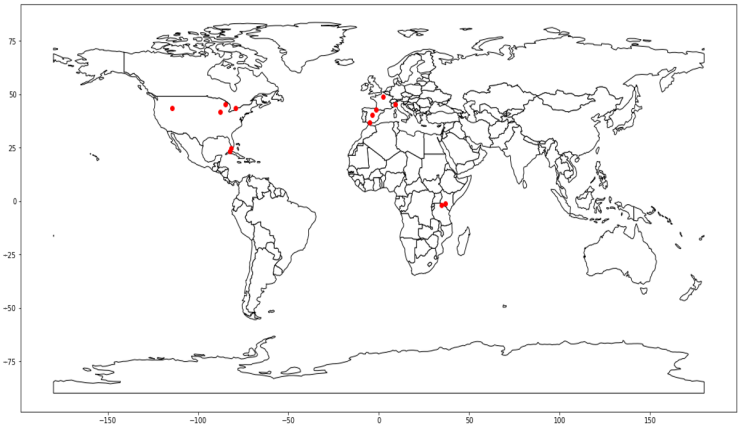
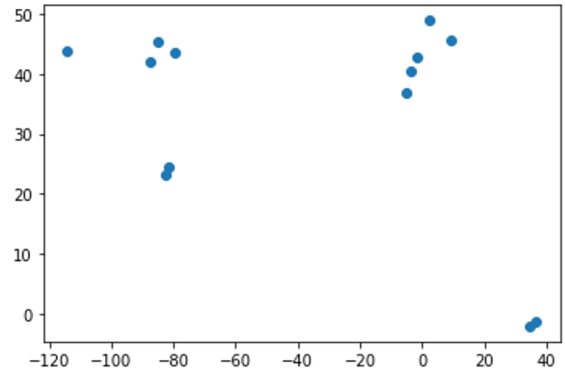
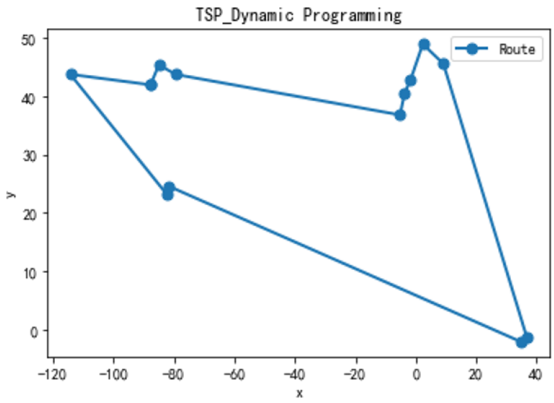
Find all locations

	location	latitude	longitude
0	Oak Park, Illinois, USA	41.887814	-87.788762
1	Horton Bay, Michigan, USA	45.284453	-85.078946
2	Paris, France	48.856697	2.351462
3	Pamplona, Spain	42.818454	-1.644256
4	Madrid, Spain	40.416705	-3.703582
5	Havana, Cuba	23.135305	-82.358963
6	Key West, Florida, USA	24.562557	-81.772437
7	Ketchum, Idaho	43.680741	-114.363662
8	Milan, Italy	45.466800	9.190500
9	Ronda, Spain	36.742134	-5.166592
10	Toronto, Canada	43.653482	-79.383935
11	Nairobi,Republic of Kenya	-1.283253	36.817245
12	Serengeti,Tanzania	-1.996626	34.742544



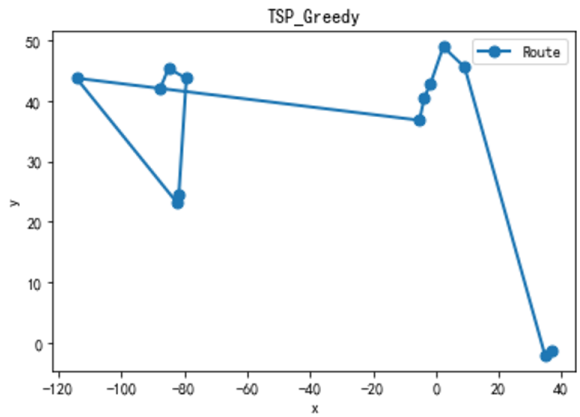
Scatter plot and DP method





DP route : 0 -> 1 -> 10 -> 9 -> 4 -> 3 -> 2 -> 8 -> 11 -> 12 -> 5 -> 6 -> 7 -> 0

Greedy Algorithm to TSP



TSP route 1 : 0-> 1-> 10-> 6-> 5-> 7-> 9-> 4-> 3-> 2-> 8-> 12-> 11

TSP route 2 : 7-> 0-> 1-> 10-> 6-> 5-> 9-> 4-> 3-> 2-> 8-> 12-> 11

My solutions to this TSP

To begin with, I need to find all locations Hemingway lived. To make the set of locations more accurate, I combine materials from Wikipedia with many other news about Hemingway's life. Finally, I get the dataframe and solve TSP based on that.

After getting the scatter plot from the world map, I use Dynamic Programming(DP) method to solve it, and the assumption is starting from Oak Park(Hemingway's born place) and going back to it finally. The route we get is ' 0 -> 1 -> 10 -> 9 -> 4 -> 3 -> 2 -> 8 -> 11 -> 12 -> 5 -> 6 -> 7 -> 0', and the route seems to be corresponding to the outline of these 13 scatter points. As for efficiency of DP, I find it costs approximately '0.97 s' on average.

I also use greedy algorithm to solve TSP but no need to return to the start point in the end. The assumption of the left graph is to set the Oak Park as the start point. The right one is based on no determined start point, which means I have to find which point is the best one to start to achieve the shortest route. Hence, we can find running time of the left graph is smaller than the right one. I test them and find the results are '0.0026 s', '0.0035 s' respectively. The routes are '0-> 1-> 10-> 6-> 5-> 7-> 9-> 4-> 3-> 2-> 8-> 12-> 11' and '7-> 0-> 1-> 10-> 6-> 5-> 9-> 4-> 3-> 2-> 8-> 12-> 11'.

Hence, we can find, if we set the start point, it is usually a longer route since we add this influential restriction. In fact, this situation is more common in real world because we seek to 'save time on way' based on we know where we are at that time.

Overall, Dynamic Programming method and Greedy Algorithm are two useful ways to TSP, but either of them has restrictions. DP is not efficient enough since we have to divide the TSP into many subproblems, and to some extent, use 'induction and deduction' to solve it step by step. As for Greedy Algorithm, although it is time-saving, it can not give us the best route at some time because it just provides us a 'current optimal solution'.

Hence, compared with DP, Greedy Algorithm may not solve TSP in a whole view(not consider the previous and future relations among steps), but it is not a bad method to TSP since it really saves time on complex and large TSP.