1 2016/09/13

1.1 The Longest Common Subsequence

From a string S of length m, there will be a considerable number of subsequence. The longest common subsequence problem tries to find the longest subset of the string that lies both in two different strings. It is used in DNA sequencing and also in UNIX or LINUX operating systems.

Let the two sequences be

$$S = \langle s_1, s_2, \ldots, s_m \rangle = S\,[1 \ldots m] \tag{1}$$

$$T = \langle t_1, t_2, \ldots, t_n \rangle = T\,[1 \ldots n] \tag{2}$$

and let the common subsequence be denoted by $U\,[1 \ldots]$ if

$$1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq m \tag{3}$$

$$1 \leq j_1 \leq j_2 \leq \ldots \leq j_k \leq n \tag{4}$$

such that $S\,[i_k] = T\,[i_t] = U\,[t]$ for all $t \in \{1, 2, \ldots, k\}$.

(Approach based on the recursive equations): Define

$$\text{LCS}\,[i, j] = \text{length of LCS of } S\,[1 \ldots i]\ \&\ T\,[1 \ldots j] \tag{5}$$

In other words, break down the problem into a smaller problem where the two strings are shorter and then find the longest common subsequence. For example, let's say there are two strings of length 5 and we want to find LCS $[3, 2]$. We first look at the last characters of both strings and break it down to two different situations.

- where $S\,[3] = T\,[2]$
- where $S\,[3] \neq T\,[2]$

In the first case, the solution would be as follows

$$\text{LCS}\,[3, 2] = 1 + \text{LCS}\,[2, 1] \tag{6}$$

and the second case,

$$\mathrm{LCS}\left[3,2\right]=\max\left\{\mathrm{LCS}\left[2,2\right],\mathrm{LCS}\left[3,1\right]\right\} \tag{7}$$

Therefore, to generalize

$$\mathrm{LCS}\left[i,j\right]=\begin{cases}\max\left\{\mathrm{LCS}\left[i-1,j\right],\mathrm{LCS}\left[i,j-1\right]\right\}, & \text{if } S\left[i\right]\neq T\left[j\right]\\ \mathrm{LCS}\left[i-1,j-1\right], & \text{if } S\left[i\right]=T\left[j\right]\end{cases} \tag{8}$$

1.2 Dynamic Programming

Dynamic programming is essentially breaking down the problem into its smaller subsets and then sequentially solving them to obtain the final solution to the original problem. It is also the reason why we call the problems that DP deals with *multi-stage decision problems*.

1.3 The Principle of Optimality

- The principle of optimility holds if
  - every optimal solution to a problem contains the optimal solutions to all sub-problems.
- However, the principle of optimality does not say
  - if you have optimal solutions to all subproblems, you can combine them to get an optimal solution of a problem.