Problem 7.2

Recurrence relation for Currency Exchange

Let CE give the maximum amount of money in Wesels that can result from n transactions starting with an initial amount of Wesels a. Let CE have access to a table of exchange rates with the dimensions $j \times j$. The value for i passed into CE is the index of the row in the *rates* table corresponding to the type of currency of the amount passed in. The Wesel row and column are both at index $i = 0$.

$$CE(n, a, i) = \begin{cases} a & \text{if } n = 0 \\ (a)rates(i)(0) & \text{if } n = 1 \\ max(CE(n-1, (a)(rates(i)(0)), 0), & \\ \qquad CE(n-1, (a)(rates(i)(1)), 1), \ldots, & \\ \qquad CE(n-1, (a)(rates(i)(j-1)), j-1)) & \text{otherwise} \end{cases}$$

Analysis for naive Currency Exchange

A naive implementation of the above recurrence would have a runtime on the order of $O(j^n)$, where, as mentioned above j is the dimension of the *rates* table (also the number of currencies) and n the number of transactions. The reason for this is that at each recursive step of CE j recursive calls are made, resulting in a j-ary branching tree of depth $n + 1$. At each recursive step, only constant operations are performed.

Analysis for DP Currency Exchange

The DP solution has a runtime on the order of $O(j^2 n)$. Aside from a few constant steps of constant runtime, all of the work is done building a two-dimensional array to store intermediate maximum values. This array has dimensions $(n + 1) \times j$. At each of n iterative steps to build this array, j optimal values must be calculated, one for each type of currency. Each optimal value is calculated by taking the maximum value attained by converting the j optimal values from the previous iteration. Thus, each of n iterative step performs $j \times j$ operations.