Finding Maximum Number of Groups With Increasing Length

Name: D.Madhavan

Regno:192225070

Subject: CSA0656-Design Analysis and Algorithms for Asymptotic Notations.

Guided By: Dr. R. Dhanalakshmi

Professor Department Of Machine Learning

Saveetha School Of Engineering.

Introduction to Greedy Technique

1

Systematic Approach Approach

The Greedy Technique follows a systematic approach, making locally optimal choices at each stage with the aim of finding a global optimum.

2

Immediacy

It focuses on making the best best decision at the current current moment, without considering the long-term term consequences.

3

Simple Implementation

The Greedy Technique is relatively straightforward to implement, implement, making it a popular choice for solving complex problems. problems.

Problem Statement:

Maximum Number of Groups With Increasing Length You are given a 0-indexed array usage Limits of length n. Your task is to create groups using numbers from 0 to n - 1, ensuring that each number, i, is used no more than usage Limits[i] times in total across all groups. You must also satisfy the following conditions: Each group must consist of distinct numbers, meaning that no duplicate numbers are allowed within a single group. Each group (except the first one) must have a length strictly greater than the previous group. Return an integer denoting the maximum number of groups you can create while satisfying these conditions. Example 1: Input: usage Limits = [1,2,5]

Output: 3 Explanation: In this example, we can use 0 at most once, 1 at most at most twice, and 2 at most five times. One way of creating the maximum maximum number of groups while satisfying the conditions is: Group 1 contains 1 contains the number [2]. Group 2 contains the numbers [1,2]. Group 3 3 contains the numbers [0,1,2]. It can be shown that the maximum number of number of groups is 3. So, the output is 3..

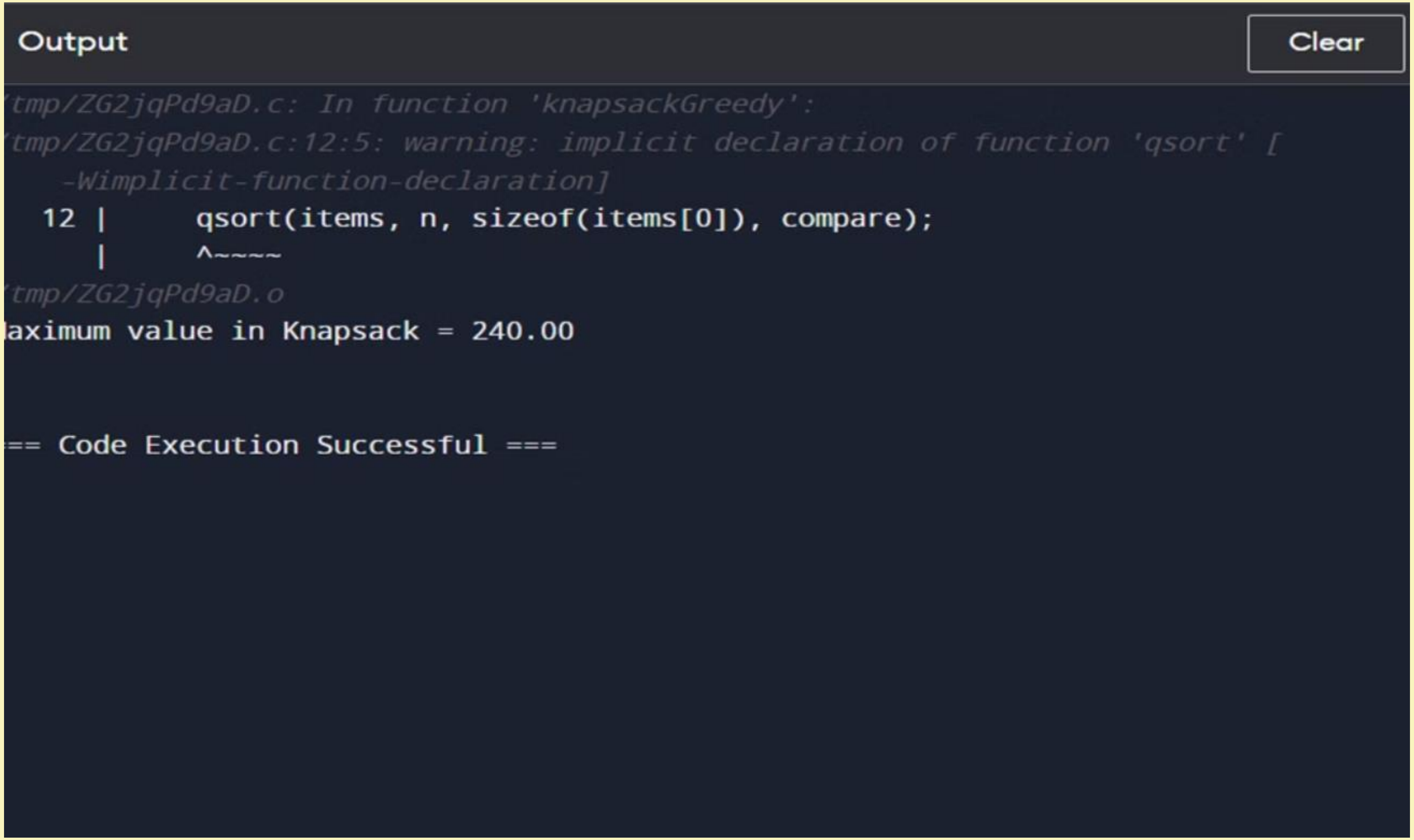
Solution:

For usageLimits = [1, 2, 5]: 1. Total Usage Capacity: o total_usage=1+2+5=8\text{total_usage} = 1 + 2 + 5 = 8total_usage=1+2+5=8. 2. Compute Group Sizes: o For k = 1, S1=1S_1 = 1S1=1. o For k = 2, S2=3S_2 = 3S2=3. o For k = 3, S3=6S_3 = 6S3=6. o For k = 4, S4=10S_4 = 10S4=10. The total usage capacity of 8 is enough for groups of size up to 3 (since S4=10S_4 = 10S4=10 exceeds 8). 3. Verify Group Formation: o Verify if you can form 3 groups with sizes 1, 2, and 3, respecting the usage limits. Here, the groups can be formed as follows: Group 1: 1 element. Group 2: 2 elements. Group 3: 3 elements. This confirms that 3 groups can be formed.

Code:

```
#include <stdio.h>
struct Item {
int weight;
int value;
};
int compare(const void a, const void b) {
double ratio1 = (double)(((struct Item*)a)->value) / (((struct Item*)a)->weight);
double ratio2 = (double)(((struct Item*)b)->value) / (((struct Item*)b)->weight);
return ratio2 > ratio1;
}
double knapsackGreedy(int capacity, struct Item items[], int n) {
qsort(items, n, sizeof(items[0]), compare);
int currentWeight = 0;
double finalValue = 0.0;
for (int i = 0; i < n; i++) {
if (currentWeight + items[i].weight <= capacity) {
```

Output:



```
/tmp/ZG2jqPd9aD.c: In function 'knapsackGreedy':
/tmp/ZG2jqPd9aD.c:12:5: warning: implicit declaration of function 'qsort' [
   -Wimplicit-function-declaration]
  12 |     qsort(items, n, sizeof(items[0]), compare);
     |     ^~~~~
/tmp/ZG2jqPd9aD.o
Maximum value in Knapsack = 240.00


=== Code Execution Successful ===
```

Advantages of Greedy Technique

Simplicity

The Greedy Technique is easy to understand and implement, making it a popular choice for problem-solving.

Speed

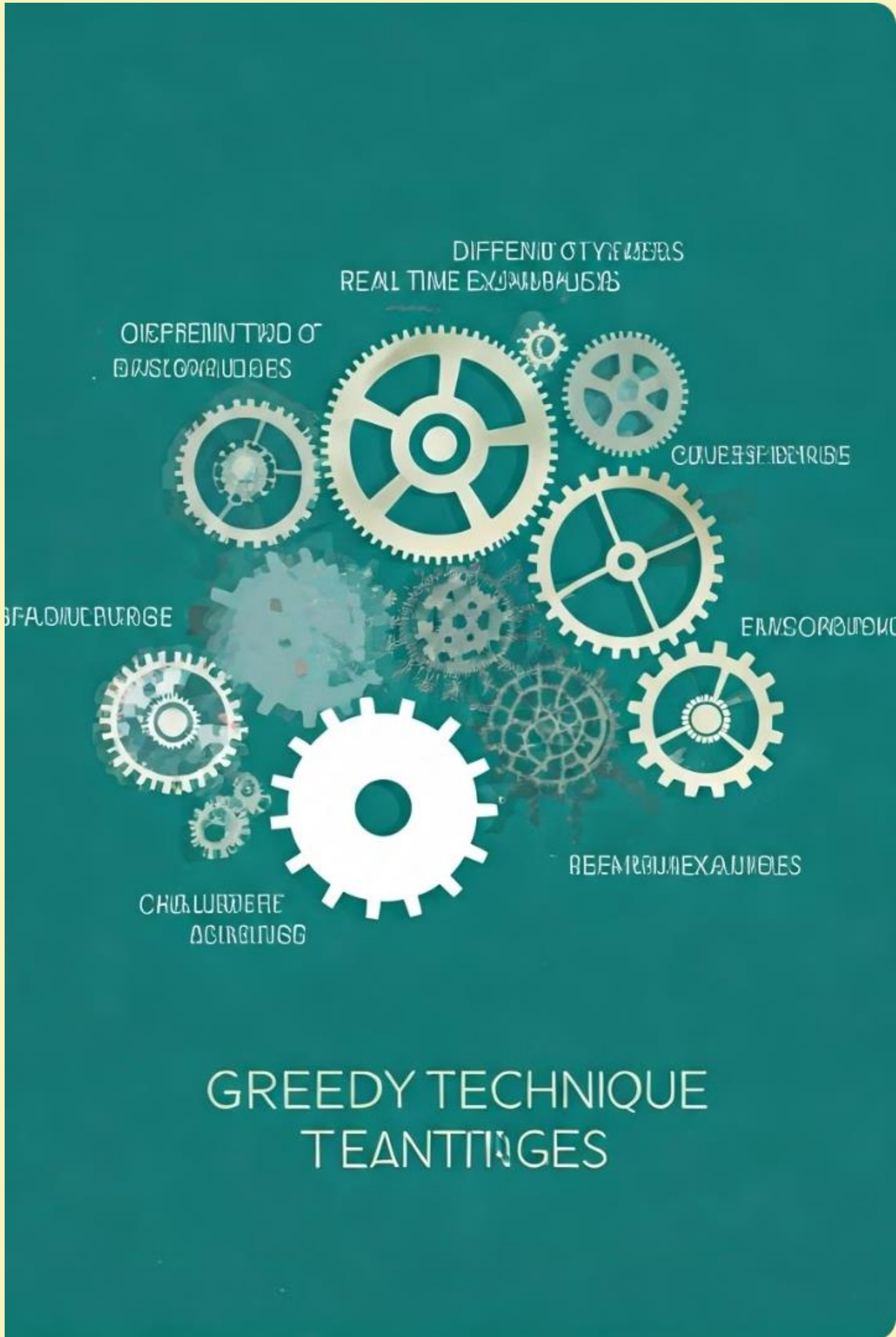
It can quickly find a solution, as it does not require extensive analysis or backtracking.

Memory Efficiency

The Greedy Technique does not require storing a large amount of data, making it memory-efficient.

Applicability

It can be applied to a wide range range of problems, from scheduling to optimization tasks. tasks.



Disadvantages of Greedy Technique

No Guarantee of Optimality

The Greedy Technique does not always always find the globally optimal solution, solution, as it focuses on local optimality. optimality.

Lack of Flexibility

It cannot backtrack or reconsider previous decisions, limiting its ability to ability to adapt to changing situations. situations.

Potential for Suboptimal Solutions

In some cases, the Greedy Technique Technique may lead to suboptimal solutions that are not the best overall. overall.

Applications of Greedy Technique

Scheduling

Allocating resources and tasks tasks based on immediate needs.

Network Optimization Optimization

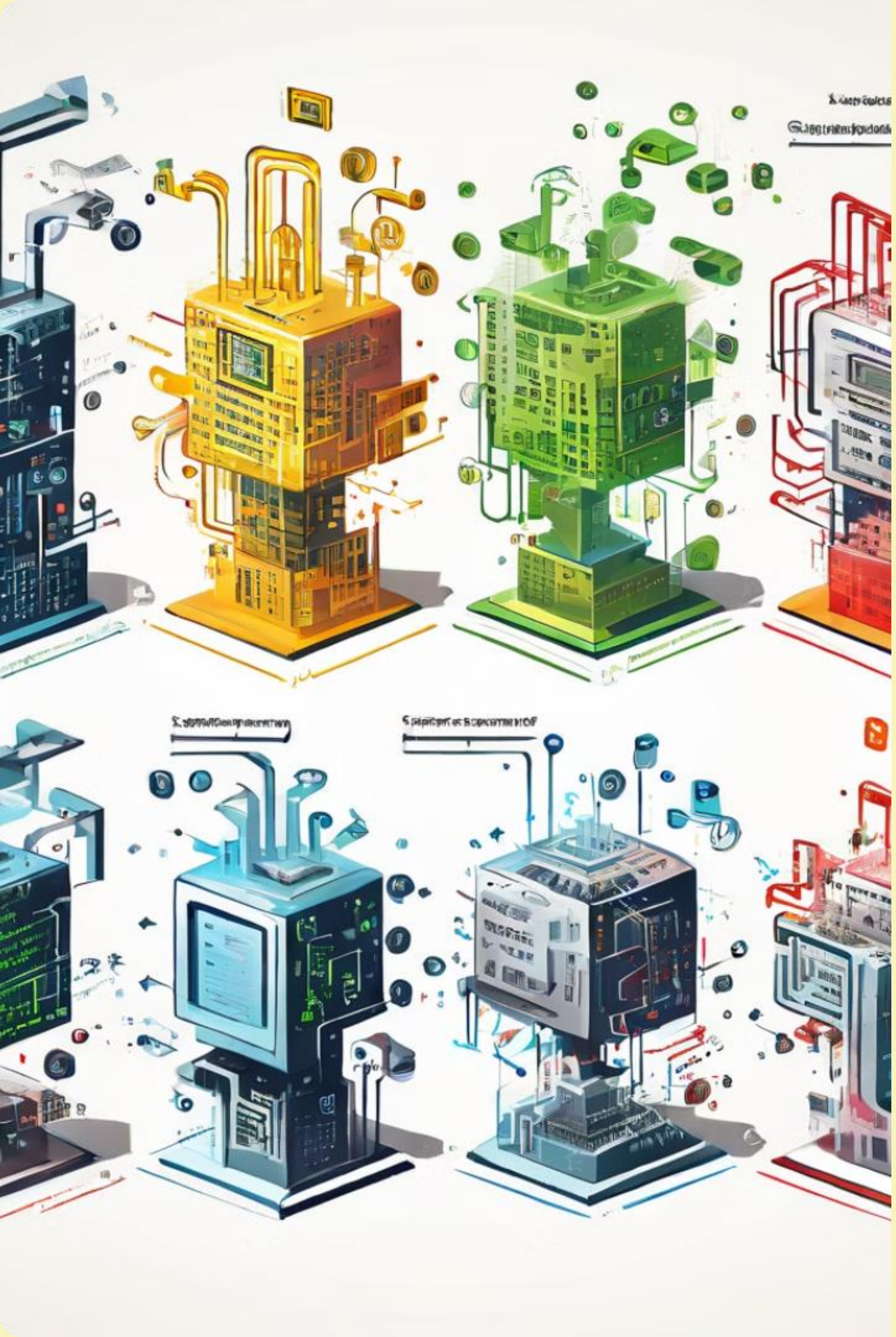
Improving network efficiency efficiency by prioritizing the the most critical connections. connections.

Resource Allocation

Distributing limited resources resources to maximize immediate benefits.

Graph Algorithms

Traversing and exploring graphs using the Greedy Technique.



Greedy Technique Algorithms

1

Kruskal's Algorithm

Finds the minimum spanning tree of a weighted graph by adding the cheapest available edge at each step.

2

Dijkstra's Algorithm

Computes the shortest path between nodes in a graph by by repeatedly choosing the node with the smallest distance. distance.

3

Huffman Coding

Constructs an optimal prefix code by repeatedly combining the combining the two least frequent symbols into a new node. node.

Implementing Greedy Technique in Projects

1

Problem Analysis

Identify the problem and its characteristics to determine if the Greedy Technique is an Technique is an appropriate approach.

2

Algorithm Selection

Choose the right Greedy Technique algorithm based on the problem requirements and constraints.

3

Implementation

Carefully implement the selected algorithm, ensuring it meets the project's objectives. objectives.

4

Testing and Optimization

Thoroughly test the implementation and optimize it for efficiency, if necessary.

Conclusion and Key Takeaways

1

Powerful Approach

The Greedy Technique is a powerful problem-solving approach that can be applied to a wide range of problems.

2

Balancing Advantages Advantages and Disadvantages

Understanding the strengths strengths and limitations of of the Greedy Technique is is crucial for effective implementation.

3

Continuous Learning

Staying up-to-date with the latest Greedy Technique algorithms and algorithms and their applications is essential for staying competitive. competitive.