**6331 - Algorithms, Autumn 2016, CSE, OSU**
**Homework 4**
**Instructor:** Anastasios Sidiropoulos

**Problem 1.** A *palindrome* is a sequence $Y = y_1, \ldots, y_{2k}$, for some integer $k$, such that for all $i \in \{1, \ldots, k\}$ we have $y_i = y_{2k+1-i}$; that is $y_1 = y_{2k}$, $y_2 = y_{2k-1}$, $\ldots$, $y_k = y_{k+1}$. For example, the sequence $1, 2, 3, 1, 1, 3, 2, 1$ is a palindrome, while the sequence $1, 2, 3, 1$ is not.

Design an algorithm which given a sequence $X = x_1, \ldots, x_n$ computes the longest subsequence of $X$ that is a palindrome. The running time of your algorithm should be polynomial in $n$ (e.g. $O(n^2)$).

**Problem 2. Indiana Jones and the temple of doom:** Let $A[1 \ldots n, 1 \ldots n]$ be an integer array with $n$ columns and $n$ rows. The array encodes the map of a room of dimensions $n \times n$. We have $A[i, j] = 1$ if there is a gold coin at location $(i, j)$ in the room, and otherwise $A[i, j] = 0$. Indiana Jones is initially located at position $(1, 1)$, which is the north-west corner of the room, and has to reach the exit at location $(n, n)$, which is the south-east corner. Indiana Jones is in a hurry, so he can move only south, or east. That is, at each step, starting at location $(i, j)$ he can either move to location $(i + 1, j)$, or $(i, j + 1)$. Note that he cannot move diagonally.

Design an algorithm that given the array $A$ as input, outputs a sequence of moves for Indiana Jones, that allows him to collect the maximum possible number of gold coins before leaving the room. The running time of your algorithm should be polynomial in $n$.

**Problem 3.** Alice and Bob play the following game. They start with a pile of $n$ sticks. Alice plays first, and takes $k$ sticks from the pile, for some integer $1 \leq k \leq \lfloor n/2 \rfloor$. Bob plays next, and he also picks at least one, and at most half of the remaining sticks. The two players keep alternating until there is only one stick left. At that point, the player who has to play next is the loser.

Here is an example:

- Initially, there are $n = 20$ sticks.
- Alice picks 10 sticks, so there are 10 left.
- Bob picks 1 stick, so there are 9 left.
- Alice picks 4 sticks, so there are 5 left.
- Bob picks 2 sticks, so there are 3 left.
- Alice picks 1 stick, so there are 2 left.
- Bob picks 1 stick, so there is 1 left, and Alice loses.

Describe an algorithm that given $n$, outputs an integer $k$, such that the best strategy for Alice is to pick $k$ sticks. That is, if it is always possible for Alice to win when she starts with $n$ sticks on the pile, then it is never possible for Bob to win when he starts with $n - k$ sticks on the pile.

Your algorithm should have running time polynomial in $n$.

Hint: Using dynamic programming, compute the optimal value for $k$ for increasing values of $n$.