Homework set 2

Note: the homework sets are not for submission. They are designed to help you prepare for the quizzes.

1. Given a string X, we denote by $X[i]$ the ith character of X. Given an n-character string A, and two additional strings B and C, we say that string A is an *interleaving* of strings B and C iff we can partition the set $\{1,\ldots,n\}$ of indices into two disjoint subsets $I = \{i_1, i_2, \ldots, i_k\}$ and $J = \{j_1, j_2, \ldots, j_{n-k}\}$, where $i_1 \leq i_2 \leq \cdots \leq i_k$ and $j_1 \leq j_2 \leq \cdots \leq j_{n-k}$ such that:

   - $I \cup J = \{1,\ldots,n\}$
   - the string $(A[i_1], A[i_2], \ldots, A[i_k]) = B$, and the string $(A[j_1], A[j_2], \ldots, A[j_{n-k}]) = C$

   In other words, A is obtained by interleaving the characters of B and C. Design an efficient algorithm, that, given as input strings A, B and C, decides whether A is an interleaving of B and C. Prove the algorithm's correctness and analyze its running time.

2. Suppose we are given a steel rod of length n. Assume also that we are given, for each $1 \leq i \leq n$, price $p(i)$ that a rod of length i sells for. Given that we can make cuts for free (and that we only cut integer lengths), provide an algorithm that efficiently computes the maximum profit that we can make by cutting up and selling our rod of length n.

3. We say that a set $A \subset \{1, 2, \ldots, n\}$ is *good* iff for all $1 \leq i \leq n-2$, among the numbers in the triple $T_i = (i, i+1, i+2)$, either one or two members of T_i belong to A. For example, the set $A = \{1, 2, 4, 5\} \subset \{1, 2, \ldots, 6\}$ is good, set $B = \{4, 5\} \subset \{1, 2, \ldots, 6\}$ is not good (none of the members of triple $\{1, 2, 3\}$ belong to B), and $C = \{2, 3, 4, 6\} \subset \{1, 2, \ldots, 6\}$ is not good (all members of triple $\{2, 3, 4\}$ belong C). Using dynamic programming, design an algorithm, that, given n, and a sequence $w_1, \ldots, w_n$ of non-negative numbers, finds a good subset $A \subset \{1, 2, \ldots, n\}$ that maximizes the sum $\sum_{i \in A} w_i$. The running time of the algorithm should be polynomial in n. Describe your algorithm in detail, prove its correctness, and analyze the running time.

4. We are given a binary counter with k bits, that supports the following two operations:

   - INCREMENT: increase the value of the counter by 1 (modulo 2^k).
   - RESET: set the value of the counter to 0.

   Show how to implement the counter as an array of bits, so that any sequence of n INCREMENT and RESET operations takes $O(n)$ time on an initially zero counter. Hint: keep a pointer to the high-order 1.