city num: 500
best_path:  22962
best_dist:  [295, 486, 354, 3, 378, 164, 487, 381, 329, 479, 205, 142, 133, 103, 389, 137, 260, 438, 75, 104,
170, 74, 242, 405, 284, 60, 371, 198, 122, 193, 232, 217, 380, 26, 125, 393, 428, 117, 298, 80, 431, 386, 323,
 143, 94, 254, 202, 271, 469, 85, 259, 357, 374, 93, 8, 91, 401, 356, 1, 395, 454, 215, 146, 316, 207, 359, 10
8, 148, 45, 322, 385, 391, 400, 388, 437, 175, 434, 171, 24, 250, 406, 134, 349, 455, 286, 338, 65, 154, 132,
124, 174, 221, 127, 310, 360, 208, 128, 14, 86, 21, 441, 33, 188, 277, 165, 464, 480, 498, 223, 290, 343, 13,
261, 151, 145, 392, 0, 77, 231, 131, 495, 218, 191, 377, 342, 430, 336, 173, 482, 181, 258, 158, 334, 367, 168
, 263, 317, 365, 6, 36, 129, 82, 23, 335, 227, 140, 99, 220, 333, 439, 410, 363, 53, 190, 119, 268, 403, 252,
163, 421, 481, 38, 234, 297, 325, 477, 291, 433, 43, 355, 397, 396, 379, 364, 10, 55, 471, 411, 327, 160, 353,
 31, 83, 308, 253, 211, 326, 49, 362, 289, 219, 485, 452, 370, 100, 425, 44, 222, 319, 4, 282, 283, 426, 123,
294, 62, 28, 460, 408, 54, 121, 30, 272, 18, 239, 399, 412, 98, 275, 90, 458, 332, 71, 110, 101, 201, 189, 187
, 47, 314, 404, 248, 449, 177, 444, 302, 456, 257, 114, 92, 450, 266, 212, 306, 488, 382, 204, 281, 304, 46, 1
85, 87, 48, 366, 347, 301, 318, 206, 307, 41, 237, 58, 159, 424, 466, 244, 346, 497, 194, 139, 233, 249, 414,
183, 20, 184, 153, 25, 199, 136, 179, 37, 203, 499, 494, 224, 243, 288, 384, 417, 167, 300, 81, 348, 156, 78,
209, 59, 475, 459, 476, 344, 328, 115, 155, 387, 236, 390, 247, 84, 368, 197, 19, 138, 144, 269, 429, 89, 340,
 69, 72, 251, 96, 278, 22, 265, 102, 5, 493, 7, 147, 12, 320, 262, 112, 462, 407, 273, 423, 402, 422, 172, 182
, 2, 351, 245, 270, 152, 468, 463, 35, 66, 106, 478, 235, 68, 376, 157, 445, 315, 34, 375, 418, 324, 162, 226,
 180, 461, 161, 40, 213, 293, 113, 409, 9, 130, 166, 413, 373, 42, 135, 492, 169, 176, 150, 276, 331, 111, 16,
 339, 312, 448, 427, 216, 472, 330, 228, 61, 490, 358, 56, 419, 27, 79, 474, 443, 432, 451, 484, 70, 372, 178,
 483, 149, 470, 88, 230, 436, 32, 361, 465, 267, 107, 50, 440, 200, 309, 186, 116, 279, 73, 305, 225, 120, 214
, 76, 345, 292, 398, 496, 467, 67, 352, 256, 39, 394, 95, 97, 489, 29, 255, 285, 457, 303, 313, 491, 64, 420,
195, 280, 141, 447, 246, 241, 264, 57, 416, 210, 192, 446, 105, 435, 341, 415, 196, 238, 240, 321, 51, 287, 29
9, 311, 118, 369, 442, 63, 52, 126, 274, 11, 15, 229, 350, 453, 109, 17, 383, 296, 473, 337, 295]
elapsed time:  0.629913330078125
```

```
(venv) E:\NCKU\4A\AI\HW2>python __main__.py
######################### Simulated Annealing:
city num: 1000
best_path:  49817
best_dist:  [679, 999, 601, 870, 557, 945, 506, 772, 374, 205, 943, 985, 883, 416, 197, 662, 944, 841, 818, 4, 282, 194, 97, 463, 720, 270, 90
3, 566, 393, 732, 172, 403, 159, 51, 13, 826, 484, 132, 697, 498, 127, 425, 637, 947, 649, 641, 611, 520, 165, 21, 888, 607, 115, 458, 283, 98
8, 895, 542, 179, 956, 391, 342, 16, 173, 802, 568, 796, 766, 207, 701, 549, 169, 558, 535, 538, 223, 406, 599, 931, 208, 93, 712, 352, 204, 7
97, 334, 715, 617, 499, 923, 120, 438, 897, 456, 183, 734, 585, 785, 333, 859, 964, 485, 36, 5, 239, 39, 102, 17, 942, 634, 84, 534, 174, 930,
 648, 560, 881, 470, 136, 104, 364, 683, 586, 933, 834, 319, 750, 547, 694, 867, 75, 201, 596, 983, 297, 54, 196, 77, 518, 953, 488, 177, 659,
 188, 995, 608, 652, 301, 655, 941, 537, 299, 289, 929, 502, 440, 515, 176, 842, 108, 597, 65, 107, 137, 471, 884, 12, 486, 950, 730, 830, 214
, 693, 292, 861, 156, 32, 19, 602, 138, 963, 420, 546, 975, 906, 702, 533, 266, 619, 590, 970, 302, 724, 497, 635, 477, 670, 835, 394, 180, 22
5, 241, 737, 751, 717, 150, 757, 979, 26, 927, 786, 195, 855, 404, 962, 877, 220, 260, 258, 419, 37, 996, 410, 832, 400, 768, 82, 246, 304, 91
5, 435, 390, 614, 324, 151, 429, 572, 563, 79, 381, 593, 31, 684, 200, 290, 644, 344, 887, 38, 479, 721, 262, 327, 489, 771, 0, 7, 455, 809, 2
38, 980, 122, 161, 318, 365, 187, 874, 495, 770, 974, 228, 446, 864, 833, 853, 158, 30, 210, 170, 439, 64, 745, 212, 765, 822, 112, 493, 951,
450, 253, 98, 359, 145, 218, 160, 473, 310, 402, 111, 215, 633, 860, 94, 9, 738, 472, 494, 876, 678, 501, 244, 774, 577, 591, 326, 355, 550, 6
06, 866, 235, 119, 640, 556, 48, 305, 671, 620, 462, 762, 948, 345, 86, 418, 657, 852, 139, 314, 685, 729, 315, 909, 764, 157, 219, 134, 422,
2, 468, 264, 56, 376, 454, 449, 43, 981, 148, 532, 629, 452, 856, 303, 430, 994, 691, 321, 106, 967, 66, 668, 261, 880, 267, 209, 224, 579, 28
4, 123, 255, 71, 128, 229, 791, 912, 984, 491, 642, 955, 526, 25, 851, 42, 539, 801, 824, 496, 716, 559, 252, 464, 45, 279, 500, 893, 813, 379
, 839, 171, 965, 52, 478, 517, 222, 623, 658, 383, 63, 969, 475, 628, 44, 921, 743, 803, 650, 718, 280, 81, 372, 708, 588, 480, 356, 531, 752,
 423, 711, 776, 904, 353, 778, 574, 639, 638, 234, 85, 580, 389, 615, 129, 946, 646, 206, 749, 672, 828, 989, 849, 811, 565, 666, 959, 113, 65
3, 727, 669, 731, 126, 624, 805, 298, 22, 49, 773, 407, 714, 347, 348, 681, 632, 143, 763, 744, 189, 240, 76, 519, 392, 442, 399, 293, 14, 755
, 548, 436, 83, 541, 525, 35, 322, 689, 529, 739, 928, 96, 907, 960, 227, 276, 939, 934, 445, 889, 312, 59, 164, 583, 510, 571, 827, 530, 153,
 761, 769, 700, 453, 309, 957, 459, 434, 594, 779, 15, 760, 900, 278, 116, 29, 902, 821, 154, 598, 257, 603, 719, 686, 28, 426, 68, 971, 360,
466, 256, 522, 121, 146, 661, 323, 313, 198, 273, 800, 961, 810, 882, 892, 140, 130, 807, 536, 351, 469, 504, 643, 976, 936, 592, 677, 878, 74
2, 490, 990, 794, 808, 817, 524, 60, 782, 250, 320, 651, 70, 57, 168, 306, 358, 444, 385, 100, 325, 395, 845, 890, 300, 759, 74, 625, 27, 781,
 829, 272, 713, 285, 922, 508, 710, 509, 819, 869, 707, 554, 783, 636, 756, 131, 431, 362, 748, 552, 476, 338, 992, 747, 67, 575, 10, 875, 409
, 740, 421, 248, 799, 698, 405, 50, 503, 886, 275, 232, 243, 366, 896, 690, 254, 316, 722, 561, 825, 528, 474, 396, 288, 211, 328, 386, 753, 2
94, 237, 424, 53, 656, 367, 926, 412, 382, 175, 905, 664, 699, 144, 216, 746, 673, 271, 865, 247, 329, 193, 152, 203, 660, 871, 460, 703, 281,
 307, 844, 332, 354, 935, 567, 898, 885, 836, 451, 461, 269, 58, 61, 726, 966, 274, 665, 133, 553, 544, 846, 162, 135, 87, 919, 287, 879, 8, 9
73, 202, 155, 24, 467, 363, 790, 775, 166, 380, 512, 667, 540, 3, 918, 972, 622, 754, 806, 413, 858, 570, 551, 616, 793, 361, 600, 349, 514, 7
35, 398, 925, 167, 674, 337, 88, 604, 427, 401, 843, 578, 784, 296, 182, 90, 190, 840, 954, 73, 706, 609, 777, 335, 914, 437, 562, 850, 582, 9
86, 417, 191, 433, 373, 230, 397, 78, 823, 124, 908, 605, 350, 688, 920, 105, 910, 109, 481, 236, 330, 217, 263, 949, 958, 226, 647, 142, 587,
 141, 91, 812, 788, 185, 523, 101, 428, 847, 901, 792, 911, 432, 370, 527, 231, 630, 545, 186, 932, 147, 576, 286, 621, 295, 457, 917, 331, 38
4, 555, 411, 513, 820, 610, 687, 618, 447, 6, 899, 952, 645, 291, 780, 695, 72, 125, 938, 265, 968, 863, 482, 831, 993, 62, 521, 18, 492, 991,
 595, 483, 414, 997, 626, 448, 99, 369, 894, 55, 23, 1, 733, 149, 371, 233, 940, 654, 982, 663, 33, 692, 40, 184, 804, 543, 178, 516, 95, 378,
 873, 213, 408, 340, 339, 631, 998, 696, 937, 357, 789, 758, 987, 387, 573, 787, 581, 564, 838, 375, 341, 741, 41, 767, 505, 368, 249, 709, 44
3, 795, 221, 676, 487, 317, 725, 584, 277, 798, 872, 114, 589, 92, 511, 682, 69, 245, 868, 736, 816, 913, 259, 415, 916, 308, 848, 815, 242, 2
51, 569, 20, 978, 857, 163, 103, 89, 336, 862, 268, 117, 627, 192, 377, 891, 34, 837, 854, 80, 704, 311, 612, 924, 47, 680, 675, 977, 46, 507,
 728, 118, 11, 343, 441, 388, 199, 110, 705, 181, 814, 723, 346, 465, 613, 679]
elapsed time:  1.2918777465820312

(venv) E:\NCKU\4A\AI\HW2>
```