Assignment 8

CSCI 3110: Design and Analysis of Algorithms

Due July 16, 2019

Banner ID: _____ Name: _____

Banner ID: _____ Name: _____

Banner ID: _____ Name: _____

Assignments are due on the due date before class and have to include this cover page. Plagiarism in assignment answers will not be tolerated. By submitting their answers to this assignment, the authors named above declare that its content is their original work and that they did not use any sources for its preparation other than the class notes, the textbook, and ones explicitly acknowledged in the answers. Any suspected act of plagiarism will be reported to the Faculty's Academic Integrity Officer and possibly to the Senate Discipline Committee. The penalty for academic dishonesty may range from failing the course to expulsion from the university, in accordance with Dalhousie University's regulations regarding academic integrity.

1. You are given a string of n characters $s[1 \ldots n]$, which you believe to be a corrupted text document in which all punctuation has vanished (so that it looks something like "itwasthebestoftimes..."). You wish to reconstruct the document using a dictionary, which is available in the form of a Boolean function **dict**$(\cdot)$: for any string w,

$$\textbf{dict}(w) = \begin{cases} true & \text{if } w \text{ is a valid word} \\ false & \text{otherwise.} \end{cases}$$

   (a) (10 pts) Give a dynamic programming algorithm that determines whether the string $s[\cdot]$ can be reconstituted as a sequence of valid words. The running time should be at most $O(n^2)$, assuming calls to dict take unit time. Start by describing the problem as an array $d[\cdot]$ and then provide a recurrence for $d[i]$ in terms of $d[j]$ where $j < i$. Then determine the dynamic programming order that solves this recurrence efficiently and give pseudocode that does this.

   (b) (Bonus: 5 pts) In the event that the string is valid, make your algorithm output the corresponding sequence of words. Use a choice array to record the optimal choices made by your algorithm and then use a backtracking algorithm to output the sequence of words.

2. **Time and space complexity of dynamic programming.** Our dynamic programming algorithm for computing the sequence alignment edit distance between strings of length m and n creates a table of size $n \times m$ and therefore needs $O(nm)$ time and space. In practice, it will run out of space long before it runs out of time. How can this space requirement be reduced?

   (a) (5 pts) Show that if we just want to compute the value of the edit distance (rather than the optimal sequence of edits), then only $O(n)$ space is needed, because only a small portion of the table needs to be maintained at any given time.

   (b) (5 pts) Now suppose that we also want the optimal sequence of edits. This problem can be recast in terms of a corresponding grid-shaped directed acylic graph, in which the goal is to find the optimal path from node $(0, 0)$ to node (n, m). It will be convenient to work with this formulation, and while we're talking about convenience, we might as well also assume that m is a power of 2. Let's start with a small addition to the edit distance algorithm that will turn out to be very useful. The optimal path in the directed acylic graph must pass through an intermediate node $(k, m/2)$ for some k; show how the algorithm can be modified to also return this value k.

   (c) (5 pts) Now consider a recursive scheme:

   Find-Path$((0, 0) \rightarrow (n, m))$
   1 compute the value k above
   2 Find-Path$((0, 0) \rightarrow (k, m/2))$
   3 Find-Path$((k, m/2) \rightarrow (n, m))$
   4 concatenate these two paths, with k in the middle

   Show that this scheme can be made to run in $O(nm)$ time and $O(n)$ space.