Greedy Algorithms Cheatsheet

Topic Overview

Greedy Algorithms in Java make locally optimal choices, used for optimization problems. This cheatsheet covers greedy techniques.

Prerequisites

Arrays

List of Subtopics

- Activity Selection
- Fractional Knapsack
- Huffman Coding
- Minimum Spanning Tree (MST)
- Dijkstra's Algorithm
- Coin Change (Greedy)
- Job Sequencing
- Minimum Coin Change
- Egyptian Fraction
- Huffman Decoding

Key Concepts Explained

- **Activity Selection**: Selects non-overlapping activities with maximum count.
- **Fractional Knapsack**: Allows partial items for maximum value.
- **MST**: Finds minimum edge weight tree connecting all nodes.

Approaches to Solve Problems with Step-by-Step Algorithms

- **Activity Selection**:
  - **Algorithm**:
    1. Sort activities by end time.
    2. Select first activity, include next non-overlapping.
    3. Repeat until no activities remain.
  - **Context**: O(n log n) time, O(1) space.
- **Fractional Knapsack**:
  - **Algorithm**:
    1. Sort items by value/weight ratio.
    2. Take items greedily, use fractions if needed.
  - **Context**: O(n log n) time, O(1) space.
- **Huffman Coding**:
  - **Algorithm**:
    1. Create min heap of characters and frequencies.
    2. Repeatedly merge two smallest, build tree.
  - **Context**: O(n log n) time, O(n) space.
- **Minimum Spanning Tree (MST)**:
  - **Algorithm**:
    1. Use Kruskal's or Prim's with greedy edge selection.
  - **Context**: O(E log E) or O((V+E) log V) time.
- **Dijkstra's Algorithm**:
  - **Algorithm**:
    1. Use min heap, initialize distances to infinity.
    2. Update distances greedily from source.
  - **Context**: O((V+E) log V) time.
- **Coin Change (Greedy)**:
  - **Algorithm**:
    1. Sort coins descending, use largest possible.
    2. Repeat until amount is zero.
  - **Context**: O(n) time, works if optimal.
- **Job Sequencing**:

  - **Algorithm**:
    1. Sort jobs by profit, schedule with max deadline.
  - **Context**: O(n log n) time.

- **Minimum Coin Change**:
  - **Algorithm**:
    1. Sort coins, use greedy with minimum count.
  - **Context**: O(n) time, not always optimal.

- **Egyptian Fraction**:
  - **Algorithm**:
    1. Convert fraction to sum of unit fractions greedily.
  - **Context**: O(log n) time.

- **Huffman Decoding**:
  - **Algorithm**:
    1. Use Huffman tree, traverse based on bits.
  - **Context**: O(n) time.

Common LeetCode Problems with Approaches

- **Jump Game (55)**: Use greedy to check reachable index.
- **Minimum Number of Arrows to Burst Balloons (452)**: Sort and merge intervals.
- **Coin Change 2 (518)**: Use greedy where applicable.

Time & Space Complexities

- Varies: O(n log n) to O((V+E) log V)
- Space: O(1) to O(V)

Important Tips & Tricks

- Verify greedy choice property before applying.
- Sort data to enable greedy decisions.
- Use heaps for efficient minimum/maximum selection.
- Handle edge cases like zero values.
- Test with counterexamples to ensure optimality.