We define subproblems involving some of the last i days. Of course, having $Opt(i)$ denote the optimal division of stocks for the days i through n is hard, since we don't know how many stocks are available on the i-th day. So the logical continuation is to think of a two-parameter function $Opt(i, k)$ indicating the optimal division profit of k stocks starting from day i.

Here are two key observations to notice. First, from the i-th day on, our selling strategy does not depend on how selling was conducted on previous days because each of the remaining k stocks will incur the same *constant* penalty in profit due to the accumulated value of the function f from previous sales. So our total profit will just decrease by k times that constant.

The next observation is that if we decide to sell y stocks on i-th day, then all the k stocks will incur a profit loss of $f(y)$ because of this particular decision regardless of when they're sold.

Taking these two observations into account we are now ready to give the recurrence for $Opt(i, k)$. It is

$$Opt(i, k) = min_{y<=k}\{p_i y + Opt(i + 1, k - y) - kf(y)\}.$$

The first term of the above recurrence is the direct profit from selling y stocks with price p_i, the middle term is the optimal profit for selling the remaining $k \quad y$ stocks starting from day $i + 1$ (according to the first observation the optimal profit itself might differ from $Opt(i + 1, k - y)$ because of prior sales, but it will be within a fixed constant and therefore the choice of y minimizing the above recurrence will be unchanged), and finally the last term is due to the second observation above - the total profit loss incurred by the decision to sell y stocks on the i-th day on the remaining stocks.

An implementation would create a two dimensional table M, whose rows would stand for the days and the columns for the stocks to sell. The matrix would be filled using the above recurrence from the bottom row to the top one. The result is a cubic algorithm. Our desired goal is $Opt(1, x)$. The matrices could also keep track of how many we picked on i-th day in the optimal solution, and that will be our output.

Note that the running time of this algorithm polynomially depends on x, the total number of stocks. Note only is this permitted by the statement of the problem, but it is also polynomial in the input size, the values of the input function f are given as a set of x values $f(1), \ldots, f(x)$.

[1] ex571.682.218