1.A Knapsack Problem:

public class Knapsack {

public static int knapsack(int W, int[] weights, int[] values) {

int n = weights.length;

int[][] dp = new int[n + 1][W + 1];

for (int i = 1; i <= n; i++) {

for (int w = 1; w <= W; w++) {

if (weights[i - 1] <= w) {

dp[i][w] = Math.max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1]);

} else {

dp[i][w] = dp[i - 1][w];

}

}

}

return dp[n][W];

}

public static void main(String[] args) {

int W = 10; // Knapsack capacity

int[] weights = {2, 3, 4, 5};

int[] values = {3, 4, 5, 6};

int maxTotalValue = knapsack(W, weights, values);

System.out.println("Maximum total value: " + maxTotalValue);

}

}

2.A Longest Common SubSequence

public class LongestCommonSubsequence {

public static int LCS(String text1, String text2) {

int m = text1.length();

int n = text2.length();

int[][] dp = new int[m + 1][n + 1];

for (int i = 1; i <= m; i++) {

for (int j = 1; j <= n; j++) {

if (text1.charAt(i - 1) == text2.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + 1;

} else {

dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);

}

}

}

return dp[m][n];

}

public static void main(String[] args) {

String text1 = "AGGTAB";

String text2 = "GXTXAYB";

int lengthOfLCS = LCS(text1, text2);

System.out.println("Length of LCS: " + lengthOfLCS);

}

}