Dennis Kuzminer
CSCI-UA 310-001 PS5

5. First, we create an m by n matrix where we create an extra row and column for gaps. Each cell will be a subproblem. Fill each of the entries in that column and row with the score of the gap (0, 1, 2, 3 ....) and continue to increment down for all n and m. Fill in each position value by scoring the next block according to its previous subproblems. If a cell has a row = column (a match), we carry down the score of the cell above and to the left (diagonally). If there is a mismatch, we will do the same as a match but increment the score. If we move from left to right or upper to lower cells, this would denote a gap, meaning the value of this subproblem would be adding the score of the left or upper subproblems to the gap score of 2. We will also consider the true value of a cell or subproblem as the minimum score required to get the answer (when choosing conflicting values for carrying down diagonal or horizontal/vertical). After filling out the cells and possible combinations, we look to the bottom right of the matrix for the smallest value (value with least penalty) and use that value to backtrack to the best possible alignment. We will move based on which nearby value has the minimum score. Even with memoization, we are looping over a 2d array with size m*n, making the runtime O(n*m).