Question 2-1

1) LCS California and Carolina

		C	A	L	I	F	O	R	N	I	A
	0	0	0	0	0	0	0	0	0	0	0
C	0	1	1	1	1	1	1	1	1	1	1
A	0	1	2	2	2	2	2	2	2	2	2
R	0	1	2	2	2	2	2	3	3	3	3
O	0	1	2	2	2	2	3	3	3	3	3
L	0	1	2	3	3	3	3	3	3	3	3
I	0	1	2	3	4	4	4	4	4	4	4
N	0	1	2	3	4	4	4	4	5	5	5
A	0	1	2	3	4	4	4	4	5	5	6

LCS: 6 - Calina

2) Minimum Edit Distance

	Empty	C	A	L	I	F	O	R	N	I	A
Empty	0	1	2	3	4	5	6	7	8	9	10
C	1	0	1	2	3	4	5	6	7	8	9
A	2	1	0	1	2	3	4	5	6	7	8
R	3	2	1	1	2	3	4	4	5	6	7
O	4	3	2	2	2	3	3	4	5	6	7
L	5	4	3	2	3	3	4	4	5	6	7
I	6	5	4	3	2	3	4	5	5	5	6
N	7	6	5	4	3	3	4	5	5	6	6
A	8	7	6	5	4	4	4	5	6	6	6

Source: California
Target: Carolina

Minimum Edit Distance: 6

Question 2-2)

Dynamic Programming is impractical for the Bin Packing problem because it conflicts with the characteristics: Optimal Substructure and Overlapping Subproblems

Optimal Substructure: Bin Packing Problem does not have an optimal substructure. Take the 0/1 Knapsack problem, if you know how to pack a Knapsack of a smaller capacity, you can use that solution for a larger problem. This principle does not apply to the bin packing problem. If we split the items into subsets & pack them using the Knapsacks, we may find that this is not the optimal solution to the overall problem. Items from one Knapsack may be a better fit in another solution. Therefore we cannot break the bin problem into smaller problems and find the optimal substructure.

Overlapping Subproblems: In Dynamic programming, we generally need to solve the same subproblems multiple times. By recording and reusing the results of the subproblems, we can speed up the computation of the overall problem. The conflict with the bin packing problem is that there are an extremely large number of subproblems due to the combinatorial nature of the problem. For each subproblem, you need to record which items are packed & where. Because of this, the subproblems are very rarely reused & therefore are not overlapping.