COMP 271 Design and Analysis of Algorithms
2003 Spring Semester
Extra Credit problem – Dynamic programming
Distributed: April 14, 2003

This is an extra credit problem and does not have to be done. Solutions returned by April 18, 2003 will be counted. Solutions can be emailed or given to the instructor or left at the main computer science department office with instructions to be put in the instructor's mailbox.

Background:

In this problem you are given a *coin game* and need to find whether a position is a "winning" or "losing" one.

The game starts with three piles of coins.

A game *position* (P_1, P_2, P_3) specifies that there are P_i coins in pile i, $i = 1, 2, 3$.

In a *legal move* a player chooses a pile and removes at least one coin from that pile. More formally, a *legal move* from position $(P_1, P_2, \ldots, P_3)$ chooses one pile i and a number of coins $1 \leq X \leq P_i$ and removes X_i coins from pile i. Note that if $P_i = 0$ then pile i is empty and that pile can not be chosen.

The players alternate *moves*. That is, the first player makes a legal move, then the second makes a legal move, then the first, then the second, etc..

The player who takes the last coin *loses*. Note that each move removes at least one coin so some player must lose. The player who doesn't lose, *wins*.

(P_1, P_2, P_3) is a *winning position* if the player whose turn it is can make a move that guarantees that he/she can win, no matter what the other player does for the rest of the game. (P_1, P_2, P_3) is a *losing position* if it is not a winning position.

Examples:

$(1, 1, 1)$ is a losing position since at each move each player must take exactly one coin. So, the player whose turn it is now must also take the last coin.
$(3, 1, 1)$ is a winning position because the current player can take two coins from pile 1, leaving $(1, 1, 1)$.

Problem 1 Describe a dynamic programming algorithm that, given position (P_1, P_2, P_3), tells whether (P_1, P_2, P_3) is a winning or losing position. Prove the correctness of your algorithm and analyze its running time.

Problem 2 Generalize the rules of the game so that instead of three piles there are now k piles. A position is a k-tuple $(P_1, P_2, \ldots, P_k)$. The rules of the

game stay the same, i.e., players alternate turns, at each turn taking coins from exactly one pile. The player taking the last coin loses.

Describe a dynamic programming algorithm for the generalized game that, given position (P_1, P_2, P_k), tells whether (P_1, P_2, P_k) is a winning or losing position. Prove the correctness of your algorithm and analyze its running time.

Problem 3 Change the rules of the game from problem 2 so that the player who takes the last coin now *wins* instead of loses. All other definitions remain the same.

Now give a dynamic programming algorithm for the generalized game that, given position $(P_1, P_2, \ldots, P_k)$, tells whether $(P_1, P_2, \ldots, P_k)$ is a winning or losing position. Prove the correctness of your algorithm and analyze its running time.