MATH578A Assignment 2

Raktim Mitra
USC ID: 1487079265 email: raktimmi@usc.edu

April 13, 2020

Q1.

normal pairwise alignment recursion

$$M[i,j] = max \begin{cases} M[i,j-1] + g(s_1[i], \_) \\ M[i-1,j] + g(\_, s_2[j]) \\ M[i-1,j-1] + g(s_1[i], s_2[j]) \end{cases}$$

Here, g is the scoring matrix with relevant scoring scheme. But we cannot have less than k contiguous matches and/or mismatches. So, if we want to take a diagonal step, then it has to be at least k diagonal step. No new restriction on taking an= insertion or deletion step. So, modified recursion steps:

k-core alignment recursion

$$M[i,j] = max \begin{cases} M[i,j-1] + g(s_1[i], \_) \ [O(1) \text{ time}] \\ M[i-1,j] + g(\_, s_2[j]) \ [O(1) \text{ time}] \\ M[i-k,j-k] + \sum_{n=0}^{k-1} g(s_1[i-n], s_2[j-n]) \ [O(k) \text{ time}] \end{cases}$$

The above simply says, when we want to take a diagonal path, we have to reset previous k-1 steps to diagonal steps also. Otherwise, we cannot take a digonal path. This ensures, we have only streches of contiguous k or more sequences of match and mismatches.

Notice, this diagonal recursion does not happen for the first $k-1$ row and $k-1$ columns. Which is expected because we can make a diagonal step in those region only if it creates a k-contiguous diagonal step which reaches outside and into the region $M[k..n, k..m]$. I.e. we will calculate those diagonal steps retrospectively when we are looking at positions further into the sequences. so, for ($s_1[1..k-1]$ and $s_2[1..k-1]$) we have to take only horizontal or vertical steps. So, for this region,

$$M[i,j] = max \begin{cases} M[i,j-1] + g(s_1[i], \_) \\ M[i-1,j] + g(\_, s_2[j]) \end{cases}$$

With the above recursive definitions, we can initialize the score matrix in the same manner as we did in normal pairwise global alignment (by taking only insertion or deletion paths for the first row and column). Now, we can apply the dynamic programming to calculate all values of M to calculate the required optimal score at $M[n,n]$. And, we can also keep the path pointers in another matrix to do traceback and construct the alignment in a straight forward manner.

Ignoring the modification for first k columns and rows, every entry of the matrix M takes O$(k+2)$ computation. So, total complexity is $O((k+2)mn)$ which is $O(kmn)$.

Q2.

Q3.

Q4.

Part a.
T _ G C A C C
T _ G _ A C C
T _ G A A C G
G _ G _ A C C
G _ A _ A C C
A A G _ A C C
T _ G _ A C C

Part b.