|  |  |
| --- | --- |
|  | Department of Computer Science and Engineering  Chandpur Science and Technology University |

**LAB-04**

**Course Title**: Algorithm Design and Analysis Sessional

**Course Code**:CSE 2202

**Submitted To-**

**Mostafiz Ahammed**

*Lecturer****Department of Computer Science and Engineering****Chandpur Science and Technology University*

**Submitted By-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl** | Class Roll | Name | Batch | Signature |
| 01 | B210101022 | Md. Rabbi Ahammed | CSE-21 | rabbi |

**Date of Submission: July, 2025**

# Experiment 01: *Coin Change Problem Using Greedy Approach*

## Objective

• Implement the Coin Change problem using the greedy algorithm.

• Evaluate its correctness and analyze where it works correctly and where it may fail.

• Understand the limitations of the greedy approach for this problem.

## Algorithm: Greedy Approach for Coin Change

1. Sort the coin denominations in descending order.

2. Pick the largest coin denomination that is less than or equal to the remaining amount.

3. Subtract the value of the chosen coin from the remaining amount.

4. Repeat steps 2 and 3 until the remaining amount is 0.

## Theoretical Solution

• The greedy approach works optimally for canonical coin systems (like US currency: 1, 5, 10, 25).

• It fails for some coin denominations because choosing the locally optimal (largest) coin may not lead to a globally optimal solution.

• Example of failure: Coins = {1, 3, 4}, Value = 6

Greedy picks: 4 + 1 + 1 = 3 coins

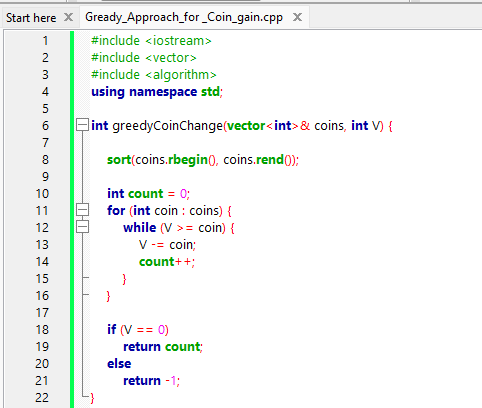
Optimal solution: 3 + 3 = 2 coins

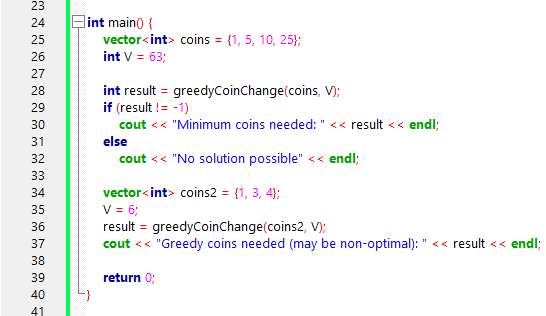
## Practical Work

## a. Pseudocode

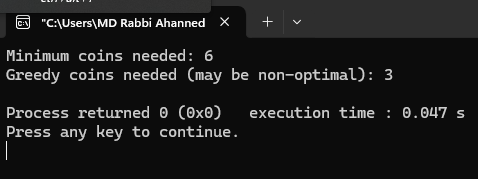
function greedyCoinChange(coins[], V):  
 sort coins in descending order  
 count = 0  
 for each coin in coins:  
 while V >= coin:  
 V = V - coin  
 count = count + 1  
 if V == 0:  
 return count  
 else:  
 return -1

## b. Source Code in C++





**Output:**

****

## Analysis Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Coin Set | Value (V) | Greedy Result | Optimal Result | Comments |
| 1 | {1, 5, 10, 25} | 63 | 6 | 6 | Works correctly (canonical) |
| 2 | {1, 3, 4} | 6 | 3 | 2 | Greedy fails to find optimum |
| 3 | {1, 7, 10} | 14 | 4 | 2 | Greedy fails |
| 4 | {1, 2, 5, 10} | 18 | 4 | 4 | Works correctly |

## Observations

• Greedy algorithm is simple and fast.

• Works perfectly for standard currency denominations.

• May produce suboptimal results for arbitrary coin denominations.

• It's easy to implement but should be used with caution in non-canonical systems.

## Challenges

• Identifying the cases where greedy fails.

• Understanding that greedy approach is not always optimal.

• Need to explore dynamic programming or other algorithms for a guaranteed optimal solution.

## Conclusion

• The greedy algorithm for the coin change problem works efficiently and correctly for canonical coin systems.

• However, it does not guarantee optimal solutions for all coin denominations.

• For non-standard coin sets, a more sophisticated method like dynamic programming is recommended.

• This experiment helped understand both the power and limitation of greedy algorithms.