Max Profit Limit \((m, l)\): Max profit you can get by cutting a rod of length \(m\) into \(\leq l\) pieces.

\((m=0, l=1)\)

Ultimately, we want to compute \(\text{Max Profit Limit} (n, k)\).

\[
\text{Max Profit Limit} (m, l) = \begin{cases} 
0 & \text{nothing to sell} \quad \text{if } m = 0 \\
\mathfrak{p}[m] & \text{if } l = 1, \text{ can't cut!} \\
\max_{1 \leq j \leq m} \left( \mathfrak{p}[j] + \text{Max Profit Limit}(m-j, l-1) \right) & \text{o.w., guess length of first piece.}
\end{cases}
\]
Store answers in $MPL[0..n,1..k]$. Pops use smaller $m+1$.

Save $m \leftarrow 0$ to $n$, $l \leftarrow 1$ to $k$.

$O(n)$ time per subproblem, $O(nk)$ problems

so alg will run in $O(n^2k)$ time.

\[ \text{MaxProfit} (P[1..n], k) : \]

\[
\begin{aligned}
\text{for } m &\leftarrow 0 \text{ to } n \\
\text{for } l &\leftarrow 1 \text{ to } k \\
\hspace{1cm} \text{if } m = 0, \quad MPL[m, l] &\leftarrow 0 \\
\hspace{1cm} \text{else if } l = 1, \quad MPL[m, l] &\leftarrow P[m] \\
\hspace{1cm} \text{else} \\
\hspace{2cm} \text{most} &\leftarrow -\infty \\
\hspace{2cm} \text{for } j &\leftarrow 1 \text{ to } m \\
\hspace{3cm} \text{if } \text{most} < P[j] + MPL[m-j, l-1] \\
\hspace{4cm} \text{most} &\leftarrow \text{most} \\
\hspace{1cm} &\text{return MPL}[n, k]
\end{aligned}
\]
3-Coloring Problem: Takes undirected

\[ G = (V, E) \]

Decide if we can partition \( V \) into three subsets \( (V_1, V_2, V_3) \)

\[
\begin{align*}
V_i \cap V_j &= \emptyset \quad \text{if } i \neq j \\
\text{but } V_1 \cup V_2 \cup V_3 &= V
\end{align*}
\]

such that the induced subgraph on each subset is complete.

\[
\text{(no restriction on edges between subsets)}
\]
Claim: 3-CGP is NP-hard.

Proof via reduction from 3-Color.

Let $G = (V, E)$ be an instance of 3-Color.

1) Compute $\overline{G}$, the complement of $G$, i.e., $uv \in E \iff uv \not\in \overline{E}$.

2) Return answer to 3-CGP on $\overline{G}$.

Takes $O(n^2)$ time.

Must prove $G$ has a proper 3-coloring if and only if 3-CGP returns Yes for $\overline{G}$. 
Suppose \( G \) has a proper 3-coloring with colors 1, 2, 3.

\[ V_\bar{w} := \text{all vertices with color } \bar{w} \text{ } \forall \bar{w} \in \{1, 2, 3\} \]

For any \( u \neq v \in V_{\bar{w}} \), \( uv \notin E \).

So, \( uv \in E \). So \( V_{\bar{w}} \) induces a complete subgraph.

Suppose \( \bar{G} \) has a 3-\( CGP \)

\[ \{V_1, V_2, V_3\} \]

Color each vertex of \( V_{\bar{w}} \) the color \( \bar{w} \).

For all \( u \neq v \in V_{\bar{w}} \) (\( \forall \bar{w} \in \{1, 2, 3\} \)), \( uv \in E \). So, \( uv \notin E \). So we made a proper 3-coloring. \( \square \)
Claim: $3$-CGP $\in \text{NP}$.

Proof: Given a Yes instance $G = (V, E)$, a good certificate/proof is the partition $V_1 \cup V_2 \cup V_3 = V$. Can verify each $V_i$ induces a complete subgraph in $O(|V|^2)$ time.

$\Rightarrow$ $3$-CGP is $\not\in \text{NP}$-complete.
Reminder: Select \([x[1..m], k]\) returns the \(k\)th element of \(x\) in sorted order.
(can be run in \(O(m)\) time.

\(\text{rank}\) is the position of an element in sorted order.

Given \(A[1..n]\) of distinct \#s, 
\(R[1..r]\) of distinct ranks.

Compute element of rank \(R[\bar{u}]\) in \(A\) for all \(\bar{u} \in \{1, \ldots, r\}^3\).

Can solve in \(O(n r)\) time by running Select \((A, R[\bar{u}])\) \(4\)-times.
Can you solve it in $O(n \log r)$ time?

Start by sorting $R[i]$ in $O(r \log r) = O(n \log r)$ time, procedure Compute Elements $\langle A[1..n], R[1..r] \rangle$.

If $r = 0$, return.

Compute element of rank $R \lceil \lfloor r/2 \rfloor \rceil$ by calling Select $\langle A[1..n], R[\lceil \lfloor r/2 \rfloor \rceil] \rangle$.

If $r = 1$, return.

Partition $A$ around $e$ in $O(n)$ time.

Compute Elements $\langle A[1.. R[\lceil \lfloor r/2 \rfloor \rceil] - 1], R[1.. \lceil \lfloor r/2 \rfloor \rceil - 1] \rangle$. 
Compute Elements \( A[R[R_{r/2}+1..n]] \), \( R[R_{r/2}+1..r] \) after decreasing each by \( R[R_{r/2}] \)

Time:

Spend \( O(n) \) time outside recursion.

\( T(n, r) \): running time on \( A[1..n], R[1..r] \).

Recursive calls have subarrays of \( A \) summing to \( n-1 \) or rank array about halves.

\[
T(n, r) = T(n_1, \frac{r}{2}) + T(n_2, \frac{r}{2}) + O(n) \quad (n_1 + n_2 = n - 1)
\]
Recursion tree:

\[ \text{root} = c_n \]
\[ c_n \]
\[ c_n_1 \]
\[ c_n_2 \]

\[ 3 = c_n \]
\[ 3 \leq c_n \]
\[ 3 \leq c_n \]

\[ \log r \text{ levels, so total time is } O(n \log r) \].

Proof of correctness: If \( r = 0 \) or \( r = 1 \), correctness is immediate thanks to base cases.
0 \leq r \leq 1.

We do compute element of rank $R[\Gamma_{\mathcal{L}}]$. Elements of ranks $R[1, \Gamma_{\mathcal{L}} - 1]$ come earlier in sorted order so it suffices to search ranks 1 to $R[\Gamma_{\mathcal{L}}] - 1$, which we do by induction on $n$.

Other elements have higher rank so it suffices to search in ranks $R[\Gamma_{\mathcal{L}} + 1, n]$, but note relative to subset those elements have rank decreased by
$R[\Gamma\Sigma]$. We find them recursively by induction on $n$, also.