Cheat sheet time complexity of all algo till midterm

Here's a cheatsheet covering Bubble Sort, Selection Sort, Insertion Sort, Divide-and-Conquer algorithms, Greedy algorithms, and Dynamic Programming.

Sorting Algorithms

1. Bubble Sort

- **Description**: Repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order.
- **Time Complexity**:
  - Best Case: (O(n))
  - Average Case: (O(n^2))
  - Worst Case: (O(n^2))

2. Selection Sort

- **Description**: Repeatedly finds the minimum element from the unsorted part and puts it at the beginning.
- **Time Complexity**:
  - Best Case: (O(n^2))
  - Average Case: (O(n^2))
  - Worst Case: (O(n^2))

3. Insertion Sort

- **Description**: Builds the final sorted array one item at a time, with the assumption that the first element is already sorted.
- **Time Complexity**:
  - Best Case: (O(n))
  - Average Case: (O(n^2))
  - Worst Case: (O(n^2))

Divide-and-Conquer Algorithms

Structure

1. **Divide**: Break the problem into smaller subproblems of the same type.
2. **Conquer**: Solve the subproblems recursively.
3. **Combine**: Combine the solutions of the subproblems to get the solution of the original problem.

Examples

1. **Binary Search**
   - **Time Complexity**: (O(log n))
2. **Quick Sort**
   - **Time Complexity**:
     - Average Case: (O(n log n))
     - Worst Case: (O(n^2))
3. **Merge Sort**
   - **Time Complexity**: (O(n log n))
4. **Strassen Multiplication**
   - **Time Complexity**: (O(n^{2.81}))
5. **Max-Min Problem**
   - **Time Complexity**: (O(n))

Greedy Algorithms

Introduction

- Greedy algorithms build up a solution piece by piece, always choosing the next piece that offers the most immediate benefit.

Elements of Greedy Strategy

1. **Greedy Choice Property**: A global optimum can be arrived at by selecting a local optimum.
2. **Optimal Substructure**: An optimal solution to the problem contains an optimal solution to subproblems.

Examples

1. **Minimum Spanning Tree**
   - **Kruskal's Algorithm**: (O(E log E))
   - **Prim's Algorithm**: (O(E + V log V))
2. **Dijkstra’s Algorithm**
   - **Time Complexity**: (O(V^2)) or (O(E + V log V))
3. **Knapsack Problem (Fractional)**
   - **Time Complexity**: (O(n log n))
4. **Activity Selection Problem**
   - **Time Complexity**: (O(n log n))
5. **Huffman Codes**
   - **Time Complexity**: (O(n log n))

Dynamic Programming

Principle of Optimality

- An optimal solution to the problem contains an optimal solution to subproblems. This is used to avoid redundant computations by storing the results of subproblems.

Examples

1. **0/1 Knapsack Problem**
   - **Description**: 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.
   - **Time Complexity**: (O(nW))