**Experiment Number:** 03

# Comparing Divide & Conquer and Dynamic Programming

## Rod Cutting Problem

## Aim:

Given a rod of some length 'n' and the price associated with each piece of the rod, the rod has to be cut and sold. The process of cutting should be in such a way that the amount (revenue) obtained from selling is maximum. To apply, implement and compare the results of divide & conquer [$O(2^n)$] and dynamic programming approach [$O(n^2)$] in solving rod cutting problem.

## Algorithm(s):

### (a) Rod Cutting Problem – Divide & Conquer Approach

```
Algorithm CutRod_DC(P[1..n], n)
    If n=0 then
        Return 0
    End If

    maxRev = -∞
    For i←1 to n do
        Rev ← CutRod_DC(P, n-i)
        If (P[i]+Rev) > maxRev then
            maxRev ← P[i]+Rev
        End If
    End For
    Return maxRev
End CutRod_DC
```

### (b) Rod Cutting Problem – Dynamic Programming Approach

```
Algorithm CutRod_DP(P[1..n], n)
    Let R[0..n] be an array – to store revenue
    R[0] ← 0
    For j←1 to n do
        maxRev = -∞
        For i←1 to j do
            Rev ← R[j-i]
            If (P[i]+Rev) > maxRev then
                maxRev ← P[i]+Rev
```

```
                    End If
                End For
                R[j] ← maxRev
            End For
            Return R[n]
        End CutRod_DP
```

## Results & Discussion:

### Comparison Table

- Test both the algorithms for the input value n=5, 15 and 30 and show the number of active operations required in a table

### Comparison Chart

- Draw a Chart for the above-recorded data.

### Conclusion

1. By applying divide & conquer for solving rod cutting problem, the algorithm require $O(2^n)$ time complexity.
2. But, by applying dynamic programming approach, the algorithm requires only $O(n^2)$ time complexity.
3. But, when compared to divide and conquer, dynamic programming approach requires more spaces to store all solutions.