**Homework 4**

1. **Consider the following problem: Longest Common Subsequence**

* **Input: arrays A[1…n] of integers and B[1…k]**
* **Output: length of longest common subsequence of the two array**

**Example: if A = [1; 3; 2; 5; 4; 3; 5; 8] and B = [2; 3; 5; 8; 1; 3; 6; 4; 7; 8], then 1; 3; 8 is a common subsequence, but so is 3; 5; 3; 8. Thus, a subsequence does not need to be a contiguous set of entries from each array – that would be called a “substring" instead of a “subsequence".**

1. **Give a dynamic programming algorithm to solve Longest Common Subsequence for an arbitrary input. Answer the following questions:**

Matlab Script and the getMax function on next two pages…

% script MaxSequenceSize

% initialize maxValue

maxValue = 0;

% input arrays

A = [1,3,2,5,4,3,5,8];

B = [2,3,5,8,1,3,6,4,7,8];

% Reverse the arrays

Areversed = fliplr(A);

Breversed = fliplr(B);

% Create a blank matrix

matrix = zeros(size(Areversed), size(Breversed));

% Iterate through each element of Areversed

for i = 1:length(Areversed)

% Iterate through each element of Breversed

for j = 1:length(Breversed)

% Look for elements that are the same

if(Areversed(i) == Breversed(j))

% Call function getMax and send a

% subarray size i-1 by j-1

% then add one and set it to the max

max = getMax(matrix(1:i-1,1:j-1)) + 1;

% Set the matrix value at i,j to the

% max var (max found + 1)

matrix(i,j) = max;

% Now, if this max happens to be bigger

% than the global max

% go ahead and replace that value.

if(max > maxValue)

maxValue = max;

end

else

% If there is no match, go ahead and set

% the value to zero

matrix(i,j) = 0;

end

end

end

% getMax function that is used in the script

function max = getMax(subMatrix)

% Initalize max

max = 0;

% Find boundries of the submatrix

[Asize, Bsize] = size(subMatrix);

% iterate through the subMatrix

for i = 1:Asize

for j = 1:Bsize

% If a value was found in the

% subMatrix that was bigger than max,

% then go ahead and set max to that value

if(subMatrix(i, j) > x)

max = subMatrix(i,j);

end

end

end

* **What do your variables mean?**

- A and B are just the input arrays.

- Areversed and Breversed are A and B reversed.

- maxValue is the maximum length of a subsequence that was found in both of the -sequences

- matrix is the matrix that holds all of the matches between the two arrays, and how what the max subsequence could be at that point.

* **How do you initialize your variables?**

- That’s pretty self explanatory in the code above

* **How do you set your variables (other than the “boundary" condition variables)?**

- Again, the code above explains

* **What order do you calculate your variables?**

- I reverse the arrays and calculate from there.

* **Where is the answer stored?**

- The answer is stored in the matrix that is created and the maxValue variable.

1. **Apply your algorithm to the above pair of arrays.**

Example matrix output:

**8 7 4 6 3 1 8 5 3 2**

**8** 1 0 0 0 0 0 1 0 0 0

**5** 0 0 0 0 0 0 0 2 0 0

**3** 0 0 0 0 2 0 0 0 3 0

**4** 0 0 2 0 0 0 0 0 0 0

**5** 0 0 0 0 0 0 0 3 0 0

**2** 0 0 0 0 0 0 0 0 0 4

**3** 0 0 0 0 3 0 0 0 4 0

**1** 0 0 0 0 0 4 0 0 0 0

Max subsequence found (Output): **4**

Note: This algorithm could be optimized in a few ways like just finding the first max value and returning it, but there may be some edge cases. Also, I could have tried to keep row and column max values so I wouldn’t have to search through each time a match is found and reduce the time it takes. I chose this layout because it is easier to understand and it works for the example data.

2. **Prove by contradiction that sqrt(5) is not rational**

3. **Find a closed form solution for F(n) = 1 + 3 + 5 +…+ (2n1), and prove it correct by induction on n.**

4. **Let A = (8; 7; 4; 3; 0; 2; 3; 4; 5; 10). Compute the following:**

5. **Prove that 3n3 is O(n3).**

6. **Prove that 4n2 + 1000000 is O(n3).**