Ejercicio - Rod-Cutting Bottom-Up

Integrantes

- Christian Echeverría 221441
- Gustavo Cruz 22779
- Josué Say 22801
- Mathew Cordero 22982
- Pedro Guzmán 22111

Descripción del Algoritmo

El algoritmo **Rod-Cutting Bottom-Up** resuelve el problema de corte de varillas utilizando un enfoque de programación dinámica de abajo hacia arriba. En lugar de resolver el problema de manera recursiva (lo que puede llevar a recomputaciones innecesarias), construye la solución de manera iterativa, almacenando los resultados previos en una tabla para evitar cálculos redundantes.

Parámetros de entrada

- `p`: Un array donde `p[i]` representa el precio de una varilla de longitud `i`.
- `n`: La longitud total de la varilla.

Pseudocódigo

```
Rod_Bottom_Up(p, n):
    # Definir el array r de tamaño n+1 para almacenar los beneficios máximos
    r = array[n + 1]

    # El beneficio máximo de una varilla con longitud 0 es 0
    r[0] = 0

    # Iterar sobre cada longitud de la varilla desde 1 hasta n
    para j desde 1 hasta n hacer:
        q = -∞  # Inicializar el beneficio máximo como menos infinito

        # Iterar sobre cada posible corte que se le pueda hacer a la varilla
        para i desde 1 hasta j hacer:
            q = max(q, p[i] + r[j - i])  # Calcular el beneficio máximo para cada longitud y
↪   cada corte
```

```
        r[j] = q  # Almacenar el beneficio máximo para la longitud j

    retornar r[n]  # Retornar el beneficio máximo encontrado para la longitud n
```

Explicación del Algoritmo

1. **Inicialización:** Se define un array `r` de tamaño `n+1` donde `r[i]` almacenará el beneficio máximo obtenido para una varilla de longitud `i`. Se inicializa con ceros, ya que el beneficio de una varilla de longitud `0` es `0`.
2. **Construcción iterativa:** Se recorren todas las longitudes de varilla desde `1` hasta `n`, calculando el beneficio óptimo para cada longitud.
3. **Cálculo del beneficio máximo:** Para cada longitud `j`, se prueban todas las posibles formas de cortar la varilla en dos partes `(i, j-i)`, tomando el máximo beneficio posible.
4. **Almacenamiento y retorno del resultado:** Se guarda el mejor beneficio encontrado en `r[j]`, y al final se devuelve `r[n]`, que representa el beneficio máximo para la longitud total de la varilla.