Exercise set 22

1. Answer in back
2. Simply restrict the number of indexes you consider to only those with |i - j| <= k.

In the original NSW algorithm you must compare every index X[1, 2, ..., m] with Y[1, 2, ..., n] essentially building the optimal sequence as it goes and storing its NW score in the array, working upwards and rightwards (i.e. building columns sequentially). Continuing the visualization of the algorithm building potential strings as it goes, each instance A[m][n] is the best NW score possible for the strings X[1,...,m] and Y[1,...,n]. Lets imagine its building the column for A[2] with a maximum of 2 gaps allowed (k=2). As the algorithm computes the values for the column A[2] it is only considering the best possible NW score of the first two entries of X (X[1,2]) versus every entry of Y (Y[1, ..., n]) sequentially. That is to say when it gets to the step of computing the NW score of A[2,3] there must be a gap in the optimal sequence, and when it computes the score of A[2,4] there must be at least two gaps in the optimal alignment (this is because it is aligning a string of size 2 and 3 and 2 and 4 respectively). Knowing this, it is a waste of time to compute entries in the column A[i] that must have more gaps in its optimal sequence than are allowed (k) so in this case A[2][5] and so on. That is, computing A[i][n < i-k] and A[i][n > i+k] is unnecessary. Simply modify the NW algorithm so that the dual for loop looks like so.

for i := 1 to m do

    for j := n - (k +1) to n + (k+1) //we want to compute n - k and n + k

The only way I can explain this is that it makes m \* (2k) comparisons which for a 6x6 comparison is equivalent to (m + n) \* k.

3. Given two non-empty strings X[1, 2, ..., m] and Y[1, 2, ..., n] there are three possible cases for the LCS of the two strings:

1. X[m] and Y[n] match and the LCS equals the LCS of X[1, ..., m-1] and Y[1, ..., n-1] plus X[m] and Y[n]

2. X[m] and Y[n] do not match and the LCS equals the LCS of X[1, ..., m] and Y[1, ..., n-1]

3. X[m] and Y[n] do not match and the LCS equals the LCS of X[1, ..., m-1] and Y[1, ..., n]

Let A be an m x n array to hold our results then;

if X[m] = Y[n] then A[m,n] = A[m-1,n-1] + 1

if X[m] /= Y[n] then A[m, n] = max(A[m-1, n], A[m, n-1]

Following the last problem our array A will be built bottom up with zeros initialized for the base case of empty strings. Indexing for A begins at zero. I'm ignoring describing the reconstruction process as it is obvious.

**LCS**

Input: Strings X[1, 2, ..., m] and Y[1, 2, ..., n]

Output: LCS value

A := (m + 1) \* (n + 1)

//base case j = 0

for i := 0 to m do

    A[i][0] = 0

//base case i = 0

for j := 0 to n do

    A[0][j] = 0

// populate the array

for i := 1 to m do

    for j := i to n do

        if X[i] == Y[j] do

            A[i][j] = A[i-1][j-1] + 1

        else do

            A[i][j] = max(A[i-1][j], A[i][j-1])

return A[m][n]