Greedy Algorithms

Definition:

A greedy algorithm builds up a solution piece by piece, always choosing the next piece that offers the most immediate benefit (local optimum), hoping this leads to a global optimum.

Key Points:

- Makes the best local choice at each step
- Does not reconsider choices (no backtracking)
- Works well when the problem has **greedy choice property** and **optimal substructure**
- Not always guaranteed to find the global optimum but is efficient and simple

---

Common Uses:

- Activity selection problem
- Huffman coding (data compression)
- Minimum Spanning Tree: Kruskal’s and Prim’s algorithms
- Shortest path: Dijkstra’s algorithm

- Fractional Knapsack problem

Characteristics:

- **Greedy choice property:** A globally optimal solution can be arrived at by choosing a local optimum.
- **Optimal substructure:** Optimal solution of the problem contains optimal solutions to subproblems.

Example – Fractional Knapsack Problem:

- Given items with weights and values
- Calculate value/weight ratio for each item
- Pick items starting from the highest ratio until the knapsack is full (fractional parts allowed)

Pseudo Code for Fractional Knapsack:

bash

CopyEdit

```
sort items by value/weight ratio descending
for item in items:
    if item.weight <= capacity:
```

```
        take full item
        capacity -= item.weight
    else:
        take fractional part of item
        break
```

Advantages:

- Simple and fast
- Usually efficient in practice
- Easy to implement

Disadvantages:

- Not guaranteed to work for all problems
- Sometimes dynamic programming or backtracking is required for optimal solutions
-