
# Assignment-1
# Decrypting Messages

February 14, 2024

## Problem statement

You are consulting for the Super Turtles. The turtles have been intercepting message transmissions between the Ripper's Foot soldiers. These messages are long strings of characters which at first glance make no sense. For example, the messages look like: "Meetmeatseven". When you look closely you figure out that it says "Meet me at seven".[1] There are several hundreds of these messages, some more complex than others. As an algorithmist, you are asked to devise a clever way to decrypt all these messages.

A simple approach is to try all possible ways to split the string (add a space character between two characters) and compute the quality of the decryption. The decryption with the highest cumulative quality should be the correct message. Given a long string of letters $y = y_1 y_2 \dots y_n$, a decryption of $y$ is a partition of its letters into contiguous blocks of letters; each block corresponds to a word in the message. The total quality of a decryption is determined by adding up the qualities of each of its blocks. (So we would get the right answer above provided that quality("meet") + quality("me") + quality("at")+ quality("seven") was greater than the total quality of any other decryption of the string.)

Assume that you are given a function $q$ that takes as its input a string and outputs its quality in $O(1)$ time (and the computation does not depend on the length of the string). The quality of a string is positive if it is more likely to be an English word. The quality of strings that have no meaning is negative. So,

[1] You could also read it as "Meet meats even" or "Meet meat seven". However, for this exercise, we only care for the decryption to make sense at the word level. We do not care if the underlying grammar is correct.

$q$("tseve") is negative. Give an efficient algorithm to find the highest quality decryption for any input string $y$.

You can assume that the string has no special characters and any word can be of length upto $O(n)$, i.e, you do not need to look up the longest word in English.

1. What is your top-choice for this problem? (1)
2. What is your subproblem for the top-choice you described above? (1)
3. Write the subproblem for your top-choice as a recurrence relation. (2)
4. Prove the optimality principle for your subproblem for the corresponding top-choice. (3)
5. Write a pseudocode for your algorithm and analyze its runtime. (3)