CSCC73
Algorithm design & analysis

Week 8 Tutorial
Question 1

Design a polynomial-time DP algorithm that, given two sequences A[1..m] and B[1..n], finds the length of a longest common subsequence of A and B. Analyze the running time of your algorithm.

Retrofit the algorithm to find an actual longest common subsequence of the given sequences.

**Hint:** Think along the lines of the algorithm to find the edit distance of two strings.

(Finding a longest common subsequence is what the Unix command “diff” does, viewing each text file as a sequence of lines.)
Question 2

Give a pseudo-polynomial time DP algorithm to solve the “knapsack with replacement” problem:

For each item \(i=1, 2, \ldots, n\), let \(v(i)\) be the value and \(w(i)\) be the weight of \(i\), where each weight is a positive integer. Let \(C\) be a positive integer (the knapsack capacity).

Assume that there is an unlimited number of each item. We want to find how many copies \(S(i)\) of each item \(i\) to steal, where \(S(i)\) is a non-negative integer, so that the stolen items have maximum value and fit in the knapsack: find \(S\) that maximizes \(\sum_i S(i)v(i)\) so that \(\sum_i S(i)w(i) \leq C\).
Question 3

1) Do longest paths of a directed acyclic graph (DAG) exhibit the optimal substructure property? That is, if p is a longest u -> v in a DAG, and p’ is a prefix of p from u to some node x, is p’ a longest u -> x path?

2) Give a DP algorithm that, given a DAG G and a node s of G, finds the weight of a longest path in G starting at s. Your algorithm should run in O(n+m) time, where n is the number of nodes and m is the number of edges of G. (Hint: Use a topological sort of G.)

3) Give an algorithm that finds the weight of a longest path in a DAG G and runs in O(n+m) time.
Question 4

Run the Bellman-Ford algorithm on the graph below where the start node is 1. Show the L-value of every node in every iteration. (Shown are the L-values after initialization.)

```

1 | 2  | 3  | 4  | 5  | 6  |
---|----|----|----|----|----|
0  | 0  | ∞  | ∞  | ∞  | ∞  |
1  | ∞  | 1  | ∞  |    |    |
2  | ∞  | 2  |    |    |    |
3  | ∞  | 3  |    |    |    |
4  | ∞  | 4  |    |    |    |
5  | ∞  | 5  |    |    |    |
```
Question 5

Run the Floyd-Warshall algorithm on the graph below. Shown next to the graph is the matrix $C(-,-,0)$. Show the matrix $C(-,-,k)$ for every other relevant value of $k$. 

\[
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
1 & 1 & 0 & 3 & 8 & 5 \\
2 & 2 & \infty & 0 & \infty & 1 \\
3 & 3 & \infty & 1 & 0 & \infty \\
4 & 4 & \infty & \infty & 2 & 0 \\
\end{array}
\]
Question 6

1) Does the Bellman-Ford algorithm find a shortest path from node s to every node with the minimum number of edges? If not, can you change it so that it does?

2) Does the Floyd-Warshall algorithm find a shortest path between every pair of nodes with the minimum number of edges? If not, can you change it so that it does?