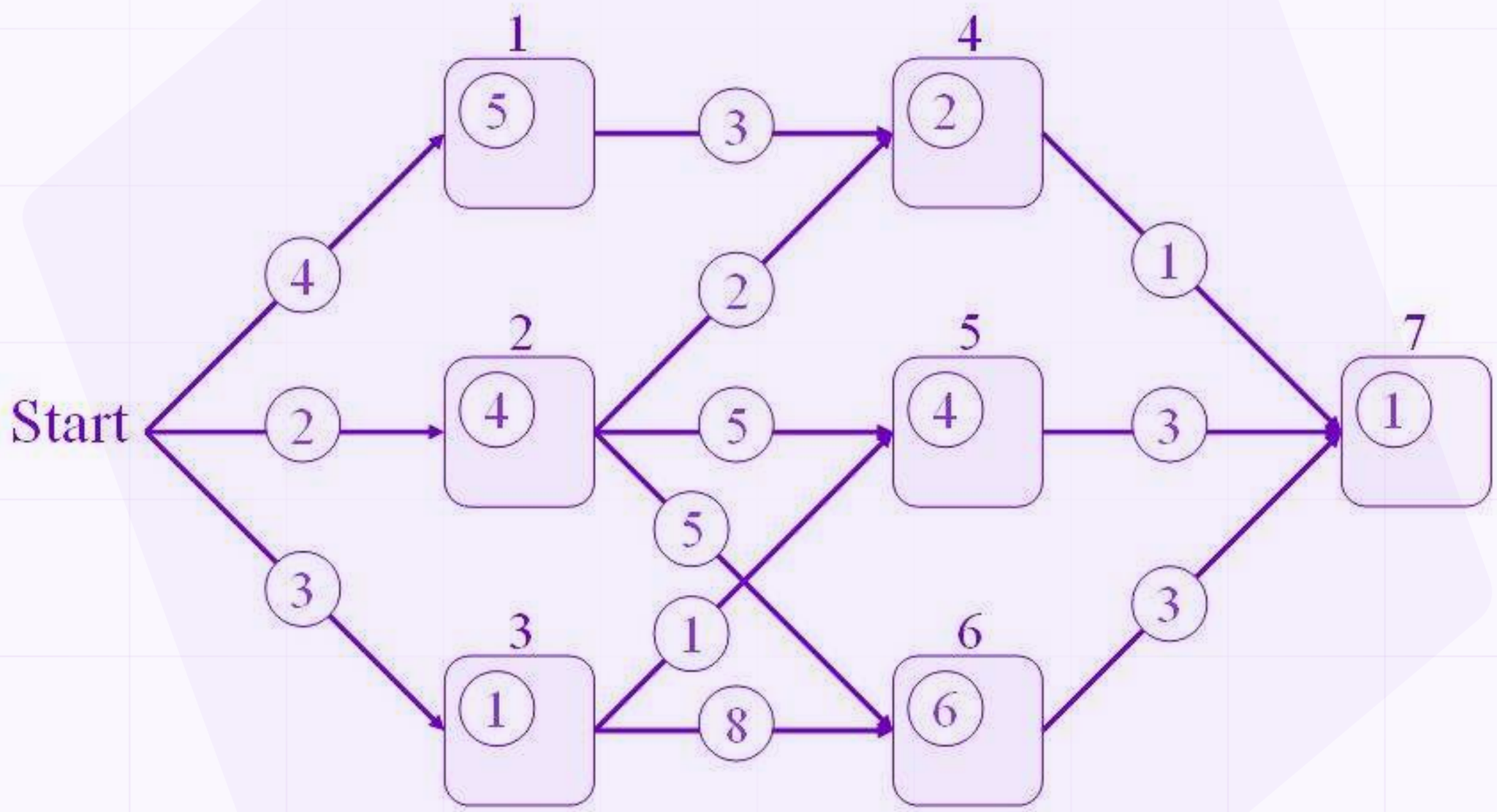
BOSSCODER ACADEMY

MASTER

DYNAMIC PROGRAMMING



Introduction to Dynamic Programming

Definition

Dynamic Programming (DP) is a method for solving complex problems by breaking them down into simpler subproblems. It is applicable when the problem can be divided into overlapping subproblems with optimal substructure.

Key Concepts

- **Overlapping Subproblems**
  The problem can be broken down into subproblems which are reused multiple times.
- **Optimal Substructure**
  The optimal solution to a problem can be constructed efficiently from optimal solutions of its subproblems.

Basic Concepts

- **Memoization**
  Top-down approach where we solve the problem by recursively solving subproblems and storing their results.
  - *Example:* Fibonacci sequence using memoization.

- **Tabulation**
  Bottom-up approach where we solve the problem by iteratively solving subproblems and storing their results in a table.
  - *Example:* Fibonacci sequence using tabulation.

```python
# Fibonacci sequence using Memoization
def fib_memo(n, memo={}):
    if n in memo:
        return memo[n]
    if n <= 2:
        return 1
memo[n] = fib_memo(n-1, memo) + fib_memo(n-2, memo)
    return memo[n]
# Fibonacci sequence using Tabulation
def fib_tab(n):
    if n <= 2:
        return 1
    table = [0] * (n+1)
    table[1], table[2] = 1, 1
    for i in range(3, n+1):
        table[i] = table[i-1] + table[i-2]
    return table[n]
```

Intermediate Concepts

- **0/1 Knapsack Problem**
  A classic DP problem where we determine the maximum value that can be obtained by putting items into a knapsack of a given capacity.
  - ***Problem Statement:*** Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

```python
def knapsack(values, weights, W):
    n = len(values)
dp = [[0 for _ in range(W + 1)] for _ in range(n + 1)]
    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                dp[i][w] = 0
            elif weights[i - 1] <= w:
                dp[i][w] = max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w])
            else:
                dp[i][w] = dp[i - 1][w]
        return dp[n][W]
```

- **Longest Common Subsequence (LCS)**
  Finding the longest subsequence common to two sequences.
  - ***Problem Statement:*** Given two sequences, find the length of their longest common subsequence.

```python
def lcs(X, Y):
    m, n = len(X), len(Y)
    dp = [[0 for _ in range(n + 1)] for _ in range(m + 1)]

    for i in range(m + 1):
        for j in range(n + 1):
            if i == 0 or j == 0:
                dp[i][j] = 0
            elif X[i - 1] == Y[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    return dp[m][n]
```

Advanced Concepts

- **Matrix Chain Multiplication**
  Determining the most efficient way to multiply a given sequence of matrices.
  - ***Problem Statement:*** Given a sequence of matrices, find the optimal way to parenthesize the matrices so that the number of scalar multiplications is minimized.

```python
def matrix_chain_order(p):
    n = len(p) - 1
    dp = [[0 for _ in range(n)] for _ in range(n)]

    for length in range(2, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            dp[i][j] = float('inf')
            for k in range(i, j):
                q = dp[i][k] + dp[k + 1][j] + p[i] *
p[k + 1] * p[j + 1]
                if q < dp[i][j]:
                    dp[i][j] = q
    return dp[0][n - 1]
```

- **Edit Distance (Levenshtein Distance)**
  Finding the minimum number of operations required to convert one string into another.
  - ***Problem Statement:*** Given two strings, compute the minimum number of edit operations (insertions, deletions, or substitutions) required to transform one string into the other.

```python
def edit_distance(str1, str2):
    m, n = len(str1), len(str2)
    dp = [[0 for _ in range(n + 1)] for _ in range(m + 1)]
     for i in range(m + 1):
        for j in range(n + 1):
            if i == 0:
                dp[i][j] = j
            elif j == 0:
                dp[i][j] = i
            elif str1[i - 1] == str2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            else:
                dp[i][j] = 1 + min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1])

    return dp[m][n]
```

Practice Questions (Basic)

Fibonacci Sequence

1. **LeetCode:** Fibonacci Number

Practice Here

Asked in :

2. **HackerRank:** Fibonacci Numbers

Practice Here

Asked in :

3. **LeetCode:** Climbing Stairs

Practice Here

Asked in :

4. **HackerRank:** The Coin Change Problem

Practice Here

Asked in :

5. **LeetCode:** House Robber

Practice Here

Asked in :

Practice Questions (Basic)

Climbing Stairs

1. **LeetCode:** Climbing Stairs

Practice Here

Asked in :

2. **HackerRank:** Davis' Staircase

Practice Here

Asked in :

3. **LeetCode:** Min Cost Climbing Stairs

Practice Here

Asked in :

4. **LeetCode:** Frog Jump

Practice Here

Asked in :

5. **LeetCode:** Reach a Number

Practice Here

Asked in :

Practice Questions (Basic)

House Robber Problem

1. **LeetCode:** House Robber

Practice Here

Asked in :

2. **LeetCode:** House Robber II

Practice Here

Asked in :

3. **LeetCode:** Delete and Earn

Practice Here

Asked in :

4. **LeetCode:** Paint House

Practice Here

Asked in :

5. **LeetCode:** Paint House II

Practice Here

Asked in :

Practice Questions (Intermediate)

0/1 Knapsack Problem

1. **LeetCode:** Partition Equal Subset Sum

Practice Here Asked in :

2. **LeetCode:** Ones and Zeroes

Practice Here Asked in :

3. **HackerRank:** Knapsack

Practice Here Asked in :

4. **LeetCode:** Last Stone Weight II

Practice Here Asked in :

5. **LeetCode:** Target Sum

Practice Here Asked in :

Practice Questions (Intermediate)

Longest Common Subsequence

1. **LeetCode:** Longest Common Subsequence

Practice Here

Asked in : 

2. **HackerRank:** Common Child

Practice Here

Asked in : 

3. **LeetCode:** Longest Palindromic Subsequence

Practice Here

Asked in : 

4. **LeetCode:** Delete Operation for Two Strings

Practice Here

Asked in : 

5. **LeetCode:** Shortest Common Supersequence

Practice Here

Asked in : 

Practice Questions (Intermediate)

Coin Change Problem

1. **LeetCode:** Coin Change

Practice Here

Asked in : 

2. **LeetCode:** Coin Change 2

Practice Here

Asked in : 

3. **HackerRank:** The Coin Change Problem

Practice Here

Asked in : 

4. **LeetCode:** Minimum Cost to Merge Stones

Practice Here

Asked in : 

5. **LeetCode:** Combination Sum IV

Practice Here

Asked in : 

Practice Questions (Advanced)

Matrix Chain Multiplication

1. **HackerRank:** Matrix Chain Multiplication

Practice Here

Asked in :

2. **LeetCode:** Minimum Difficulty of a Job Schedule

Practice Here

Asked in :

3. **LeetCode:** Burst Balloons

Practice Here

Asked in :

4. **LeetCode:** Stone Game

Practice Here

Asked in :

5. **LeetCode:** Strange Printer

Practice Here

Asked in :

Practice Questions (Advanced)

Edit Distance (Levenshtein Distance)

1. **LeetCode:** Edit Distance

Practice Here

Asked in :

2. **HackerRank:** The Minimum Edit Distance

Practice Here

Asked in :

3. **LeetCode:** Minimum Insertion Steps to Make a String Palindrome

Practice Here

Asked in :

4. **LeetCode:** Delete Operation for Two Strings

Practice Here

Asked in :

5. **LeetCode:** Regular Expression Matching

Practice Here

Asked in :

Practice Questions (Advanced)

Longest Increasing Subsequence

1. **LeetCode:** Longest Increasing Subsequence

Practice Here

Asked in :

2. **HackerRank:** Longest Increasing Subsequence

Practice Here

Asked in :

3. **LeetCode:** Russian Doll Envelopes

Practice Here

Asked in :

4. **LeetCode:** Longest String Chain

Practice Here

Asked in :

5. **LeetCode:** Maximum Length of Pair Chain

Practice Here

Asked in :

EASY LEVEL

Interview Questions (Leetcode)

1. Climbing Stairs

Practice Here

Asked in :

2. House Robber

Practice Here

Asked in :

3. Fibonacci Number

Practice Here

Asked in :

4. Min Cost Climbing Stairs

Practice Here

Asked in :

5. Best Time to Buy and Sell Stock

Practice Here

Asked in :

MEDIUM LEVEL

Interview Questions (Leetcode)

1. Coin Change

Practice Here

Asked in :

2. Longest Increasing Subsequence

Practice Here

Asked in :

3. Longest Common Subsequence

Practice Here

Asked in :

4. Maximum Subarray

Practice Here

Asked in :

5. Target Sum

Practice Here

Asked in :

HARD LEVEL

Interview Questions (Leetcode)

1. Edit Distance

Practice Here

Asked in :

2. Burst Balloons

Practice Here

Asked in :

3. Super Egg Drop

Practice Here

Asked in :

4. Distinct Subsequences

Practice Here

Asked in :

5. Minimum Insertion Steps to Make a String Palindrome

Practice Here

Asked in :

BOSSCODER ACADEMY

WHY BOSSCODER?

1000+ Alumni placed at Top Product-based companies.

More than **136% hike** for every **2 out of 3** working professional.

Average package of **24LPA.**

The syllabus is most **up-to-date** and the list of problems provided covers **all important topics.**

Lavanya
Meta

Course is very well **structured** and **streamlined** to crack any **MAANG** company

Rahul .
Google

EXPLORE MORE