Assignment 2

Start Assignment

Due Wednesday by 11:59pm **Points** 100 **Submitting** a file upload

For each of the following problems, you need to provide a dynamic programming solution, prove correctness and analyze the running time and space complexity. Moreover, implement each algorithm in a compiled language of your choice and perform experiments to determine the relationship between the input size and the running time (wall clock time). Put both the theoretical and experimental work in a LaTex document. Submit all your code, LaTex sources, and PDF of the report.

1. **Weighted approximate common substring [50 points]**. Given two input strings, not necessarily of the same length, based on alphanumeric characters [A-Z], determine the **best** common substring. A substring is a **contiguous** sequence of characters within a string, and the score that determines the best substring is defined as the sum of the weights w_l for each character in the sequence (i.e. w_A is the weight of matching letter A) and a penalty $-\delta$ for each mismatch (negative penalty term to drive down the score). **In your experiments**, consider situations in which $w_l = 1$ and $\delta = 10$ and in which w_l is proportional to the frequency of the letter in English and δ takes values between the smallest and the largest weight (multiple experiments for 10 intermediate values). **Note:** you are now allowed to add gaps in the solution, i.e. both matched substring have the same length. **Example**: inputs "ABCAABCAA" and "ABBCAACCBBBBBB". The substring "CAABC" that starts at position 3 in the first string and position 4 in the second, has a score $2 * w_C + 2 * w_A - \delta$ since the B is mismatched in the second string.
2. **Interval-based constant best approximation [50 points]**. Given a set of N points (x_i, y_i) with integer values for x_i between 1 and M and real values for y_i, find a partitioning of the interval $[1, M]$ into contiguous intervals such that the error of approximating points in each interval element by the average value of y in the interval is minimized. You need to add a penalty factor proportional to the total number of intervals the solution has. For example, if you have $x \in [1, 100]$ and you partition the X dimension in intervals $[1 - 10], [11, 20], .., [91, 100]$ the penalty is $10 * \delta$. **Hint:** determine separately the formula for the error of approximating a set of values y_i by their average; think about how you can compute this quantity incrementally to reduce the running time of the algorithm. For this problem, experiment with both an array and a hash (memorization) version of the solution and compare the actual memory usage for both.
3. **BONUS [30points]:** Generalize problem 2 to a 2D grid for the input, i.e. the input is of the form x_i, y_i, z_i with $x_i, y_i \in [1..M]$ and z_i a real number. You must both develop the theory and implement this version of the algorithm and test it similarly to the setup above.