Homework Assignment No. 3

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

Due Monday February 25, 2018

Four 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. (30 points) Consider the following problem Minimum Vertex Cover:

   Input: a graph $G = (V, E)$;
   Output: a subset $C \subseteq V$ such that
   (1) $\forall (u, v) \in E$, either $u \in C$ or $v \in C$;
   (2) $|C|$ is the minimum.

   The subset $C$ of vertices is called a vertex cover, which covers all edges in the graph.

   Use the search tree method to construct recursive, an non-naïve exhaustive search algorithm for problem Minimum Vertex Cover which has time complexity $O(1.5^n)$.

2. (30 points) Consider the Casino Dice Decoding Problem (ref. Chapter 15, Example-2 in lecture note 3, NewNote3.pdf). Note that the problem resembles the Assembly line problem. Fill out a dynamic programming table that computes functions $p(S, i, F)$ and $p(S, i, L)$ for sequence of digits $S = 6665432$. You can choose to write a program to or manually calculate for the table. Show how the computation can tell the most probable hidden sequence of dice that may have been used to roll the given 7 digits.

3. (20 points) Fill out a dynamic programming table that computes for the 0-1 knapsack
problem on the following input data: \( B = 10, i = 1, 2, 3, 4 \), and
\[
\begin{align*}
  s_1 &= 2, v_1 = 10 \\
  s_2 &= 3, v_2 = 40 \\
  s_3 &= 5, v_3 = 40 \\
  s_4 &= 3, v_4 = 50
\end{align*}
\]

4. (40 points) Consider following Article Grading problem. A teacher grades students’ articles based on number of letter corrections on an article. The teacher may correct an article with the following four types of editing and reward/penalty of points:
\[
\begin{align*}
  &\text{a letter is kept} & +5 & \text{points} \\
  &\text{a letter is substituted} & -2 & \text{points} \\
  &\text{a letter is inserted} & -3 & \text{points} \\
  &\text{a letter is deleted} & -3 & \text{points}
\end{align*}
\]

Given an original article \( X \) and its edited version \( Y \), there may be more than one way to interpret how the article \( X \) has been edited into \( Y \). You are required to formulate a dynamic programming solution that can find the “best way” to compare \( X \) and \( Y \). That is, your algorithm can compute the maximum number of points gained through an editing process identified from \( X \) and \( Y \). This problem can be thought as a generalization of the LCS problem (where the situation of "a letter is kept" is rewarded 1 point while no penalty applies on a letter "substitution", "insertion", or "deletion").

Let the original article and edited version be \( X \) and \( Y \), respectively. Follow the idea used in the problem analysis for LCS, consider any prefix \( x_1 \ldots x_i \) of \( X \) and any prefix \( y_1 \ldots y_j \) of \( Y \). Assume \( \text{Grade}(i, j) \) to be the maximum score gained when prefix \( x_1 \ldots x_i \) has been edited to \( y_1 \ldots y_j \). There are four possible editing actions between letter \( x_i \) and letter \( y_j \):

1. \( x_i \) is kept as \( y_j \) (only if \( x_i = y_j \));
2. \( x_i \) was substituted by \( y_j \) (only if \( x_i \neq y_j \));
3. \( y_j \) was inserted after the position \( x_i \) in the edited version;
4. \( x_i \) was deleted (\( x_i \) will not appear in the edited version anymore);

**Part A.** Based on these 4 cases, you are required to give recurrence formulate for \( \text{Grade}(i, j) \), i.e., to define it with the same function \( \text{Grade} \) (with smaller parameter values, however). Also give the base cases.
Part B. Write an iterative algorithm to compute function $Grade$; the algorithm needs to include steps to remember the choices of editing actions.

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.