# Assignment 8. **Dynamic Programming.**

Given March 25, due Monday, April 6.

**1.** A very simple two person game is played as follows. There is an $N \times N$ array of rewards. $P(i,j)$ is the reward you get for landing at square $(i,j)$. Neither $i$ nor $j$ is allowed to go below 1 or above $N$. There are two players, called $A$, and $B$, who take turns moving. Player $A$ is allowed to replace $i$ by $i+1$ or $i-1$, but may not change $j$. Player $B$ is allowed change $j$ to $j+1$ or $j-1$ but is not allowed to change $i$. After each move, a player collects the reward $P(i,j)$(the $(i,j)$ that the player has just moved to). The game continues until each player has made $L$ moves. Each player wants to maximize the sum of his or her payouts but doesn't care what the other player gets. Each player assumes that the other player will make the optimal move, for that player, in any given situation. For example, starting at $(7,7)$, $A$ may move to $(6,7)$ and collect $P(6,7)$. Then $B$ may go to $(6,8)$ and collect $P(6,8)$. Then $A$ may go to $(7,8)$ and collect $P(7,8)$, for a total of $P(6,7)+P(7,8)$, and so on.

**a.** Suppose we know $S(i,j,k)$, which is the total reward that $A$ will get if there are $k$ moves left starting from $(i,j)$ if both players use their optimal strategies, and $T(i,j,k)$ which is the total reward that $B$ will get, if it is $B$'s turn and there are $k$ moves left. This is known for all $(i,j)$. Show how to compute $S(i,j,k+1)$ (also for all $(i,j)$) in terms of this information.

**b.** Use this to find an efficient way for $A$ or $B$ to compute their optimal moves.

**2.** (from a book, where the solution may also be found) A certain string processing language allows the programmer to break a string into two pieces. Since this involves copying the old string, it costs $n$ units of time to break a string of $n$ characters. Suppose a programmer wants to break a string into many pieces. For example, suppose we wish to break a 20 character string after characters 3, 8, and 10. If the breaks are made in left to right order, then the first break costs 20 units, the second 17 units, and the third 12 units, for a total of 49 units. Clearly, the order in which the breaks are made can affect the total amount of time used. If we break first at 10, then at 3 and lastly at 8, the work is $20+10+7=37$. Devise a dynamic programming method that determines the optimal order of breaking, given $k$ breaks at locations $l(1) < l(2) \cdots < l(k) < n$.

**3.** (from the same source) We have a list `int v[k];` of $k$ "coin values", which are positive integers. We want to write a positive integer, $x$ as a sum of coin values using the fewest coins. That is, we want to find an expression,

$$x = n_1 \cdot v_1 + \cdots n_k \cdot v_k \quad ,$$

with non negative integer coefficients $n_i \geq 0$, that minimizes the total number of coins, $n_1 + \cdots + n_k$. Find a dynamic programming way to do this. The "cost to go" function, $C(x)$, is the number of coins needed to make $x$. If we know $C(y)$ for all $y < x$ then we can find $C(x)$ by asking how to add one coin. What is the total work to compute $C(x)$. There may be values that cannot be reached through any coin combination. for example, if all the $v_j$ are even numbers. Your algorithm should be efficient when $k$ is not too large.

**4.** Stenographers have symbols for commonly used words as well as for all the letters in the alphabet. We want to use a similar idea for compression of files that consist of sequences of characters. Each character has a unique integer that codes it. Some words also have their own integers. For example, we might transmit the letter *a* as 1 and the word *and* as the number 67. Then, we could transmit the word "and" by sending the sequence $(1, 14, 4)$ (a sequence of 3 numbers), or by sending the single number 67. Suppose "do" is assigned the number 68. Then, we could transmit the word "ando" by sending $(1, 14, 4, 15)$, (4 numbers), or $(67, 15)$ (2 numbers), or $(1, 14, 68)$ (3 numbers). Our goal is to find the representation of a long character string with the shortest integer sequence. We will represent the code in the form of a trie. A path through the trie is a sequence of characters followed by an integer, which is the code for that sequence. Every individual character has a code. Find a dynamic programming algorithm that computes the shortest representation of a given sequence. The algorithm might work in two stages. First compute the number of integers in the shortest representation of the final substring going from character $k$ to the end of the string, character $n$ (using dynamic programming). Then use this information to compute the shortest representation.