Rod cutting summary

- The problem: Given a rod of length n and a table of prices $p[i]$ for $i = 1, 2, 3, ..., n$, determine the maximal revenue obtainable by cutting up the rod and selling the pieces.
- Notation and choice of subproblem: We denote by $maxrev(x)$ the maximal revenue obtainable by cutting up a rod of length x. To solve our problem we call $maxrev(n)$.
- Recursive definition of $maxrev(x)$:

> **maxrev**(x)
>
> if ($x \leq 0$): return 0
>
> For i = 1 to x: compute $p[i] + \texttt{maxrev}(x - i)$ and keep track of max
>
> RETURN this max

- Why correct? tries *all* possibilities for first cut and recurses on the rest—which correct bec. of optimal substructure.
- Dynamic programming solution, recursive (top-down) with memoization:

> We create a table of size $[0..n]$, where $table[i]$ will store the result of $maxrev(i)$. We initialize all entries in the table as 0. We call $maxrevDP(n)$ and return the result.
> **maxrevDP**(x)
>
> if ($x \leq 0$): return 0
>
> IF $table[x] \neq 0$: RETURN $table[x]$
>
> For i = 1 to x: compute $p[i] + \texttt{maxrevDP}(x - i)$ and keep track of max
>
> $table[x]$ = max
>
> RETURN $table[x]$

Running time for $maxrevDP(n) : \Theta(n^2)$

- Dynamic programming, iterative (bottom-up):

> **maxrevDP_iterative(x)**
>
> create $table[0..n]$ and initialize $table[i] = 0$ for all i
>
> for $(k = 1; k \leq n; k++)$
>
> for $(i = 1; i \leq k; i++)$
> set $table[k] = \texttt{max}\{table[k], p[i] + table[k - i]\}$
>
> RETURN $table[n]$

Running time for $maxrevDP\_iterative(n) : \Theta(n^2)$

- Computing full solution (without storing additional information while filling the table):

> Input: The table $table[0..n]$ as computed above, where $table[i]$ stores the maxrev obtainable from a rod of length i.
> Output: the set of cuts corresponding to $table[n]$
> $curLength = n$
> while ($curLength > 1$) do:
>
> for ($i = 1; i \leq curLength; i++$)
>
> //is the value $table[curLength]$ achieved via a first cut of length i ?
>
> if $table[curLength] == (p[i] + table[curLength - i])$:
>
> output that a cut of length i was made
>
> $curLength = curLength - i$

Running time: $\Theta(n^2)$, no extra space

- Computing full solution (with storing additional information while filling the table):

In addition to $table[0...n]$ we use an array $firstcut[0..n]$ where $firstcut[i]$ will store the first cut in $maxrev(i)$. We can extend the algorithm for computing maxrevDP(x) (either recursive or iterative) to also compute $firstcut[x]$: basically if the maximum revenue for x is achieved with the first cut being of length k, we'll store that $firstcut[x] = k$.

> Input: The table $table[0..n]$ as computed above, where $table[i]$ stores the maxrev obtainable from a rod of length i. And $firstcut[0..n]$ where $firstcut[i]$ will store the first cut in $maxrev(i)$.
> Output: the set of cuts corresponding to $table[n]$
> $curLength = n$
> while ($curLength > 1$) do:
>
> output a cut of length $firstcut[curLength]$
>
> $curLength = curLength - firstcut[curLength]$

Running time: $\Theta(n)$, with $\Theta(n)$ extra space for $firstcut[0..n]$