Note on Rod Cutting

This is covered in §15.1. Here is Prof. Spencer’s view of it.

We can sell a rod of length \( I \) for \( P[I] \). These values are given to us. (Everything here is integral.)

Now we have a rod of length \( N \). How can we best cut the rod into pieces so as to maximize our revenue?

There are useful heuristics for this but here we give a method, a form of *dynamic programming* that gives the exact answer and does it in time \( O(N^2) \).

We create an array \( R[S] \) which will be the maximal total revenue we can get starting with a rod of length \( S \). While our goal is to find \( R[N] \) our method is to “work up” to this goal by finding \( R[1], R[2], \ldots \) until we reach \( R[N] \). We initialize with \( R[0] = 0 \). (If you like, set \( R[1] = P[1] \) as well.) So our program will start:

\[
\text{FOR } S = 1 \text{ to } N \quad (* \text{ Now want to find } R[S] *)
\]

We want (in the guts of the FOR loop) to find \( R[S] \) where we *already know* \( R[0], R[1], \ldots, R[S-1] \). The key is to think about the first cut of the rod. We don’t know where we should make it, it will be at some \( I \) where \( 1 \leq I \leq S \). (\( I = S \) would be selling the entire rod as a single piece.) *Suppose* we did cut it at \( I \) so we would receive revenue \( P[I] \) for the first piece. The remaining rod now has length \( S - I \). We would now (and this is a feature of dynamic programming) want to cut up that piece so as to get the maximal revenue but we *already know* that we will get \( R[S-I] \) from that piece. So then our total revenue would be \( P[I] + R[S-I] \). (Note that if we sell the rod of length \( S \) as a single piece we get \( P[S] + R[0] = P[S] \) so this is included.)

Which \( I \) should we choose for the first cut? Try them all! Pick that \( I \) which gives the maximal value of \( P[I] + R[S-1] \). Finding a max takes time \( O(S) \), with a single loop:

\[
\text{MAX} = 0
\]

\[
\text{FOR } I = 1 \text{ to } S
\]

\[
\text{IF } P[I] + R[S-I] \geq \text{MAX} \text{ THEN } \text{MAX} \leftarrow P[I] + R[S-I]
\]

\[
\text{END FOR}
\]

This \( \text{MAX} \) will be our value for \( R[S] \). Here is the whole program. It is a double loop and the time is \( O(N^2) \).
\( R[0] = 0 \)
FOR \( S = 1 \) to \( N \)
    \( MAX = 0 \)
    FOR \( I = 1 \) to \( S \)
        IF \( P[I] + R[S - I] \geq MAX \) THEN \( MAX \leftarrow P[I] + R[S - I] \)
    END FOR
    \( R[S] \leftarrow MAX \)
END FOR
RETURN \( R[N] \)

What if you want to actually find the optimal cut? When we are calculating \( R[S] \) we find that \( I \) which does maximize \( P[I] + R[S - I] \). We do this by having another array \( FIRSTCUT[S] \). We modify the calculation of \( MAX \) by:

\( MAX = 0 \)
FOR \( I = 1 \) to \( S \)
    IF \( P[I] + R[S - I] \geq MAX \) THEN
        \( MAX \leftarrow P[I] + R[S - I] \)
        \( FIRSTCUT[S] = I \)
    END IF
END FOR

In this approach \( FIRSTCUT[S] \) keeps changing but its last value (the one that sticks) is that \( I \) with \( P[I] + R[S - I] = MAX \).

Now to print out the cuts for \( N \) we (\( REM \) denotes the remaining part of the rod):

\( REM = N \)
WHILE \( REM > 0 \)
    PRINT \( FIRSTCUT[REM] \)
    \( REM \leftarrow REM - FIRSTCUT[REM] \)
END WHILE