Define an array of size (n+1) called OPT, and let OPT[i] be the optimal number of dollars through the i-th act. Also let OPT[0] = 0

Our base case is act 1, in which the optimal solution is m[1]. (since it is the only one possible to perform).
Thus OPT[1] = m[1].

Otherwise, to find the optimal dollars through the i-th act, we need to add the pay for the i-th act (m[i]) and the optimal pay through the (i-p[i]-1)th act (OPT[i-p[i]-1]).
In the case that (i-p[i]-1) <= 0, use 0 as the optimal value.

We need to minus one at the end because we cannot perform p[i] previous acts, not including the ith act.
For example, if p[9] = 4 then we need the optimal pay through the (9-4-1 = 4)th act, since we cannot perform in act 5, or (9-4). Thus we have to use the value of OPT[4] and not OPT[5].

In other words, OPT[i] = m[i] + OPT[max(0, i-p[i]-1)]

Finally, the optimal pay value for the final (nth) act is OPT[n].

There are O(n) cases and each of them takes O(1) time to fill in.
Therefore, the total time is O(n).