|  |  |  |  |
| --- | --- | --- | --- |
| **Academic Year** | **2025 - 26** | **Experiment No.** | **4** |
| **Course & Semester** | **S.E. – Sem. III** | **Subject Name** | **Analysis of Algorithm** |
| **Experiment Type** | **Software Performance** | **Subject Code** | **25PCC12CS05** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Student:** |  | **Roll No.:** |  |
| **Date of Performance:** |  | **Date of Submission:** |  |
| **LO Mapping** | 25PCC12CS05.1: Analyze the time and space complexity of algorithms.  25PCC12CS05.3: Apply greedy strategy to solve optimization problem. | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Indicator** | **Poor** | **Average** | **Good** | | Timeline Maintains submission deadline (3) | Submission not done (0) | One or More than One week late (1-2) | Maintains deadline (3) | | Completion and Organization (3) | N/A | Document is just acceptable (1-2) | Completed whole document and neatly organized (3) | | Program Performance (2) | Could not perform at all (0) | Implemented few parts (1) | Full implementation (2) | | Knowledge In depth knowledge of the Experiment (2) | Unable to answer questions (0) | Unable to answer few questions (1) | Able to answer all questions (2) | |
| **Assessment Marks:**   |  |  | | --- | --- | | Timeline |  | | Completion and Organization |  | | Program Performance |  | | Knowledge |  | |
| Total: (Out of 10) |
| Teacher’s Sign: Student Sign: |

**Experiment No. 4**

**AIM:** To Identify and implement an algorithm to be used by vending machines to determine the optimal combination of coins to give as change to customers.

**THEORY:**

The aim of this experiment is to design an algorithm for vending machines to determine the optimal combination of coins to provide as change. This problem, known as the **Coin Change Problem**, can be efficiently solved using a **Greedy Algorithm**. The algorithm works by selecting the largest coin denomination that does not exceed the remaining change to be given, and subtracting that from the total. This process repeats until the required change is given.

The **Greedy Algorithm** is effective because it minimizes the number of coins returned by always choosing the largest possible coin at each step. This approach is efficient with a time complexity of O(n), where n is the number of coin denominations. The algorithm ensures that vending machines can quickly and optimally provide change to customers, enhancing user experience and reducing processing time.

**ALGORITHM:**

 **Input**:

* A list of available coin denominations.
* The target amount that needs to be made using the coins.

 **Sort the coins** in descending order:

* Sort the coins in descending order so that the largest coin is considered first.

 **Initialize variables**:

* Set remaining\_amt as the target amount.
* Initialize total\_coins to 0, which will store the total number of coins used.
* Set num\_coins to store the number of coins used for each denomination.

 **Iterate over the sorted coins**:

* For each coin, check how many times it can fit into the remaining\_amt.
* Calculate num\_coins = floor(remaining\_amt / coin).
* Add num\_coins to total\_coins.
* Subtract the value of the coins used from the remaining\_amt (i.e., remaining\_amt -= num\_coins \* coin).

 **Output the result**:

* If any coins were used, print the number of coins for each denomination.
* If remaining\_amt == 0, print the total number of coins used.
* If remaining\_amt > 0 after iterating through all the coins, print that the amount cannot be made with the given coins.

 **End**.

**CODE:**

#include <stdio.h>

int n;

void sort(int coins[])

{

    int i, j, max\_idx, temp;

    for(i = 0; i < n - 1; i++){

        max\_idx = i;

        for(j = i + 1; j < n; j++){

            if(coins[j] > coins[max\_idx])

                max\_idx = j;

        }

        temp = coins[i];

        coins[i] = coins[max\_idx];

        coins[max\_idx] = temp;

    }

}

int main() {

    int i, amount, remaining\_amount;

    printf("Enter number of coins: ");

    scanf("%d", &n);

    int coins[n];

    printf("Enter coins:");

    for(i=0;i<n;i++){

        scanf("%d", &coins[i]);

    }

    printf("Enter amount: ");

    scanf("%d", &amount);

    remaining\_amount = amount;

    sort(coins);

    for(i=0; i<n; i++){

        printf("You will need %d of coins %d\n", remaining\_amount/coins[i], coins[i]);

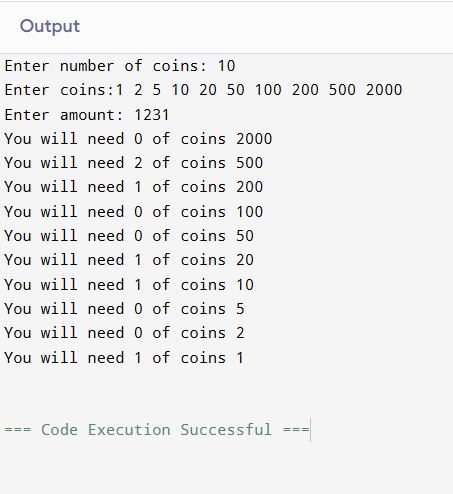
        remaining\_amount = remaining\_amount%coins[i];

    }

    return 0;

}

**OUTPUT:**

****

**POST LAB QUESTIONS**

**1. Can the Greedy Algorithm fail to find the optimal solution? Provide an example.**

**Ans.** Yes, if the coin denominations are not canonical (don’t form a standard system).

Example: Coins {1, 3, 4}, amount 6.

* + Greedy: 4 + 1 + 1 = 3 coins
  + Optimal: 3 + 3 = 2 coins.

**2. What data structures are used in your implementation, and why are they appropriate?**

**Ans.**

Array (coins[]) – Stores coin denominations for iteration and sorting.

Variables (amount, remaining\_amount) – Track the calculation state efficiently.  
Arrays are appropriate due to fixed, small input size and constant-time access.

**3. Why is the Greedy Algorithm suitable for solving the Coin Change Problem in vending machines?**

**Ans.** Coin denominations in real-world systems (e.g., 1, 2, 5, 10, 20, 50, 100, etc.) are canonical.

Guarantees optimal results quickly.

Low computational overhead, suitable for fast, real-time dispensing.

**CONCLUSION:**

The Greedy Algorithm is efficient and practical for coin change problems in real-world systems with canonical denominations, such as vending machines. By leveraging simple data structures like arrays and straightforward logic, it ensures quick computation with minimal resources. However, it is not universally optimal for all coin systems, highlighting the need to analyze the denomination structure before applying the greedy approach.