# Department of Information Technology, NITK

Design and Analysis of Algorithms (IT257) - Laboratory Exercise 5
March 27, 2023

As some of you know well, and others of you may be interested to learn, several languages (including Chinese and Japanese) are written without spaces between the words. Consequently, software that works with text written in these languages must address the word segmentation problem—inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like *"meetateight"* and deciding that the best segmentation is *"meet at eight"* (and not *"me et at eight,"* or *"meet ate ight,"* or any of a huge number of even less plausible alternatives). How could we automate this process?

A simple approach that is at least reasonably effective is to find a segmentation that simply maximizes the cumulative *"quality"* of its individual constituent words. Thus, suppose you are given a black box that, for any string of letters $x = x_1x_2...x_k$, will return a number *quality(x)*. This number can be either positive or negative; larger numbers correspond to more plausible English words. (So *quality("me")* would be positive, while *quality("ght")* would be negative.)

Given a long string of letters $y = y_1y_2...y_n$, a segmentation of y is a partition of its letters into contiguous blocks of letters; each block corresponds to a word in the segmentation. The total quality of segmentation is determined by adding up the qualities of each of its blocks. (So we'd get the right answer above provided that *quality("meet")* + *quality("at")* + *quality("eight")* was greater than the total quality of any other segmentation of the string.) Give an efficient algorithm that takes a string $y$ and computes a segmentation of maximum total quality. (You can treat a single call to the black box computing *quality(x)* as a single computational step.)