Algorithm_python Repository – Time and Space Complexity Analysis

This document provides the time and space complexity analysis of all algorithms implemented in the repository **Algorithm_python**. The repository contains Python scripts and Jupyter Notebooks from the Advanced Algorithms course, covering Dynamic Programming, Greedy, and Randomized Algorithms, as well as basic searching, sorting, and mathematical routines.

Searching & Sorting

- **Binary Search (Rightmost Occurrence)**
  Time Complexity: O(log n)
  Space Complexity: O(1)
- **Selection Sort**
  Time Complexity: O(n^2)
  Space Complexity: O(1)

Mathematical Algorithm

- **Greatest Common Divisor (Euclidean Algorithm)**
  Time Complexity: O(log(min(a,b)))
  Space Complexity: O(1)

Dynamic Programming

- **Longest Common Subsequence (LCS)**
  Time Complexity: O(n*m)
  Space Complexity: O(n*m)
- **Matrix Chain Multiplication**
  Time Complexity: O(n^3)
  Space Complexity: O(n^2)
- **0/1 Knapsack**
  Time Complexity: O(n*W)
  Space Complexity: O(n*W)
- **Unbounded Knapsack**
  Time Complexity: O(n*W)
  Space Complexity: O(W)
- **Coin Change Problem**
  Time Complexity: O(n*amount)
  Space Complexity: O(amount)

Greedy Algorithms

- **Huffman Coding**
  Time Complexity: O(n log n)
  Space Complexity: O(n)
- **Activity Selection**
  Time Complexity: O(n log n)
  Space Complexity: O(1)

- **Fractional Knapsack**
  Time Complexity: O(n log n)
  Space Complexity: O(1)
- **Minimum Spanning Tree (Prim’s / Kruskal’s)**
  Time Complexity: O(E log V)
  Space Complexity: O(V+E)

Randomized Algorithms

- **Randomized QuickSort**
  Time Complexity: Average: O(n log n), Worst: O(n^2)
  Space Complexity: O(log n)
- **Randomized Selection**
  Time Complexity: Average: O(n), Worst: O(n^2)
  Space Complexity: O(1)