Design and Analysis of Algorithms – Notes

Unit 1: Introduction and Analysis Techniques

1. Algorithm

Detailed Definition: An algorithm is a step-by-step procedure or set of rules to solve a specific problem or perform a computation. It takes inputs, processes them, and produces outputs.

Characteristics of Good Algorithms:

- **Input:** Must have zero or more inputs
- **Output:** Must produce at least one output
- **Definiteness:** Each step must be clear and unambiguous
- **Finiteness:** Must terminate after a finite number of steps
- **Effectiveness:** Each step must be basic enough to be done exactly

Example: Recipe for cooking, directions to reach a location

2. Pseudocode

Detailed Definition: Pseudocode is a simplified, human-readable representation of an algorithm that uses a mixture of natural language and programming language-like structures.

Purpose:

- Helps in planning before actual coding
- Easy to understand and modify
- Language independent

Example:

```
BEGIN
    READ number
    IF number % 2 == 0 THEN
        PRINT "Even"
    ELSE
        PRINT "Odd"
    END IF
END
```

3. Complexity Analysis

Detailed Definition: The process of determining how the resource requirements (time and space) of an algorithm grow as the input size increases.

Types of Complexity:

- **Time Complexity:** How execution time increases with input size
- **Space Complexity:** How memory usage increases with input size

4. Asymptotic Notation

Detailed Definition: Mathematical notations used to describe the limiting behavior of functions, commonly used to analyze algorithm performance.

Types:

- **Big-O (O):** Upper bound (worst-case)
- **Omega (Ω):** Lower bound (best-case)
- **Theta (Θ):** Tight bound (average-case)

Examples:

- O(1): Constant time
- O(log n): Logarithmic time
- O(n): Linear time
- O(n^2): Quadratic time

5. Recursive Algorithms

Detailed Definition: Algorithms that solve problems by breaking them into smaller subproblems of the same type and calling themselves.

Components:

- **Base Case:** Condition to stop recursion
- **Recursive Case:** Part where function calls itself

Example - Factorial:

```
factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

6. Probabilistic Analysis

Detailed Definition: Analyzing algorithms by considering probability distributions over possible inputs.

Video Links:

- **English:** Introduction to Algorithms
- **Hindi:** Algorithm Basics in Hindi

Unit 2: Divide and Conquer

1. Divide and Conquer Methodology

Detailed Definition: A problem-solving paradigm that breaks a problem into smaller subproblems, solves them recursively, and combines their solutions.

Three Steps:

1. **Divide:** Break problem into smaller subproblems

2. **Conquer:** Solve subproblems recursively
3. **Combine:** Merge solutions of subproblems

2. Binary Search

Detailed Definition: An efficient search algorithm that finds the position of a target value within a sorted array by repeatedly dividing the search interval in half.

Process:

- Compare target with middle element
- If matches, return position
- If target is smaller, search left half
- If target is larger, search right half

Time Complexity: O(log n)

3. Merge Sort

Detailed Definition: A sorting algorithm that divides the array into halves, sorts each half, and then merges them back together.

Steps:

1. Divide array into two halves
2. Recursively sort each half
3. Merge two sorted halves

Time Complexity: O(n log n)

4. Quick Sort

Detailed Definition: A sorting algorithm that picks a 'pivot' element and partitions the array around the pivot, then recursively sorts the subarrays.

Steps:

1. Choose pivot element
2. Partition array around pivot
3. Recursively sort left and right partitions

Time Complexity: O(n log n) average, $O(n^2)$ worst-case

5. Selection Algorithms

Detailed Definition: Algorithms to find the kth smallest/largest element in an array.

Examples:

- Quickselect algorithm
- Median of medians algorithm

6. Recurrence Relations

Detailed Definition: Equations that describe functions in terms of their values on smaller inputs, used to analyze recursive algorithms.

Video Links:

- **English:** Divide and Conquer Algorithms
- **Hindi:** Divide and Conquer in Hindi

Unit 3: Greedy Techniques

1. Greedy Strategy

Detailed Definition: An algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit.

Characteristics:

- Makes locally optimal choice at each step
- Hope that local optimal leads to global optimal
- Never reconsider choices

2. Job Sequencing

Detailed Definition: Scheduling jobs with deadlines to maximize profit, where each job takes unit time and has a deadline and profit.

Approach:

- Sort jobs by decreasing profit
- Schedule each job as late as possible without missing deadline

3. Fractional Knapsack

Detailed Definition: A variation of knapsack problem where items can be broken into fractions, and the goal is to maximize total value without exceeding capacity.

Approach:

- Calculate value-to-weight ratio for each item
- Sort items by decreasing ratio
- Take items in order until knapsack is full

4. Minimum Spanning Tree (MST)

Detailed Definition: A spanning tree of a connected, undirected graph that has minimum possible total edge weight.

Algorithms:

- **Prim's Algorithm:** Grows tree from starting vertex
- **Kruskal's Algorithm:** Adds edges in increasing weight order

5. Dijkstra's Algorithm

Detailed Definition: Finds the shortest path from a source vertex to all other vertices in a graph with non-negative edge weights.

Process:

1. Initialize distances: 0 for source, ∞ for others
2. Select vertex with minimum distance

3. Update distances of neighbors
4. Repeat until all vertices processed

6. Prim's Algorithm

Detailed Definition: Finds minimum spanning tree by growing tree from starting vertex, always adding cheapest connection.

Video Links:

- **English:** Greedy Algorithms
- **Hindi:** Greedy Algorithms in Hindi

Unit 4: Dynamic Programming

1. Dynamic Programming Principles

Detailed Definition: A method for solving complex problems by breaking them down into simpler subproblems and storing their solutions to avoid redundant computations.

Key Features:

- **Optimal Substructure:** Optimal solution contains optimal sub-solutions
- **Overlapping Subproblems:** Same subproblems solved multiple times

2. Matrix Chain Multiplication

Detailed Definition: Finding the most efficient way to multiply a sequence of matrices to minimize the number of scalar multiplications.

Approach:

- Break problem into subproblems of multiplying smaller chains
- Use table to store minimum costs
- Build solution bottom-up

3. Longest Common Subsequence (LCS)

Detailed Definition: Finding the longest subsequence common to two sequences (subsequence elements appear in same order but not necessarily contiguous).

Application:

- DNA sequence comparison
- File difference detection

4. Optimal Binary Search Trees

Detailed Definition: Constructing a binary search tree with minimum expected search cost given access frequencies for keys.

Video Links:

- **English:** Dynamic Programming
- **Hindi:** Dynamic Programming in Hindi

Unit 5: Graph Algorithms

1. Graph Traversals

Detailed Definition: Systematic methods for visiting all vertices and edges of a graph.

Depth-First Search (DFS)

- Explore as far as possible along each branch before backtracking
- Uses stack (implicit or explicit)
- Applications: Cycle detection, topological sort

Breadth-First Search (BFS)

- Explore all neighbors at current depth before moving deeper
- Uses queue
- Applications: Shortest path in unweighted graphs

2. Topological Sorting

Detailed Definition: Linear ordering of vertices in a directed acyclic graph (DAG) such that for every directed edge u→v, u comes before v.

Applications: Task scheduling, dependency resolution

3. Strongly Connected Components

Detailed Definition: Maximal sets of vertices in a directed graph where every vertex is reachable from every other vertex.

Algorithm: Kosaraju's algorithm

4. Shortest Path Algorithms

- **Dijkstra's:** Non-negative weights
- **Bellman-Ford:** Handles negative weights
- **Floyd-Warshall:** All pairs shortest paths

5. Spanning Trees

- **Minimum Spanning Tree:** Tree connecting all vertices with minimum total weight
- **Algorithms:** Prim's, Kruskal's

Video Links:

- **English:** Graph Algorithms
- **Hindi:** Graph Algorithms in Hindi

Unit 6: Backtracking and Branch and Bound

1. Backtracking

Detailed Definition: A systematic way to iterate through all possible configurations to solve computational problems, abandoning partial candidates when they cannot lead to valid solutions.

Process:

1. Build solution incrementally
2. Abandon path when it cannot lead to solution
3. Backtrack to try alternatives

2. N-Queens Problem

Detailed Definition: Placing N chess queens on an N×N chessboard so that no two queens threaten each other.

Approach:

- Place queens column by column
- Check for conflicts with previously placed queens
- Backtrack when no valid position found

3. Hamiltonian Cycle

Detailed Definition: Finding a cycle in a graph that visits each vertex exactly once and returns to starting vertex.

4. Sum of Subsets

Detailed Definition: Finding all subsets of a set of positive integers that sum to a given target value.

5. Branch and Bound

Detailed Definition: An algorithm design paradigm for discrete and combinatorial optimization problems that uses bounding functions to avoid generating some subproblems.

Difference from Backtracking:

- Uses bounding function to prune branches
- More efficient for optimization problems

Video Links:

- **English:** Backtracking Algorithms
- **Hindi:** Backtracking in Hindi

Unit 7: Complexity Theory and Advanced Topics

1. Complexity Classes

P (Polynomial Time)

Detailed Definition: Problems that can be solved in polynomial time by a deterministic Turing machine.

Examples: Sorting, searching, shortest path

NP (Nondeterministic Polynomial Time)

Detailed Definition: Problems whose solutions can be verified in polynomial time.

Examples: Boolean satisfiability, Hamiltonian path

NP-Complete

Detailed Definition: The hardest problems in NP; if any NP-complete problem has a polynomial-time solution, then all NP problems do.

Examples: Traveling salesman, graph coloring

2. P vs NP Problem

Detailed Definition: The unsolved question of whether every problem whose solution can be quickly verified can also be quickly solved.

3. Approximation Algorithms

Detailed Definition: Algorithms that find near-optimal solutions for NP-hard problems in polynomial time.

Types:

- Constant factor approximation
- Polynomial-time approximation schemes

4. Randomized Algorithms

Detailed Definition: Algorithms that make random choices during execution.

Types:

- **Las Vegas:** Always correct, running time random
- **Monte Carlo:** May be incorrect, fixed running time

5. Amortized Analysis

Detailed Definition: Analyzing the average time per operation over a worst-case sequence of operations.

6. String Algorithms

- **Pattern Matching:** KMP algorithm, Rabin-Karp
- **String Compression:** Huffman coding
- **Edit Distance:** Minimum operations to transform one string to another

Video Links:

- **English:** Complexity Theory
- **Hindi:** P vs NP in Hindi

Important Algorithm Analysis Techniques

1. Time Complexity Analysis Methods

- **Counting Operations:** Count basic operations
- **Recurrence Relations:** For recursive algorithms
- **Master Theorem:** For divide-and-conquer recurrences
- **Amortized Analysis:** For data structure operations

2. Space Complexity Analysis

- **Auxiliary Space:** Extra space used by algorithm

- **Input Space:** Space for storing input
- **Total Space:** Sum of auxiliary and input space

3. Algorithm Design Strategies

1. **Brute Force:** Try all possibilities
2. **Divide and Conquer:** Break into subproblems
3. **Greedy:** Locally optimal choices
4. **Dynamic Programming:** Store subproblem solutions
5. **Backtracking:** Systematic trial and error
6. **Branch and Bound:** Prune search space

4. Real-World Applications

- **Search Engines:** String matching, ranking
- **GPS Navigation:** Shortest path algorithms
- **Social Networks:** Graph algorithms
- **E-commerce:** Recommendation systems
- **Bioinformatics:** Sequence alignment

Practice Resources

- **Online Judges:** LeetCode, HackerRank, CodeForces
- **Books:** "Introduction to Algorithms" (CLRS)
- **Visualization:** VisuAlgo.net

Final English Video: Algorithms Full Course
Final Hindi Video: Algorithms Complete Course