To understand the concept easily:

What is a Greedy Algorithm?

A Greedy Algorithm is like making the best and easiest choice at every step without worrying about the future.

You don’t think too far ahead—you just grab the best-looking candy right now and hope it works out!

Why do we need it?

Because sometimes we want a quick and simple way to solve a problem, and the greedy way gets us a good (and often best) answer without doing a lot of thinking or trying every possible way.

It saves time and is easy to use for many problems.

How does it work?

Imagine you're picking candies from a box. You want the most candies with the least money. A Greedy Algorithm will:

Look at all the candies.

Pick the one that gives you the most candies for the lowest price.

Repeat until you can't choose anymore.

You keep choosing the best option available at each moment!

Where is it used in real life?

Coin change: Finding the smallest number of coins to pay someone.

Candy packing: Filling a jar with the most valuable candies without overflowing.

Task scheduling: Choosing the shortest homework first to finish faster.

Route finding: Always turning onto the shortest road you see first.

Who created this algorithm?

There’s no one single inventor, but David Huffman is famous for using greedy methods in Huffman coding (used in file compression like ZIP files).

The greedy idea has been around a long time—it's like how humans naturally try to make good choices quickly!

What is the main purpose of this algorithm?

To find a quick and often best solution by always choosing what seems best at that moment. It doesn’t look back or try every possible path—it just keeps grabbing the best choice now.

Real-Life Example: The Candy Jar You’re at a candy store. You have ₹10 and want to buy the most candies.

Candies cost:

Lollipop = ₹5

Chocolate = ₹2

Gummy = ₹1

You use the greedy method:

First, buy the candy that gives you most for less—the ₹1 gummies.

You buy 10 gummies!

You're super happy because you have the most candies!

That’s greedy thinking—you choose the best deal right now each time.

Viva and concept clarity:

**1. What is a Greedy Algorithm?**

**Answer:**  
A Greedy Algorithm is an approach that builds a solution step-by-step by always choosing the option that looks best at the moment (locally optimal choice), with the hope that this leads to the overall best solution (globally optimal).

**Example:**  
In the **Coin Change** problem, if you need ₹27 and have coins of ₹10, ₹5, and ₹1, greedy picks the largest coin each time:

* ₹10 → ₹17 left
* ₹10 → ₹7 left
* ₹5 → ₹2 left
* ₹1 × 2 → ₹0  
  ✅ Total = 5 coins

**✅ 2. When does Greedy Algorithm work best?**

**Answer:**  
Greedy works well when:

* The problem has **optimal substructure** (sub-problems can be solved independently).
* It satisfies the **greedy-choice property** (a locally optimal choice leads to a globally optimal result).

**Example:**  
In **Fractional Knapsack**, choosing the item with highest **profit/weight** ratio at each step leads to the best solution.

**✅ 3. Give an example where Greedy fails.**

**Answer:**  
If the coin denominations are: ₹9, ₹6, ₹1 and the amount is ₹12:  
Greedy picks:

* ₹9 → ₹3 left
* ₹1 × 3 → total 4 coins  
  But optimal is:
* ₹6 + ₹6 → total **2 coins**  
  So, greedy doesn’t always give the best solution.

**✅ 4. Difference between Greedy and Dynamic Programming?**

| **Feature** | **Greedy** | **Dynamic Programming** |
| --- | --- | --- |
| Decision making | Locally optimal | All subproblems considered |
| Memory usage | Low | High (stores results) |
| Time complexity | Usually faster | Usually slower |
| Example problem | Activity selection | 0/1 Knapsack |

**✅ 5. Explain Greedy Algorithm with a real-life example.**

**Answer:**  
Imagine you’re choosing snacks at a fair. You have ₹100 and each stall has different prices. You try to buy the most expensive snack you can afford at each step.  
You’re not looking ahead — just picking what’s best now.  
That’s what a greedy algorithm does: make the best choice at each step, hoping it leads to a great overall result.

**✅ 6. Common Greedy Algorithm problems?**

**Answer:**

* Coin Change
* Activity Selection
* Fractional Knapsack
* Huffman Coding
* Job Scheduling
* Prim’s and Kruskal’s for Minimum Spanning Tree

**✅ 7. What is the Greedy-Choice Property?**

**Answer:**  
It means that **locally optimal choices** (the best at each step) will lead to a **globally optimal solution**.

**Example:**  
In the **Activity Selection** problem, picking the activity that ends earliest ensures we can fit more activities.