1. Consider approximate pattern matching again. In Exercise 6.4 we located the start positions of approximate occurrences by tracing paths to the 0-row of the dynamic programming matrix. This traceback could be avoided: Extend the approximate matching algorithm to compute, for each \( j \) that is the end position of some approximate occurrence, a start position \( s \) such that \( T[s \ldots j] \) is the shortest approximate occurrence of \( P \) that ends at \( T[j] \). The modification should not increase the asymptotic complexity of the algorithm.

2. Present the Smith-Waterman dynamic programming table for computing optimal local alignments between the strings “AINOAT” and “NOKIA”. Score matches by +2, and mismatches and insertions/deletions by −1. What is the optimal local alignment of these strings?

3. Show that a center string can be found in polynomial time. What is the complexity? What does this give as the total complexity of computing a multiple alignment by the center star method?

4. FASTA and BLAST exclude from closer examination those database sequences that do not contain exact occurrences of “\( k \)-tuples” or “words”. Such a strategy can be formally justified under a unit-cost scoring scheme which only penalizes the number of single-character differences (mismatches, insertions and deletions): Let a pattern \( P[1 \ldots n] \) be broken into \( d + 1 \) disjoint blocks \( B_1 = P[1 \ldots i_1], B_2 = P[i_1 + 1 \ldots i_2], \ldots, B_{d+1} = P[j_d + 1 \ldots n] \). Then a substring \( T' \) of text \( T \) can be an approximate occurrence of \( P \) with at most \( d \) differences only if some block \( B_h \) has an exact occurrence in \( T' \). Explain why this holds.

5. Fill and return the course feedback form, which will be available at the course homepage.