# Laboratorio 1 Rod Cutting


- Josue Marroquin 22397
- Sebastian Huertas 22295


## Repositorio

- https://github.com/xtsebas/RodCutrting

## Definición de la función para solucionar el problema

In [6]:
```python
def rod_cutting_fixed(prices, n):
    """
    Calcula la máxima ganancia posible al cortar una varilla de longitud n.
    :param prices: Lista de precios donde prices[i] representa el precio de la vari
    :param n: Longitud total de la varilla.
    :return: La máxima ganancia y la lista de cortes óptimos.
    """
    dp = [0] * (n + 1)  # dp[i] guarda la mejor ganancia para varilla de longitud i
    cuts = [0] * (n + 1)  # cuts[i] guarda la mejor primera longitud de corte para

    # Llenar la tabla DP
    for i in range(1, n + 1):  # Longitud de la varilla actual
        max_profit = -1  # Inicializamos en -1 para detectar valores correctos
        for j in range(1, min(i + 1, len(prices))):  # Posibles cortes
            possible_profit = prices[j] + dp[i - j]
            if possible_profit > max_profit:
                max_profit = possible_profit
                cuts[i] = j  # Guardar el corte óptimo
        dp[i] = max_profit  # Guardar el mejor precio encontrado

    # Reconstrucción de la solución: Lista de cortes óptimos
    length = n
    optimal_cuts = []
    while length > 0:
        optimal_cuts.append(cuts[length])
        length -= cuts[length]

    return dp[n], optimal_cuts
```

## Casos de test

In [7]:
```python
def test_rod_cutting_fixed():
    test_cases = [
        ([0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30], 4, 10),  # Caso básico
```

```
        ([0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30], 8, 22),  # Varilla larga
        ([0, 3, 5, 8, 9, 10, 17, 17, 20, 24, 30], 5, 15),  # Diferente tabla de pre
        ([0, 2, 5, 7, 8, 10, 15, 17, 20, 24, 30], 7, 17),  # Variación en valores
        ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 10, 10)       # Caso donde conviene no
    ]

    print("\nResultados de las pruebas del algoritmo Rod Cutting:")
    for i, (prices, n, expected) in enumerate(test_cases):
        result, cuts = rod_cutting_fixed(prices, n)
        individual_prices = [prices[cut] for cut in cuts]
        print(f"\n 🔹 Test {i+1}:")
        print(f"   - Precios disponibles: {prices}")
        print(f"   - Longitud de la varilla: {n}")
        print(f"   - Ganancia máxima esperada: {expected}")
        print(f"   - Ganancia máxima obtenida: {result}")
        print(f"   - Cortes realizados: {cuts}")
        print(f"   - Precios por cada corte: {individual_prices}")
        print(f"   - Suma total verificada: {sum(individual_prices)}")
        assert result == expected, f"❌ Error en el caso de prueba {i+1}"

test_rod_cutting_fixed()
```

Resultados de las pruebas del algoritmo Rod Cutting:

- Test 1:
  - Precios disponibles: [0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30]
  - Longitud de la varilla: 4
  - Ganancia máxima esperada: 10
  - Ganancia máxima obtenida: 10
  - Cortes realizados: [2, 2]
  - Precios por cada corte: [5, 5]
  - Suma total verificada: 10

- Test 2:
  - Precios disponibles: [0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30]
  - Longitud de la varilla: 8
  - Ganancia máxima esperada: 22
  - Ganancia máxima obtenida: 22
  - Cortes realizados: [2, 6]
  - Precios por cada corte: [5, 17]
  - Suma total verificada: 22

- Test 3:
  - Precios disponibles: [0, 3, 5, 8, 9, 10, 17, 17, 20, 24, 30]
  - Longitud de la varilla: 5
  - Ganancia máxima esperada: 15
  - Ganancia máxima obtenida: 15
  - Cortes realizados: [1, 1, 1, 1, 1]
  - Precios por cada corte: [3, 3, 3, 3, 3]
  - Suma total verificada: 15

- Test 4:
  - Precios disponibles: [0, 2, 5, 7, 8, 10, 15, 17, 20, 24, 30]
  - Longitud de la varilla: 7
  - Ganancia máxima esperada: 17
  - Ganancia máxima obtenida: 17
  - Cortes realizados: [1, 2, 2, 2]
  - Precios por cada corte: [2, 5, 5, 5]
  - Suma total verificada: 17

- Test 5:
  - Precios disponibles: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
  - Longitud de la varilla: 10
  - Ganancia máxima esperada: 10
  - Ganancia máxima obtenida: 10
  - Cortes realizados: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
  - Precios por cada corte: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
  - Suma total verificada: 10