University of Waterloo
CS 341 — Algorithms
Spring 2014
Assignment 5

Due: Wednesday, July 30, 2014, at 3:00pm (Last day of classes).

Assignment Guidelines

- Use A5-coversheet.pdf for the first page of your assignment or format the first page of your assignment in *exactly* the same manner.
- Answers the questions in the same order that they are given in the assignment.
- Make it very clear to the marker where the answer to one question ends and the answer to the next one begins.
- You may lose up to 20% on a question because it is difficult to read or difficult to understand.
- You may only use the techniques discussed in class.
- Your whole assignment is considered late if either the programming question or the written portion is submitted late.

Assignment Questions

1. **[15 points]** You are planning a round-the-world trip which will involve visiting n cities: $0 \to 1 \to 2 \to \cdots \to n$. For traveling between city i and $i+1$ ($0 \leq i < n$) you need to choose between two modes of transportation: train or plane. You are starting at the train station of city 0 and want to end up at the airport of city n. The goal is to minimize the cost of the trip. The cost of transferring between the train station and the airport of city i, either direction, is given by b_i, $0 \leq i \leq n$. The cost of traveling from city i to city $i+1$ via train is t_i and by plane p_i, $0 \leq i < n$.

   You prefer taking the train but your traveling companion prefers flying. As a result of a compromise, an additional constraint to the optimization problem is that you want to **use the plane exactly $n/2$ times**. Assume that n is divisible by 2.

   Design an $O(n^2)$ dynamic programming algorithm that finds the cost of an optimal solution to the travel planning problem. Your answer to this part of the question should consist of an elaboration of each of the following four points:

   - Optimal substructure.
     What is the optimal substructure that your algorithm is taking advantage of?

- Subproblem definition.
  Give a precise definition of what the subproblems are.
- Recurrence relation.
  Derive a recurrence relation on the optimal solutions of the subproblems.
- Compute optimal solutions.
  Indicate briefly how the optimal solutions to the subproblems can be computed in a bottom up fashion.

2. [**15 points**] One of the most ubiquitous operations in computer algebra is to compute the greatest common divisor (GCD) of a collection of positive integers. A question that arises is whether there exists a small cardinality subset of the numbers that has the same GCD as the entire set. Formally:

   SUBSET-GCD-DEC
   Instance: A set S of positive integers and an integer k.
   Question: Does there exist $S' \subset S$ with $\mid S' \mid \leq k$ such that GCD(S') = GCD(S)?

   For example, consider the collections $S_1 = \{42, 30, 70\}$ and $S_2 = \{91, 104, 95\}$. In both cases we have GCD(42, 30, 70) = GCD(91, 104, 95) = 1, but for S_2 we have GCD(91, 95) = 1, whole for S_1 no proper subset has GCD equal to one.

   Prove that SUBSET-GCD-DEC is NP-complete.

   Hint: Reduce VERTEX-COVER-DEC to SUBSET-GCD-DEC.

3. [**15 points**] Allocating Paintings to Art Galleries
   This question explores the complexity of allocating paintings to art galleries. Specifically, there are t paintings, denoted $P_1, \ldots, P_t$, that a collector wishes to allocate to K art galleries, denoted $G_1, \ldots, G_k$. There are s art critics, denoted $A_1, \ldots, A_s$ and each critic A_i has selected a subset of paintings $Q_i \subseteq \{P_1, \ldots, P_t\}$ that he or she wishes to view.

   The collector is rather eccentric and wishes to ensure that each critic A_i is forced to visit at least two galleries in order to view all of the paintings in Q_i.

   One way to formulate the GALLERY-ALLOCATION problem is as follows:

   Input: a set $P = \{P_1, \ldots, P_t\}$, s subsets $Q_1, \ldots, Q_s \subseteq P$ and a positive integer $K \geq 2$.
   Output: "Yes" if and only if there exists a function $f : P \longrightarrow \{1, \ldots, K\}$ such that $\mid \{f(P_j) : P_j \in Q_i\} \mid > 1$ for all i, $1 \leq i \leq s$.

   (a) Prove that the GALLERY-ALLOCATION problem is in NP by defining a suitable certificate and then giving a polynomial-time certificate verification algorithm.

   (b) Prove that the GALLERY-ALLOCATION problem is NP-complete, by a polynomial transformation from 3-CNF-SAT. Here are some hints to get you started:
   The Reduction. We are given an instance of 3-CNF-SAT, i.e., a 3-CNF Boolean formula F using n variables $x_1, \ldots, x_n$, having m clauses $C_1, \ldots, C_m$. We want to

construct an instance of GALLERY-ALLOCATION, which is defined by a set P, a collection of subsets of P, and an integer $K \geq 2$. Define $P = \{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}, z\}$ where z is a new element. For each clause C_i, create a subset Q_i consisting of the three literals in C_i, along with z. In addition, for each $i = 1, \ldots, n$, we create a subset $Q_{m+i} = \{x_i, \overline{x_i}\}$. Finally, we choose $K = 2$.

4. [**15 points**] Create an efficient program, in C++, that implements the dynamic programming algorithm for Longest Common Subsequence (LCS). LCS is described briefly in the course slides but in more detail on slides 223–234 of the Alternate version of the course slides linked here: `http://www.student.cs.uwaterloo.ca/~cs341/slides/cs341slides_Alt.pdf` and also on pages 390–397 (Section 15.4) of the course textbook.

   The input to your program is two sequences of (0 or more) strictly lower case letters. Each sequence is given on a separate line and the two sequences are followed by a blank line. Each sequence is at most 64K characters.

   For example, given the following input in stdin:

   ```
   bacboambpuabtabera
   xcomyxyxpyxuyxtyerx
   ```
   ← *blank line*

   Your program should output to stdout, the number of matching characters in the longest common subsequence on one line followed by the characters themselves on a second line. For the above example, the output is:

   ```
   8
   computer
   ```

   We will only test your program using valid input as described above.

   Your program will be marked on the clarity and efficiency of the source code and correctness (determined by the performance of your code on test cases).