Problem 1. Dynamic Programming  Consider a variation of the Longest Increasing Subsequence Problem. We are given an array $L$ of $n$ distinct alphabetic letters $a[1], a[2], \ldots, a[n]$ in unsorted order. Each letter has a weight and we want to find an alphabetically sorted (in increasing order) subsequence with the maximum total weight. For example, suppose $A$ has weight 5, $B$ has weight 1, $C$ has weight 6, and $D$ has weight 3. Then given the array $[B, A, B, C, D, B]$, the increasing subsequence $[B, C, D]$ has weight $1 + 6 + 3 = 10$ while the increasing subsequence $[A, C, D]$ has the maximum weight of $5 + 6 + 3 = 14$.

Describe a dynamic programming solution to find the weight of the maximum weight increasing subsequence. Specifically describe:

1. In words, what each index of the dynamic programming matrix (or array) should contain.
2. The size of the matrix (or array) to use.
3. The base cases to fill the matrix (or array).
4. A recursive formula for the other entries.
5. How to retrieve the total weight of the alphabetically sorted subsequence of maximum total weight once the matrix (or array) is filled.
6. The total time needed to fill the whole matrix (or array) in Big-Oh notation in terms of $n$. Briefly justify.
7. How to find the actual subsequence.

Problem 2. Greedy Algorithms  Recall the Longest Increasing Subsequence (LIS) problem from homework: we are given an array of $n$ distinct integers in unsorted order and we want to find the longest increasing subsequence in the array. Consider the following greedy algorithms for the problem. For each approach, either prove that the greedy approach is optimal or give an example where it does not yield an optimal solution. If giving an example, your example should include the list of numbers, the greedy algorithm’s solution, and the optimal solution.

(a) Add the first number in the list as the first value in the LIS. Then, from the second number to the end of the list, add the number to the LIS if it is larger than the most recently added number.

(b) First, scan the array of numbers to find the minimum number and add this number as the first value in the LIS. Then, starting from the position of the minimum number to the end of the list, add a number to the LIS if it is larger than the most recently added number.

Now, suppose the list is sorted (in increasing order) but may contain duplicates. Does the greedy strategy of part (a) yield an optimal solution? Either prove that the greedy approach is optimal or give an example where it does not yield an optimal solution. If giving an example, your example should include the list of numbers, the greedy algorithm’s solution, and the optimal solution.

Problem 3. Data Encoding

(a) Consider the following greedy strategy for data encoding where the goal is to use the fewest number of bits to encode a text file. First, sort the characters in decreasing order of frequency. Let $c_1, c_2, \ldots, c_n$ denote the characters in this sorted order. Now, assign $c_1$ a code of 0 (the shortest code), $c_2$ a code of 1, $c_3$ a code of the binary equivalent of $\lceil \log_2(n) \rceil$ (the longest code).

Either prove this algorithm is correct or explain why it does not yield an optimal encoding.
(b) In our discussion of Lempel-Ziv decoding we found that for certain encoded strings, it is possible that the code we need is not yet in the dictionary. We discussed that this situation can occur for strings of the form $X <\text{char}> Y$ where $X <\text{char}>=Y$. In our example, the issue occurred for the substring $X <\text{char}> Y = AAA$, so $X = Y = AA$ (and $<\text{char}>= A$. Give a substring where this situation occurs for $X \neq Y$.

**Problem 4. Maximum Reliability Path**

We are given a directed graph $G = (V,E)$ and, for each edge $(u,v) \in E$, the probability $f(u,v)$ that the edge $(u,v)$ may fail (break down). These probabilities are independent; that is, the probability that no edge fails in the path $(u_1,u_2,\ldots,u_k)$ is $(1 - f(u_1,u_2))(1 - f(u_2,u_3))\cdots(1 - f(u_{k-1},u_k))$. This product of probabilities is called the reliability of the path $(u_1,\ldots,u_k)$.

Describe an algorithm that given $G$, the probability $f(u,v)$ for all $(u,v)$, and two vertices $s,t \in V$, finds the path from $s$ to $t$ of maximum reliability.

**Problem 5. Graphs**

Consider the problem of finding connected components in an undirected graph. A connected component of a graph is a connected subgraph. Formally, in a graph $G = (V,E)$, a connected component is a subset $S \subseteq V$ such that for every pair of vertices $u,v$ in $S$, there is a path from $u$ to $v$.

Describe a $O(|V| + |E|)$-time algorithm that finds the connected components of an undirected graph. If the graph has $k$ connected components, the algorithm should assign a component number from $1 \ldots k$ to every vertex in the graph.