CS3000: Algorithms & Data — Summer '25 — Laney Strange

Dynamic Programming: Rod Cutting

The algorithm below is a top-down method that uses Memoization to implement Dynamic Programming to solve the rod-cutting problem. It is similar to a recursive algorithm, except that we remember the solutions to smaller subproblems so that they need not be re-computed.

The rod-cutting problem seeks to find an optimal way to cut a rod of length n. We are given a list of prices p_i for $i = 1, 2, 3, .., n$, and we want to determine a way to cut the rod such that we maximize our total revenue r_n.

MEMOIZED-CUT-ROD(p, n)

```
1  let r[0 : n] be a new array
2  for i = 0 to n
3        r[i] = −∞
4  return MEMOIZED-CUT-ROD-AUX(p, n, r)
```

MEMOIZED-CUT-ROD-AUX(p, n, r)

```
 1  if r[n] ≥ 0
 2        return r[n]
 3  if n == 0
 4        q = 0
 5  else
 6        q = −∞
 7        for i = 1 to n
 8              q = max{q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n − i, r)}
 9  r[n] = q
10  return q
```

We typeset the procedures above with the following LaTeX:

```
\begin{codebox}
\Procname{$\proc{Memoized-Cut-Rod}(p, n)$}
\li let $r[0:n]$ be a new array
\li $\For \; i \gets 0 \To n$
\Do
\li $r[i] \gets -\infty$
\End
\li \Return $\proc{Memoized-Cut-Rod-Aux}(p, n, r)$
\end{codebox}

\begin{codebox}
\Procname{$\proc{Memoized-Cut-Rod-Aux}(p, n, r)$}
\li \If $r[n] \ge 0$
```

```
\Then
\li \Return $r[n]$
\End
\li \If $n == 0$
\Then
\li $q \gets 0$
\End
\li \Else
\li $q \gets -\infty$
\li \For $i \gets 1 \To n$
\Do
\li $q \gets$ max$\{q, p[i] + \proc{Memoized-Cut-Rod-Aux}(p, n-i, r)\}$
\End
\End
\li $r[n] \gets q$
\li \Return $q$
\end{codebox}
```