HW5

0810546 張理為

Top-down:

- 通過將問題遞歸地分解為更小的 rod length 的子問題並將這些子問題的結果存儲在一個 table 中，來遞歸地解決問題
- 使用遞歸函數 cut_rod 時，檢查 table 以查看是否已經計算了該長度的解答。如果存在，此函數直接返回預先計算的值。否則，它通過遞歸地考慮所有可能的切割方式來計算解答
- 避免重複計算同個子問題
- 此方法的缺點為對於較長的 rod length，可能會導致大量的函數調用和 stack 使用

Bottom-up:

- 在這種方法中，通過從最小的子問題 (rod length 為 0) 開始，逐步建立更大 rod length 的解答，直到達到所需的長度
- cut_rod 函數會遍歷所有可能的長度，從 1 到 n，通過考慮所有可能的切割方式來計算每個長度的最大利潤。結果存儲在 memo 數組中，該數組從較小的長度逐步填充到較大的長度。

- 避免使用 recursive function，且保證每個子問題只被求解一次，並將結果存儲供將來使用
- 這種方法通常可以使用更少的內存，因為 memorization table 是通過 iterate 填充的，而不是使用 call stack。且此方法還可以更好地控制內存使用量，通過預先分配數組大小。

Top-down 在實行方面較為直觀，因為它遵循問題的遞歸定義。然而，由於遞歸的緣故，使用此方法時需要仔細處理函數調用，並且可能具有較高的開銷。

Bottom-up 的動態規劃方法則避免了遞歸，通過自底向上的方式迭代計算解答。在時間和空間複雜度方面，它通常更高效，尤其是在解決較大問題的情況下。