COM2031 Advanced Algorithms, Autumn Semester 2019

Lab 6: Dynamic Programming
Knapsack, and Sequence Alignment

Purpose of the lab

Your task is to implement the Knapsack algorithm presented in the Lectures on Dynamic Programming.

1. Knapsack (core)

Implement the Knapsack algorithm (as presented in the lecture or in the Kleinberg/Tardos book) based on dynamic programming. You will be provided with a Java Template `KnapSack` class, but feel free to use any other programming language of your choice or implement from scratch without using the template. Implement it making adequate use of the data structures that the standard (class/template) libraries provide (Arrays, but also such as Containers, Collections etc) and basic algorithms that are part of the standard libraries (such as Comparator, Arrays.sort or Collections.sort for Java).

1. Use the provided `Item` class that manages the data that make up an Item or design your own data structures as you see fit.
2. Within the `KnapSack` class, implement the `knapsack` static method. It takes a list of Items and return the array of OPT-values M.
3. Also implement the `findSolution` method that calculates the maximal value set for a given weight limit from the table of OPT-values and returns the selected items in an appropriate data structure.
4. Finally, let your knapsack algorithm run on the sample set provided in the `main` method. Upon success, continue testing using the `testKnapSackSimple` method. It contains more random and border values test cases to experiment with.

2. Sequence Alignment (additional)

The Sequence Alignment problem takes two strings and finds the best alignment to maximize the number of matching characters and minimize the penalty of mismatched and gaps. The output is the alignment of the strings, character by character including possible mutations (accepted mismatches of small penalty) and accepted gaps. The penalty is calculated as:

1. A penalty of p_{gap} occurs if a gap is inserted between the string.
2. A penalty of p_{xy} occurs for mis-matching the characters of X and Y.

Examples:

Input : X = CG, Y = CA, p_gap = 3, p_xy = 7
Output : X = CG_, Y = C_A, Total penalty = 6

Input : X = AGGGCT, Y = AGGCA, p_gap = 3, p_xy = 2
Output : X = AGGGCT, Y = A_GGCA, Total penalty = 5

Input : X = CG, Y = CA, p_gap = 3, p_xy = 5
Output : X = CG, Y = CA, Total penalty = 5

Input : X = AGGGCT, Y =AGGCA, p_gap = 3, p_xy = 2
Output : X = AGGGCT, Y = A_GGCA, Total penalty = 5

Implement the Sequence Alignment Algorithm from the Sequence Alignment lecture slides. This is the Needleman-Wunsch algorithm (https://en.wikipedia.org/wiki/Needleman–Wunsch_algorithm).

Check the code in `SeqAlign` class for an example applying a simplified scoring scheme for the Needleman–Wunsch algorithm Sequence Alignment algorithm

Can you see how you can make the `SeqAlign` classes work for 3 Strings?