KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai,Erode – 638060

DEPARTMENT OF INFORMATION TECHNOLOGY

FAKE COIN DETECTION USING DIVIDE AND CONQUER ALGORITHM

A MICRO PROJECT REPORT

FOR

DESIGN AND ANALYSIS OF ALGORITHMS (22ITT31)

SUBMITTED BY

SHREYAA J (23ITR151)

KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai,Erode – 638060

DEPARTMENT OF INFORMATION TECHNOLOGY

FAKE COIN DETECTION USING DIVIDE AND CONQUER ALGORITHM

A MICRO PROJECT REPORT

FOR

DESIGN AND ANALYSIS OF ALGORITHMS(22ITT31)

SUBMITTED BY

SHREYAA J (23ITR151)

KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai,Erode – 638060

DEPARTMENT OF INFORMATION TECHNOLOGY

<u>BONAFIDE CERTIFICATE</u>

Name	: SHREYAA J
Course Code	: 22ITT31
Course Name	: DESIGN AND ANALYSIS OF ALGORITHMS
Semester	: IV

Certified that this is a bonafide record of work for application project done by the above student for 22ITT31-DESIGN AND ANALYSIS OF ALGORITHMS during the academic year 2024-2025.

Submitted for the Viva Voice Examination held on ______________________

Faculty Incharge Head of the Department

ABSTRACT

The Fake Coin Detection problem is a classic example used to illustrate the efficiency of divide and conquer algorithms. In this project, I implement a digital simulation of identifying a single fake coin—lighter or heavier—among a set of identical-looking coins using the divide-and-conquer strategy. The algorithm recursively divides the coin set into three groups, compares their weights using a balance scale simulation, and narrows down the suspect group in logarithmic time. This approach significantly reduces the number of weighings required compared to linear search methods. The project demonstrates algorithmic thinking, optimal problem-solving, and provides interactive visualization to enhance user understanding. Potential applications include fault detection, process optimization, and algorithm learning tools.

TABLE OF CONTENTS

1.0 INTRODUCTION

Identifying a counterfeit item among a group of seemingly identical items has long been a popular problem in the study of algorithms. The *Fake Coin Detection* problem is a classic example that challenges one to find a single fake coin that differs in weight (lighter) using the minimum number of weighings

In this project, we simulate the fake coin detection process using a divide-and-conquer strategy, where a set of coins is recursively divided into three groups. A balance scale is then used to compare the weights of two of the groups, helping us eliminate two-thirds of the coins in each step and quickly narrow down the fake coin. This method is far more efficient than a brute-force approach and illustrates the power of algorithmic optimization.

The project includes an interactive web-based visualization that allows users to input the number of coins, observe the grouping and weighing process, and follow the algorithm as it narrows down to the fake coin. The goal is to provide an educational and engaging tool that demonstrates how divide and conquer works in practice.

1.1 PURPOSE

The main purpose of this project is:

- Make complex algorithmic concepts more accessible and understandable, especially for students and learners.
- Highlight the efficiency and elegance of divide and conquer techniques in reducing problem-solving time and complexity.
- Encourage logical thinking, problem decomposition, and strategic decision-making through an interactive platform.
- Serve as a learning aid for computer science education, particularly in the areas of algorithms and problem-solving strategies.

1.2 OBJECTIVE

The main objective of this project is to design and implement an efficient system to detect a single fake coin from a group of visually identical coins using the **Divide and Conquer** algorithm. The specific goals include:

- To simulate the classic fake coin problem through a step-by-step digital visualization.
- To apply the divide and conquer strategy to reduce the number of comparisons and identify the fake coin with minimal weighings.
- To enhance understanding of algorithmic problem-solving through interactive learning.
- To demonstrate the practical application of theoretical computer science concepts in a visually intuitive way.

1.3 METHODOLOGY OVERVIEW

The project follows a structured approach to detect the fake coin using the Divide and Conquer methodology. The steps involved in the implementation are:

1. **UserInput:**

   The user enters the total number of coins to be analyzed. One coin is randomly (or manually) designated as fake, being either lighter or heavier.

2. **CoinGrouping:**

   The set of coins is divided into three nearly equal groups. This trichotomy is essential to apply the divide and conquer logic effectively.

3. **WeighingProcess:**

   Two of the groups are weighed against each other using a simulated balance scale. Based on the result:

   - If the weights are equal, the fake coin is in the third group.
   - If one group is heavier or lighter, the fake coin lies in that group depending on whether it's supposed to be heavier or lighter.

4. **RecursiveDivision:**

   The identified group containing the fake coin is further divided into three subgroups, and the weighing process is repeated.

5. **BaseCase:**

   This recursive process continues until only one coin remains—identified as the fake coin.

6. **Visualization:**

   Each step is visually presented to the user, showing the groups, the weighing result, and the narrowing process, making the logic intuitive and educational.

2. PROBLEM STATEMENT

The Fake Coin Detection Problem is a classic algorithmic challenge in which a single counterfeit coin must be identified from a group of n visually identical coins. The fake coin differs in weight—lighter than the genuine coins—but its exact deviation is unknown. The only allowed operation is comparing the weights of two groups of coins using a balance scale. The goal is to identify the fake coin using the minimum number of weighings.

This project focuses on implementing a Divide-Into-Three algorithm, a recursive divide and conquer approach that splits the coin set into three nearly equal parts in each step. This method drastically reduces the number of required weighings compared to brute-force or linear search methods.

3.0 The Fake Coin Detection Problem Methodology

3.1 Input & Initialization

- Accept the number of coins (*n*) from the user.
- Randomly (or manually) assign one coin as the fake coin.
- Initialize flags to indicate whether the fake coin is lighter or heavier.
- Prepare a visual or logical representation of coins

3.2 Divide & Compare

- Divide the list of coins into three groups as equally as possible.
- Simulate a balance scale to weigh two of the three groups:
    - If both are equal, the fake coin is in the third group.
    - If unequal, depending on whether the fake coin is known to be lighter or heavier, determine which group contains it.

3.3 Recursive Detection

- Recursively call the divide-and-compare logic on the identified suspect group.
- Repeat the process until only one coin remains.
- This coin is identified as the fake one.
- Record or display the number of weighings performed.

3.4 Visualization & Output

- Provide a step-by-step visual representation of the steps
- Upon detecting the fake coin, display:
- The index/position of the fake coin.
- Whether it was lighter or heavier & The total number of weighings used.

FAKE COIN PROBLEM ALGORITHM:

```
 if length(C) == 1:
      return C[0] // Only one coin left; it is the fake one.
   Divide C into 3 groups: G1, G2, G3 such that:
      G1 = C[0...k-1], G2 = C[k...2k-1], G3 = C[2k...]
      (Groups may differ slightly in size)
   result = weigh(G1, G2)
   if result == "equal":
      return FAKE-COIN(G3) // Fake is in G3
   else if result == "left heavier":
      // Fake is either heavier in G1 or lighter in G2
      // Since we don’t know fake type, track both G1 and G2
      return FAKE-COIN(G1 + G2)
   else if result == "right heavier":
      // Fake is either heavier in G2 or lighter in G1
      return FAKE-COIN(G1 + G2)
```

BRUTE FORCE APPROACH ALGORITHM:

```
FAKE-COIN-BRUTE(C):
   for i from 1 to n-1:
      result = weigh(C[0], C[i])
      if result != "equal":
         return C[i]  // C[i] is fake or C[0] is fake
   return C[0]  // All coins are equal to C[0]; it must be fake
```

IMPLEMENTATION :

4.1 Input & Initialization

```
const totalCoins = parseInt(prompt("Enter the number of coins:"));
const coins = new Array(totalCoins).fill(10);
const fakeIndex = Math.floor(Math.random() * totalCoins);
const fakeIsLighter = true;
coins[fakeIndex] = fakeIsLighter ? 9 : 11;
let weighings = 0;
```

4.2 Divide & Compare

```
function weigh(group1, group2) {
   weighings++;
   const weight1 = group1.reduce((a, b) => a + b, 0);
   const weight2 = group2.reduce((a, b) => a + b, 0);
   if (weight1 === weight2) return "equal";
   return weight1 > weight2 ? "left heavier" : "right heavier";
}
function splitIntoThree(coinsArray) {
   const len = coinsArray.length;
   const third = Math.floor(len / 3);
   const group1 = coinsArray.slice(0, third);
   const group2 = coinsArray.slice(third, 2 * third);
   const group3 = coinsArray.slice(2 * third);
   return [group1, group2, group3];
}
```

4.3 Recursive Detection

```
function findFake(coinsArray, offset = 0) {
  if (coinsArray.length === 1) {
    return offset;
  }
  const [g1, g2, g3] = splitIntoThree(coinsArray);
  const result = weigh(g1, g2);
  if (result === "equal") {
    return findFake(g3, offset + g1.length + g2.length);
  } else if ((result === "left heavier" && fakeIsLighter) ||
         (result === "right heavier" && !fakeIsLighter)) {
    return findFake(g1, offset);
  } else {
    return findFake(g2, offset + g1.length);
  }
}
```

4.4 Visualization & Output

```
const fakeFoundAt = findFake(coins);
console.log(" Fake Coin Detection Complete!");
console.log(`Fake coin is at index: ${fakeFoundAt}`);
console.log(`Weight of fake coin: ${coins[fakeFoundAt]}
(${fakeIsLighter ? "Lighter" : "Heavier"})`);
console.log(`Total number of weighings: ${weighings}`);
```

DIFFERENCE BETWEEN BRUTEFORCE AND DIVIDE AND CONQUER:

Brute Force:

Concept:

- Compare coins one by one or in small groups.
- Try every possible combination or check until you find the fake coin.

How it works:

1. Start comparing two coins.
2. If they weigh the same, continue with the next pair.
3. When a mismatch is found, the lighter (or heavier) one is the fake.
4. Continue until the fake is found.

Time Complexity:

- Worst-case: O(n) comparisons
- Each coin might need to be compared individually

Divide and Conquer Approach (Divide into Three Groups)

Concept:

- Divide the coins into three groups as evenly as possible.
- Use the balance scale to compare two groups.
- The result narrows down the suspect group by 1/3 each time.

How it works:

1. Divide the n coins into 3 groups: A, B, C.
2. Compare Group A and Group B:
    - If equal → fake is in Group C.
    - If unequal → fake is in the lighter (or heavier) group.
3. Repeat the process with the suspect group.

4. Continue until only 1 coin remains.

Time Complexity:

- $O(\log_3 n)$ comparisons (much faster than brute force)

Pros:

- Highly efficient.
- Optimal use of the balance scale.
- Scales well for large inputs.

Cons:

- Slightly complex logic.
- Needs coin count to be divisible by 3 or handle remainders.

Feature	**Brute Force**	**Divide and Conquer**
Strategy	Linear comparisons	Recursive elimination
Time Complexity	$O(n)$	$O(\log_3 n)$
Efficiency	Low	High
Ideal for	Small number of coins	Large number of coins
Use of Balance Scale	Minimal	Optimal
Logic Complexity	Simple	Moderate

5.0. RESULTS:

Step-by-Step Procedure

Step 1: Label the coins

Let the coins be labeled from C1 to C9:

C1, C2, C3, C4, C5, C6, C7, C8, C9

Step 2: First Weighing – Divide into 3 groups

Split into 3 equal groups:

- Group A: C1, C2, C3
- Group B: C4, C5, C6
- Group C: C7, C8, C9

Weigh Group A vs Group B:

Weigh: (C1 + C2 + C3) vs (C4 + C5 + C6)

Step 3: Analyze First Weighing Result

Case 1: Equal

→ Fake coin is in Group C: C7, C8, C9

→ Proceed to Step 4A

Case 2: Left side heavier

→ Either:

- Fake coin is heavier in Group A or
- Fake coin is lighter in Group B

  → Fake is in C1–C3 or C4–C6

  → Proceed to Step 4B

Case 3: Right side heavier

→ Either:

- Fake coin is heavier in Group B or
- Fake coin is lighter in Group A
  → Fake is in C1–C3 or C4–C6
  → Proceed to Step 4B

Step 4A: If fake is in C7, C8, C9

Weigh: C7 vs C8

- If equal → Fake is C9
- If unequal → Compare weight direction with previous result (heavier or lighter) to decide

Result: Fake coin identified in 2 weighings

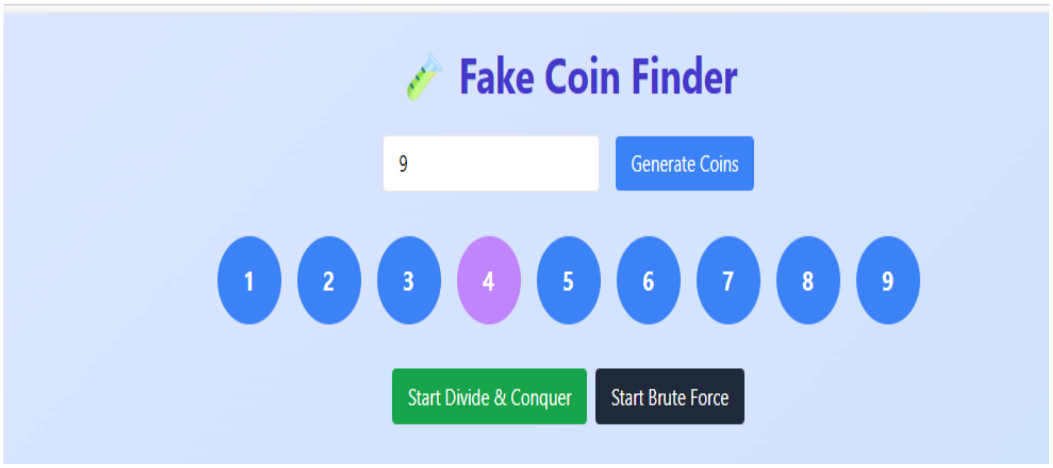
Step 4B: If fake is in C1–C6

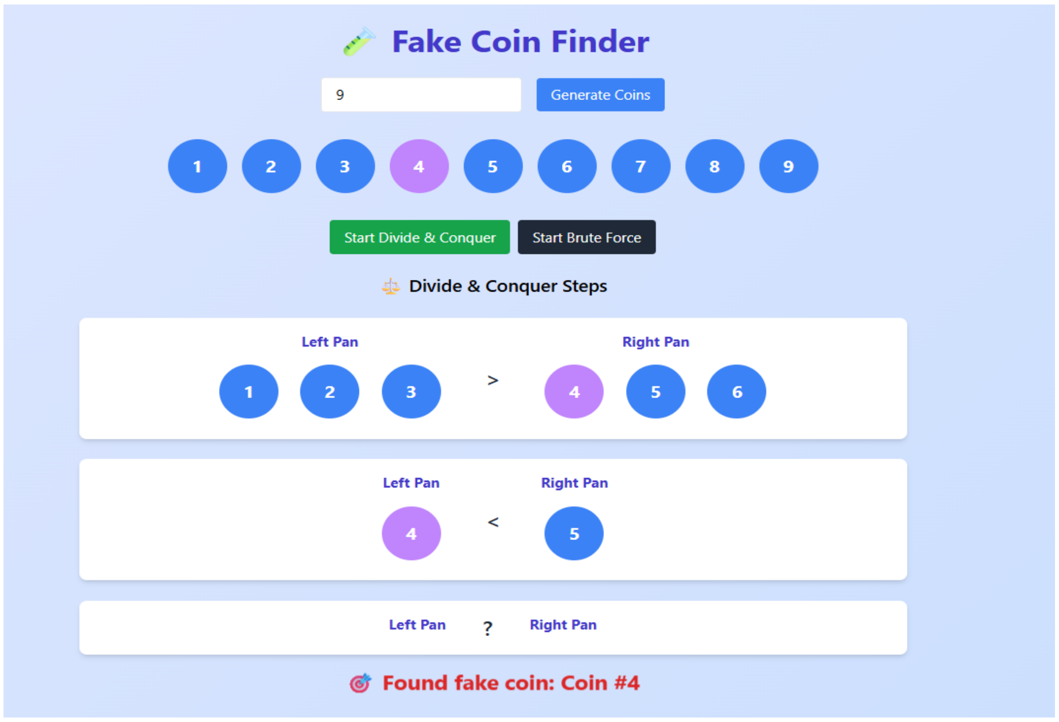
Choose one coin from each sub-group suspected:

Weigh: C1 vs C4

Sub-cases:

- If equal → Fake is in C2, C3, C5, C6
  Do: Weigh C2 vs C5
  - If equal → Fake is in C3 or C6 → Weigh them
  - If unequal → Compare direction to decide fake
- If unequal → Based on whether C1 or C4 is heavier/lighter and the original imbalance, you can identify the fake directly.





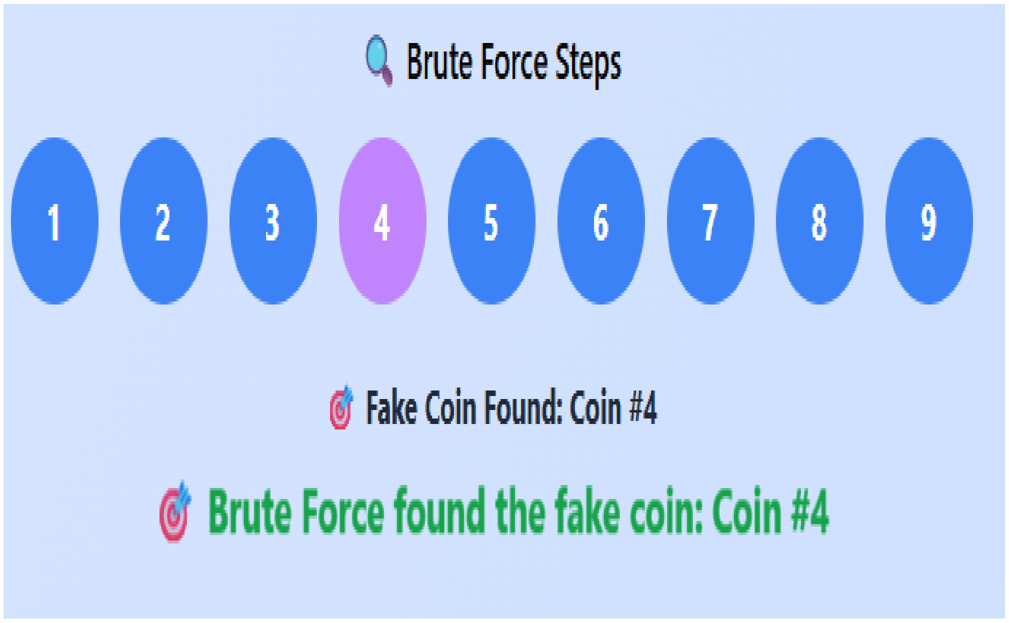
Divide & Conquer Steps

Left Pan 1 2 3 > Right Pan 4 5 6

Left Pan 4 < Right Pan 5

Left Pan ? Right Pan

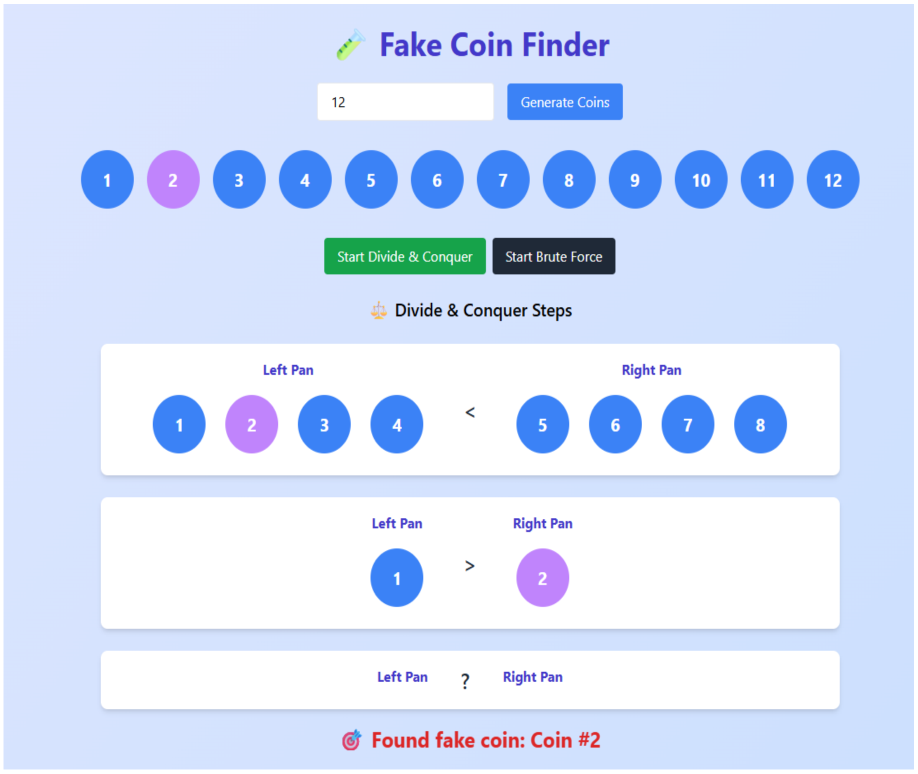
Found fake coin: Coin #4



🔍 Brute Force Steps

🎯 Fake Coin Found: Coin #4

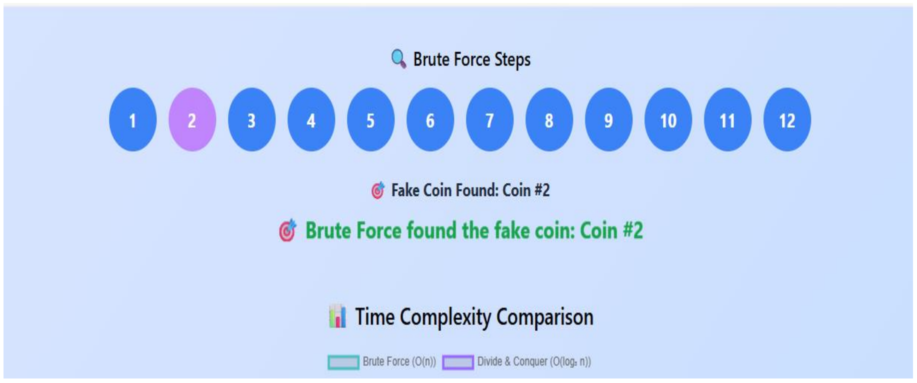
🎯 Brute Force found the fake coin: Coin #4



🧪 Fake Coin Finder

⚖️ Divide & Conquer Steps

🎯 Found fake coin: Coin #2

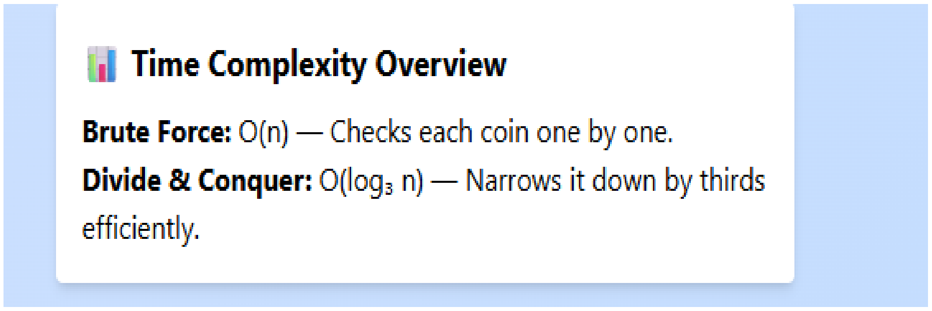


🔍 Brute Force Steps

🎯 Fake Coin Found: Coin #2

🎯 Brute Force found the fake coin: Coin #2

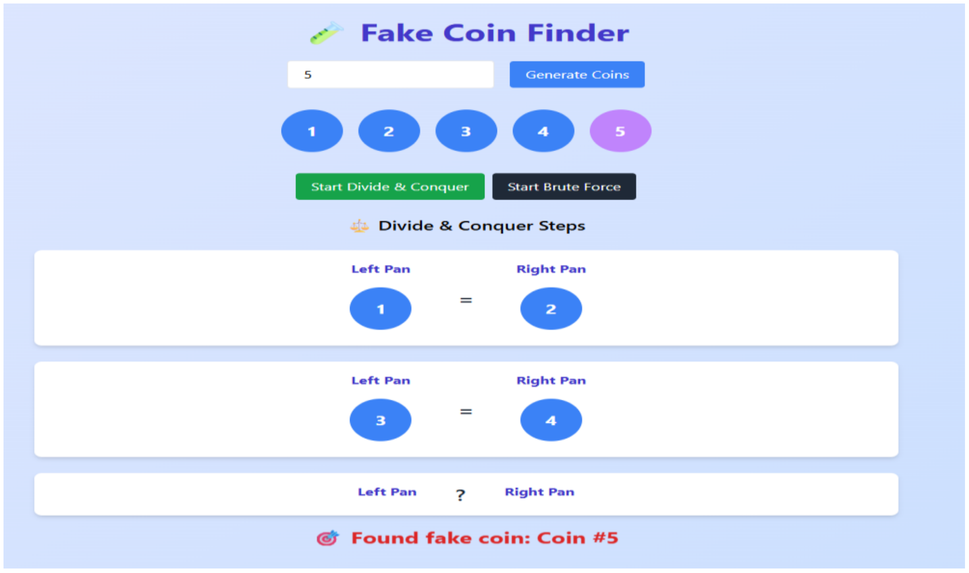
📊 Time Complexity Comparison



📊 Time Complexity Overview

Brute Force: O(n) — Checks each coin one by one.

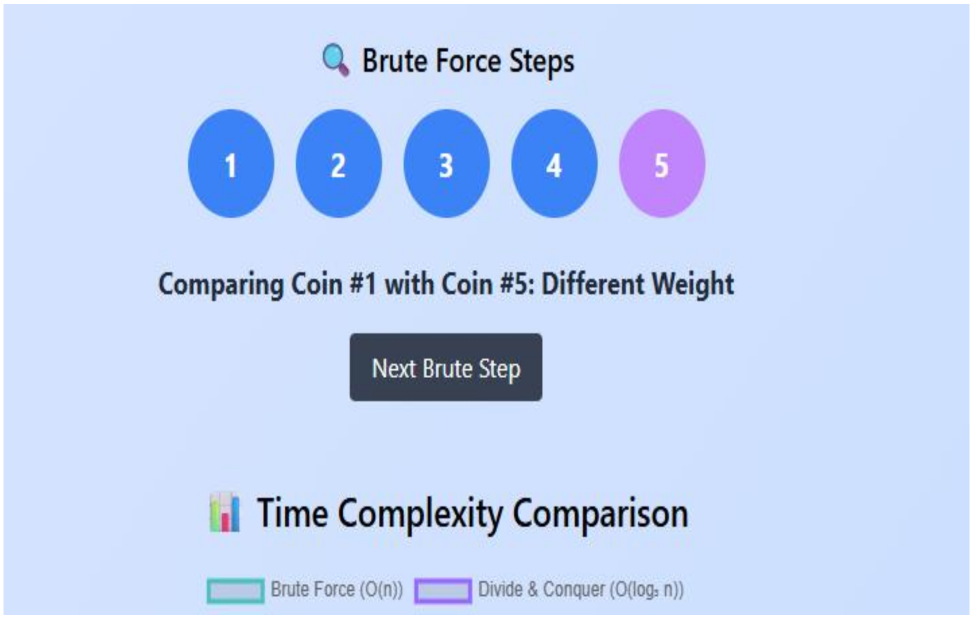
Divide & Conquer: O($\log_3$ n) — Narrows it down by thirds efficiently.

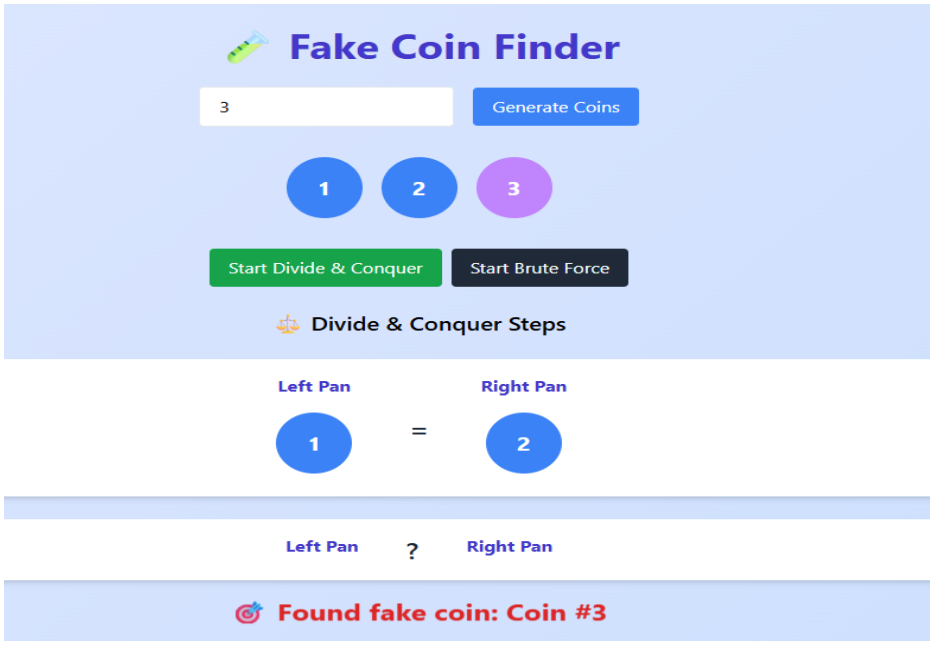


Fake Coin Finder

⚖️ Divide & Conquer Steps

🎯 Found fake coin: Coin #5


Brute Force Steps
1
2
3
4
5
Comparing Coin #1 with Coin #5: Different Weight
Next Brute Step
Time Complexity Comparison
Brute Force (O(n))
Divide & Conquer (O(log₃ n))



GITHUB LINK: https://github.com/shreyaajahan/DAA-micropro