CS41 Lab 7

The lab this week focuses on dynamic programming. The purpose of this lab is to gain practice using this technique to solve problems. This lab includes a coding portion!

Steel rods problem. Suppose you own a factory that produces long steel rods. Each rod you produce is n meters long, but it is often more profitable to cut the rod and sell the (shorter) pieces individually. The problem confronting you is: how do you cut the n-meter steel rod into pieces to maximize your revenue?

- **Input:** n (the length of one rod) and a list (of length n) of prices P where $P[i]$ is the revenue from selling a rod of length i.
- **Output:** Maximum possible revenue.

1. Consider a greedy approach which always chooses the cut k that maximizes the revenue $P[k]$. The idea is to make this cut, sell that rod, and then repeat with the remaining $(n-k)$-meter rod.

   Come up with a counterexample demonstrating that this greedy approach is not optimal.

2. Consider a greedy approach which always chooses the cut k that maximizes *revenue per meter* $\frac{P[k]}{k}$. The idea is to make this cut, sell that rod, then repeat with the remaining $(n-k)$-meter rod.

   Come up with a counterexample demonstrating that this greedy approach is not optimal.

3. Let's try to use dynamic programming for this problem.

   (a) What is the value of the optimal solution?

   (b) What structure of this problem makes dynamic programming a promising approach? Write an expression for the value of the optimal solution in terms of sub-problems. (Hint: try phrasing it in terms of the best place to make the left-most cut.)

   (c) What will you store in an array for a dynamic programming solution to this problem? Make a plan and sketch the algorithm for filling in values of this array.

   (d) Instead of writing pseudocode, let's write *actual* code to solve this problem!
   Retrieve the code from github for this lab.
   Implement a dynamic programming solution to the steel rod problem in the file `dp.cpp`.

   (e) Compare the runtime of your dynamic programming solution to the runtime of one of the brute-force solutions. Is it faster? By how much?

   Some files that you may find useful for reference:

   - `geninput.cpp` randomly generates an input to the steel rods problem.
   - `bruteforce.cpp` contains an implementation of a brute force solution.
   - `cutrod.cpp` contains a recursive brute-force-ish solution.
   - `cutrod2.cpp` contains a recursive, but slightly cleverer, brute-force-ish solution.