Homework Assignment No. 4

CSCI 4470/6470 Algorithms, CS@UGA, Fall 2019

Due Monday October 21, 2019

Five questions with total **120** *points.*

Homework submission should be in hardcopy. If for a reason an email submission is necessary, it needs an approval from this instructor. Submission needs to be received by 5:00pm on the due day.

1. (**20 points**) Consider the LCS problem that finds the longest common subsequence (lcs) between two input sequences $x_1x_2 \dots x_n$ and $y_1y_2 \dots y_m$. Define objective function $B(i, j)$ to be length of an lcs between **suffixes** $x_ix_{i+1} \dots x_n$ and $y_jy_{j+1} \dots y_m$.

   Please give a recursive formula to compute $B(i, j)$ in the "backward" fashion. What are the base cases for $B(i, j)$?

2. (**30 points**) Consider the following problem Max Sequence Matching as an extension to the LCS problem. Between two sequences $x_1x_2 \dots x_n$ and $y_1y_2 \dots y_m$, a *matching* is a choice of two sets of indexes $1 \le i_1 < i_2 < \cdots < i_k \le n$ and $1 \le j_1 < j_2 < \cdots < j_k \le m$, for some $k \ge 0$, such that

$$\sum_{r=1}^{k} \mu(x_{i_r}, y_{j_r}) = \mu(x_{i_1}, y_{j_1}) + \mu(x_{i_2}, y_{j_2}) + \cdots + \mu(x_{i_k}, y_{j_k})$$

   called *matching score*, achieves the maximum. Here $\mu(w, z)$ is a given predefined numerical score function between any two symbols w and z. For example, when $\mu(w, z) = 1$ for $w = z$ and $\mu(w, z) = 0$ for $w \ne z$, this problem becomes the LCS problem. In general, however, $\mu(w, z)$ may take positive and negative values depending on symbol w and z.

   For example, two sequences `relativity` and `reality` may have the following matchings

```
rela tivity        relativity          re lativity
re al i  ty        real i  ty          real  ity
```

with different matching scores. To solve this problem with dynamic programming, we define objective function $S(a, b)$ to be the maximum matching score between prefixes $x_1x_2 \ldots x_a$ and $y_1y_2 \ldots y_b$.

Please give a recursive formula to compute $S(a, b)$ in the "forward" fashion, including base cases. You can assume that score function μ is already given along with the two input sequences.

Note: Because pairwise score is given by μ, which may be arbitrary, even if the two "tails" x_i and y_j are the same symbol, matching them may not necessarily lead to a globally optimal solution.

3. (**30 points**) Consider the following problem called MAX PARENTHESIZATION: given an input sequence $x_1x_2 \ldots x_n$ consisting of four symbols `(`, `)`, `[`, and `]`, find a subsequence that is a legal parenthesis expression with the maximum number of paired parentheses. For example, for input sequence `](([])(][)(]`, the following subsequences are all legal parenthesizations

`()()` `([])` `(())[]` `[][]` `(())()`

where the 3rd and 5th expressions have the most paired parentheses.

A legal parenthesization expression can have paired parentheses in *parallel* and/or *nested* patterns. For example, the 1st and 4th expression examples have two *parallel* paired parentheses, respectively. The 2nd has two nested pairs while the 3rd and 5th have both nested and parallel pairs.

To approach this problem with dynamic programming, we define the objective function $P(i, j)$, for $1 \leq i \leq j \leq n$, to be the maximum number of paired parentheses in a legal expression as a subsequence of the input.

Please give a recursive formula to compute $P(i, j)$ in the "inside" fashion. For this problem, an "inside" solution should exhaustively examine the following situations: (1) would x_i and x_j form a legal pair? if so, should they definitely be paired? (2) if x_i and y_j form a pair, what is reduced subproblem and corresponding objective function? (3) will the expression be formed by two parallel expressions? if so, where is the joining position and what are the reduced subproblems and corresponding objective functions?

4. (**20 points**) Though the LCS problem is solved with dynamic programming, there is a **greedy strategy** applied to the situation when the two tails x_i and y_j are the same symbol. The LCS algorithm introduced in the class always retain the common symbol as a part of the lcs of prefixes $x_1x_2 \ldots x_i$ and $y_1y_2 \ldots y_j$. Use the *swapping method* to prove that this strategy does not compromise the optimality of the computed lcs.

5. (**20 points**) Michael drives from city A to city B along US Route 1 and he decides to stop at as few gas stations as possible. His strategy is to only stop at a gas station when the remaining gas in the tank is not enough to reach the next gas station. Use the *swapping method* to prove that Michael's strategy does allow him to stop at the fewest gas stations during his trip.

The answers must be word-processed or typed. You may substitute formulae and figures with hand-writings. Your submitted algorithms should be in the pseudo-code, not in any specific programming language. Answers deviating from these requirements will be returned without grading.

The answers must be the student's own work. Idea sharing and referencing to others' work (including those online) are not allowed. Plagiarism and other forms of academic dishonesty will be handled within the guidelines of the Student Handbook and reported to the University.