**Algorithm Design-II (CSE 4131)**

**TERM PROJECT REPORT**

**(March'2023-July'2023)**

**On**

**Coin Change Problem using Brute Force and Greedy Algorithm**

***Submitted By***

|  |  |
| --- | --- |
| **Group:** | **H11** |
| **Debopam Aich** | **2141001033** |
| **Shaikh Shohail Parwej** | **2141016146** |
| **Dhrub Khatry** | **2141011043** |

**B.Tech. 4th Semester CSE (H)**



**Department of Computer Science and Engineering**

**Institute of Technical Education and Research**

**Siksha ‘O’ Anusandhan Deemed To Be University**

**Bhubaneswar, Odisha-751030**

**(June 2023)**

**DECLARATION**

We, Debopam Aich (Regd no: 2141001033), Shaikh Shohail Parwej (Regd no: 2141016146), Dhrub Khatry (Regd no: 2141011043), do hereby declare that this term project entitled “**Coin Change Problem using Brute Force and Greedy Algorithm**” is an original project work done by us and has not been previously submitted to any university or research institution or department for the award of any degree or diploma or any other assessment to the best of our knowledge.

|  |  |  |
| --- | --- | --- |
| **Debopam Aich** | **(2141001033):** |  |
| **Shaikh Shohail Parwej** | **(2141016146):** |  |
| **Dhrub Khatry** | **(2141011043):** |  |

## CERTIFICATE

This is to certify that the thesis entitled ” **Coin Change Problem using Brute Force and Greedy Algorithm**” submitted by Debopam Aich (Regd no: 2141001033), Shaikh Shohail Parwej (Regd no: 2141016146), Dhrub Khatry (Regd no: 2141011043) of B.Tech. 4th Semester Comp. Sc. and Engg., ITER, SOADU is absolutely based upon their own work under my guidance and supervision.

The term project has reached the standard fulfilling the requirement of the course Algorithm Design 2 (CSE4131). Any help or source of information which has been available in this connection is duly acknowledged.

Rangaballav Pradhan

Assistant Professor,

Department of Comp. Sc. and Engg.

ITER, Bhubaneswar 751030,

Odisha, India

Prof. (Dr.) Debahuti Mishra

Professor and Head,

Department of Comp. Sc. and Engg.

ITER, Bhubaneswar 751030,

Odisha, India

**ABSTRACT**

Automatic money exchange systems are very rarely found but are needed at

certain times. The purpose of this project is to make it easier for people who want to

exchange money from big to small nominal. The obstacle faced in this exchange system

is the use of algorithms needed to run the system. Some method that can be used to

design and implement this system is using Greedy Algorithm and Brute force

algorithms. By using this greedy algorithm, a problem with the shortest route search

technique can be completed quickly, but the results generated by the greedy algorithm

are not always optimal, while brute force can solve the problem of money exchange

optimally but need a longer time. In this subject we trying to solve this problem by

making an application using python language to implement the greedy and brute force

algorithm in solving of this problem. After making the code algorithm and

implementing the greedy and brute force method, we conclude that our program can

solve this coins exchange problem just in a second but sometimes the result is not

optimal (with the most smallest fraction), but if using brute force algorithm we can get

the optimal total of money changes but the time will extremely increase up to 20 times

higher depend on the amount of the total money that want to be changes.

in this experiment the user will input the money that want to change, in the program we

already have a predetermined amount and money that has been prepared to exchange

money.

Automatic money exchange systems are very rarely found but are needed at

certain times. The purpose of this project is to make it easier for people who want to

exchange money from big to small nominal. The obstacle faced in this exchange system

is the use of algorithms needed to run the system. Some method that can be used to

design and implement this system is using Greedy Algorithm and Brute force

algorithms. By using this greedy algorithm, a problem with the shortest route search

technique can be completed quickly, but the results generated by the greedy algorithm

are not always optimal, while brute force can solve the problem of money exchange

optimally but need a longer time. In this subject we trying to solve this problem by

making an application using python language to implement the greedy and brute force

algorithm in solving of this problem. After making the code algorithm and

implementing the greedy and brute force method, we conclude that our program can

solve this coins exchange problem just in a second but sometimes the result is not

optimal (with the most smallest fraction), but if using brute force algorithm we can get

the optimal total of money changes but the time will extremely increase up to 20 times

higher depend on the amount of the total money that want to be changes.

in this experiment the user will input the money that want to change, in the program we

already have a predetermined amount and money that has been prepared to exchange

money.

Automatic money exchange systems are very rarely found but are needed at certain times. The purpose of this project is to make it easier for people who want to exchange money from big to small nominal. The obstacle faced in this exchange system is the use of algorithms needed to run the system. Some method that can be used to design and implement this system is using Greedy Algorithm and Brute force algorithms. By using this greedy algorithm, a problem with the shortest route search technique can be completed quickly, but the results generated by the greedy algorithm are not always optimal, while brute force can solve the problem of money exchange optimally but need a longer time. In this subject we trying to solve this problem by making an application using python language to implement the greedy and brute force algorithm in solving of this problem. After making the code algorithm and implementing the greedy and brute force method, we conclude that our program can solve this coins exchange problem just in a second but sometimes the result is not optimal (with the most smallest fraction), but if using brute force algorithm we can get the optimal total of money changes but the time will extremely increase up to 20 times higher depend on the amount of the total money that want to be changes. in this experiment the user will input the money that want to change, in the program we already have a predetermined amount and money that has been prepared to exchange money.

**CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | Title Page | i |
|  | Declaration | ii |
|  | Certificate | iii |
|  | Abstract | iv |
|  |  |  |
| **1.** | **INTRODUCTION** | **1** |
|  | * 1. Overview   2. Motivation   3. Problem Definition   4. Mathematical Formulation   5. Specifications of the algorithm |  |
|  |  |  |
| **2.** | **DESIGNING ALGORITHMS** | **3** |
|  | 2.1 Mathematical Formulation  2.2 Pseudocode  2.3 Description of steps  2.4 Examples |  |
|  |  |  |
| **3.** | **ANALYSIS OF ALGORITHM** | **6** |
|  |  |  |
| **4.** | **IMPLEMENTATION** | **7** |
|  |  |  |
| **5.** | **RESULTS AND DISCUSSION** | **9** |
|  |  |  |
| **6.** | **LIMITATIONS** | **12** |
|  |  |  |
| **7.** | **FUTURE ENHANCEMENTS** | **13** |
|  |  |  |
|  | **REFERENCES** | **15** |

1. **INTRODUCTION**

**1.1 Overview:**

The coin change problem is a classic algorithmic problem that involves finding the minimum number of coins needed to make a given amount of change. It is a widely studied problem in computer science and has applications in various domains, such as financial systems and vending machines.

This report will discuss the comparison of Greedy and Brute Force algorithm in the example of optimization problem, namely the problem of money exchange. In this case we want to see and know which algorithm has an optimum result and more fast. In this experiment we will exchange money in Rupee currency. In this case we will Exchange money with a predetermined amount and money that has been prepared to exchange money. The money will be exchanged using smaller money and will give the minimum and equivalent to the money that has been exchanged. The method used to solve this coin-change problem is Brute Force and Greedy Algorithm which will be used for comparison.

**1.2 Motivation:**

The coin change problem arises in situations where we need to provide optimal change to customers or minimize the number of coins used in transactions. Solving this problem efficiently is crucial for designing efficient algorithms for financial systems and improving user experience in vending machines or cash registers.

**1.3 Problem Definition:**

Given a set of coins with different denominations and a target amount, the goal of the coin change problem is to find the minimum number of coins needed to make the target amount. We assume an unlimited supply of coins for each denomination.

**1.4 Mathematical Formulation:**

Let's define the problem formally:

- Input: A set of coin denominations {d1, d2, ..., dn} and a target amount T.

- Output: The minimum number of coins needed to make the amount T using the given denominations.

**1.5 Specifications of the algorithm:**

In this report, we will explore two approaches to solve the coin change problem: brute force and greedy algorithms.

- Brute force algorithm: This approach involves considering all possible combinations of coins and selecting the one with the minimum number of coins.

- Greedy algorithm: The greedy approach selects the largest denomination coin that is less than or equal to the remaining amount and repeats this process until the target amount is reached.

1. **DESIGNING ALGORITHMS**

In this section, we will discuss the design of the brute force and greedy algorithms for solving the coin change problem.

**2.1 Mathematical Formulation:**

The mathematical formulation of the coin change problem has already been defined in the introduction. We have the following inputs and outputs:

**Input:**

- A set of coin denominations {d1, d2, ..., dn}.

- Target amount T.

**Output:**

- The minimum number of coins needed to make the amount T using the given denominations.

**2.2 Pseudocode:**

Now, let's provide the pseudocode for both the brute force and greedy algorithms.

**Brute Force Algorithm:**

function coinChangeBruteForce(coins, amount):

if amount == 0:

return 0

minCoins = INFINITY

for coin in coins:

if coin <= amount:

numCoins = 1 + coinChangeBruteForce(coins, amount - coin)

minCoins = min(minCoins, numCoins)

return minCoins

**Greedy Algorithm:**

function coinChangeGreedy(coins, amount):

coins.sort(reverse=True)

numCoins = 0

for coin in coins:

while coin <= amount:

amount -= coin

numCoins += 1

return numCoins

**2.3 Description of Steps:**

Let's describe the steps of each algorithm:

**Brute Force Algorithm:**

1. If the target amount is 0, we don't need any coins, so we return 0.

2. Initialize the minimum number of coins needed to a very large value (INFINITY).

3. Iterate through each coin in the given set of coins.

4. If the coin's value is less than or equal to the remaining amount, recursively call the `coinChangeBruteForce` function with the updated amount (amount - coin).

5. Calculate the number of coins needed for the current combination by adding 1 to the result of the recursive call.

6. Update the minimum number of coins if the current combination requires fewer coins.

7. Return the minimum number of coins needed.

**Greedy Algorithm:**

1. Sort the coins in descending order.

2. Initialize the number of coins needed to 0.

3. Iterate through each coin in the sorted list of coins.

4. While the coin's value is less than or equal to the remaining amount, subtract the coin value from the amount and increment the number of coins needed.

5. Repeat step 4 until the remaining amount becomes 0.

6. Return the total number of coins needed.

**2.4 Examples:**

Let's demonstrate the two algorithms with an example:

**Example:**

Consider the following input:

- Coin denominations: {1, 5, 10, 25}

- Target amount: 30

**Brute Force Algorithm:**

coinChangeBruteForce(coins = {1, 5, 10, 25}, amount = 30)

**Step-by-step execution:**

- Select coin 25, recursively call `coinChangeBruteForce(coins, 5)`.

- Select coin 5, recursively call `coinChangeBruteForce(coins, 0)`.

- Return 0 since the amount is 0.

- Return 1 + 0 = 1 (1 coin of denomination 5 needed).

- Return 1 + 1 = 2 (1 coin of denomination 25 and 1 coin of denomination 5 needed).

The minimum number of coins needed is 2.

**Greedy Algorithm:**

coinChangeGreedy(coins ={1, 5, 10, 25}, amount = 30)

**Step-by-step execution:**

- Select coin 25, subtract 25 from the amount (30 - 25 = 5), increment the number of coins.

- Select coin 5, subtract 5 from the amount (5 - 5 = 0), increment the number of coins.

The minimum number of coins needed is 2.

These examples illustrate the execution of the brute force and greedy algorithms for the given input.

In the next section, we will analyze and compare the two algorithms in terms of time complexity, space complexity, and optimality.

1. **ANALYSIS OF ALGORITHM**

**3.1 Greedy Algorithm**

The Greedy algorithm is the algorithm most often used in solving optimization problems. Greedy itself is taken from English, which means “serakah”, “tamak” or “rakus”. In accordance with these meanings, the Greedy principle is "Take what you can get now". The greedy algorithm will form a step-by-step solution. At each step, we must make choices that can produce optimum results. Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are best fit for Greedy

**3.2 Brute Force Algorithm**

Brute Force Algorithms refers to a programming style that does not include any shortcuts to improve performance, but instead relies on sheer computing power to try all possibilities until the solution to a problem is found. The Brute force algorithm is a straightforward approach to solving a problem, usually based on a problem statement. The brute force algorithm solves problems very simply, directly and in a clear way (obvious way).

- Analyse the time complexity

Since in our program we use Brute Force and Greedy Algorithm to find the optimum and minimal money change so, the time complexity are:

1. Brute Force

Number of coins in the array: 4

The number of coins to be exchanged: n

Time Complexity: 4^n

1. Greedy Algorithm

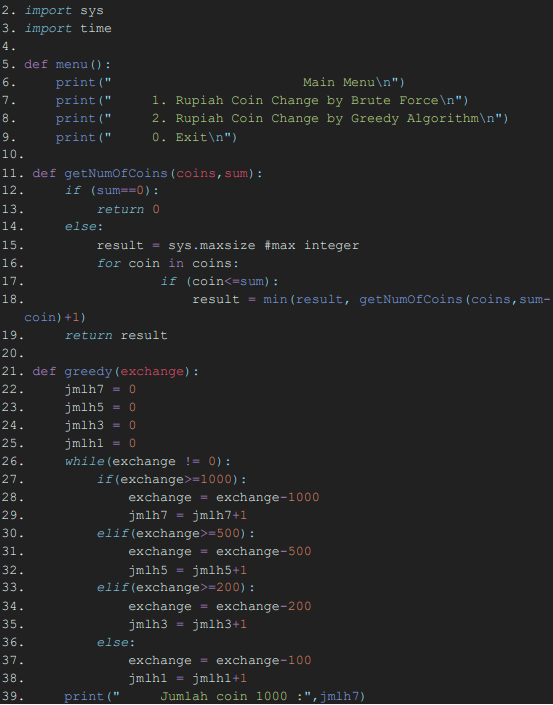
Number of coins in the array:4

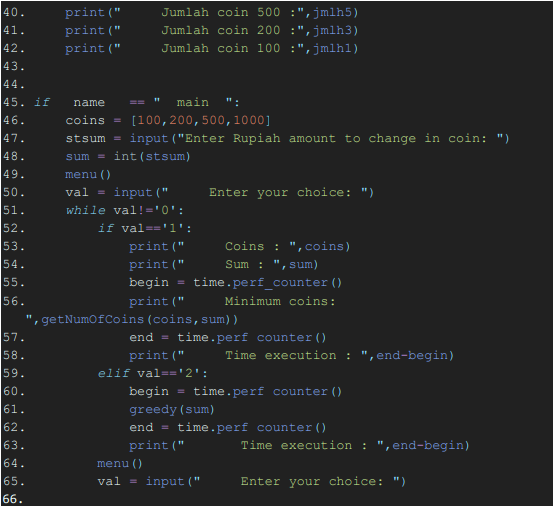
The number of coins to be exchanged: n

Time Complexity: N

1. **IMPLEMENTATION**

**4.1 Code:**





**4.2 Description:**

The brute force algorithm utilizes recursion and checks all possible combinations of coins to find the minimum number of coins needed. It iterates through each coin and makes recursive calls, updating the minimum number of coins each time a new minimum is found.

The greedy algorithm sorts the coins in descending order and iterates through them. It repeatedly subtracts the largest possible coin from the remaining amount until the amount becomes zero. The number of coins used is incremented in each iteration.

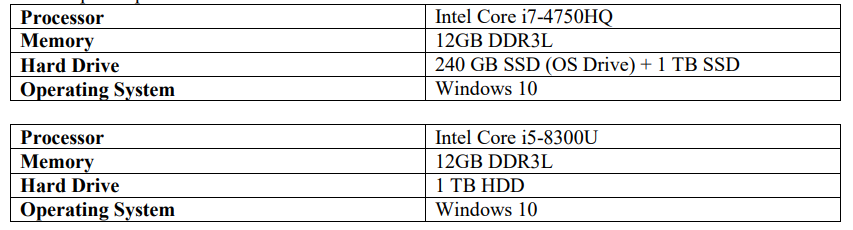
1. **RESULTS AND DISCUSSION**

**5.1 Study case**

The program has provided Rupee in the form of fractions [100, 200, 500, 1000] and will receive input money that will be exchanged by the user. Users can see the minimum output of money that has been exchanged and the execution of the time needed by the program.

**5.2 Computer specifications**

Table 1. Showing the specifications of the used



On this experiment we used different spec of laptop like we explain on the Computer specifications above. A significant sight is on the processor that used to run our code one using Intel Core i7 gen-4 with 240GB SSD on the other hand using Intel core i5 gen-8 but without SSD on board.

Below is the result of experiment that tested in two different specs with the same input (3000 rupiah) and same code of algorithm to.

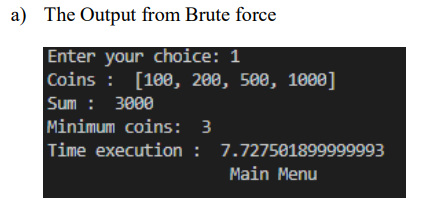


Figure 1. Brute Force (using intelcore i5)

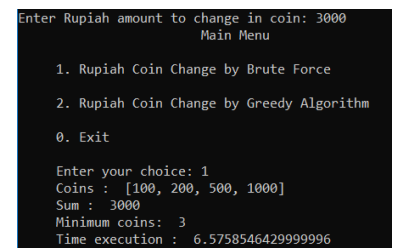


Figure 2. Brute Force (using intelcore i7)

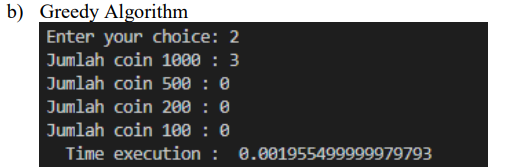


Figure 3. Greedy Algorithm (using intelcore i5)

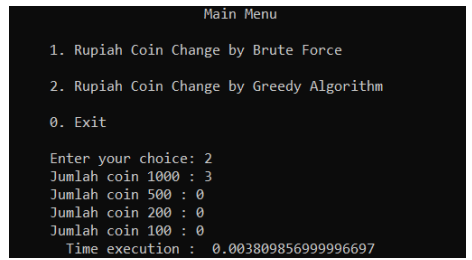
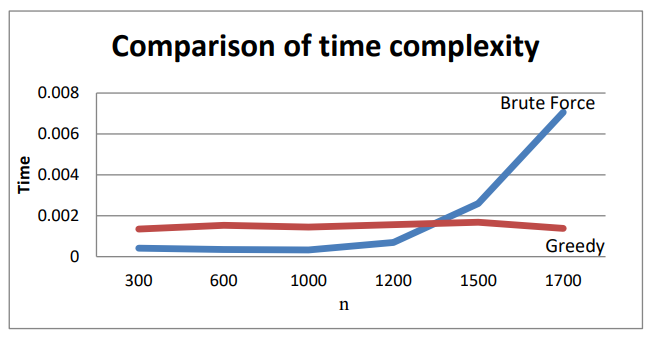


Figure 4. Greedy Algorithm (using intelcore i7)

After several test we conclude that hardware is influence on speed of processing the code like in the result picture above, in brute force method it took 1.1 second faster for Intel i7 with SSD than Intel i5 that using only common HDD.



Graph 1. Comparion of Time Complexity from two algorithms.

From graph 1, the greedy algorithm (red) and brute force (blue) have different time executions. as in the example of exchanging money for Rs.1,700, for bruteforce time in problem solving is greedy algorithm almost 7x faster to solve the coin change problem when greedy is done. From this experiment it can be concluded that the greedy algorithm is faster even though the results are not always optimum.

For Brute force the more money will be exchanged, the longer the execution time will be, while the Greedy Algorithm has an execution time that is faster than brute force.

1. **LIMITATIONS**

Both the brute force and greedy algorithms for the coin change problem have certain limitations. Let's discuss them:

**6.1 Brute Force Algorithm Limitations:**

1. Exponential Time Complexity: The brute force algorithm explores all possible combinations of coins, leading to exponential time complexity. As the number of coins and the target amount increase, the algorithm's execution time grows significantly, making it impractical for large inputs.

2. Redundant Calculations: The algorithm may perform redundant calculations as it explores all combinations, even if they have already been evaluated. This leads to inefficiency in terms of time and computational resources.

3. Lack of Optimization: The brute force algorithm does not incorporate any optimization techniques to prune unnecessary branches or exploit specific properties of the problem. It relies solely on exhaustive exploration.

**6.2 Greedy Algorithm Limitations:**

1. Not Always Optimal: The greedy algorithm does not always provide the optimal solution for the coin change problem. It may produce a solution that uses the minimum number of coins in some cases, but it can fail to do so in others. This is because it selects coins greedily without considering the overall optimal combination.

2. Denomination Dependency: The greedy algorithm's effectiveness depends on having a specific property called "coin denomination dependency," where larger denominations are multiples of smaller denominations. If the coin denominations do not exhibit this property, the greedy algorithm may fail to produce the correct result.

3. Lack of Flexibility: The greedy algorithm relies on sorting the coins in descending order. If the coin denominations need to be dynamically updated or rearranged, the algorithm may not be suitable without modifications.

4. Limited Applicability: The greedy algorithm is primarily useful for finding approximate solutions or providing quick outputs. It may not be suitable for scenarios where an optimal solution is necessary.

Overall, both the brute force and greedy algorithms have their limitations in terms of time complexity, optimality, and applicability to certain problem instances. It is important to consider these limitations when choosing an algorithm to solve the coin change problem, and alternative algorithms such as dynamic programming can be explored to overcome these limitations.

1. **FUTURE ENHANCEMENTS**

While the brute force and greedy algorithms for the coin change problem have their limitations, there are potential enhancements and modifications that can be made to improve their performance and address specific challenges. Here are some future enhancements that can be considered:

**7.1 Brute Force Algorithm Enhancements:**

1. Memoization: Implementing memoization techniques such as dynamic programming can help avoid redundant calculations in the brute force algorithm. By storing the results of previously computed subproblems, we can significantly improve the algorithm's efficiency and reduce the time complexity.

2. Pruning Techniques: Introduce pruning techniques to eliminate unnecessary branches during the exploration process. This can involve adding conditions to stop exploring certain combinations that are known to be suboptimal or have already been evaluated.

3. Optimal Substructure: Explore the problem's optimal substructure to identify patterns and properties that can be exploited to reduce the search space. This can help optimize the brute force algorithm by avoiding unnecessary computations.

**7.2 Greedy Algorithm Enhancements:**

1. Modified Greedy Strategies: Explore alternative greedy strategies that take into account additional factors beyond the coin denominations. For example, considering the quantity of available coins for each denomination or the value of the remaining amount can lead to improved solutions.

2. Backtracking: Combine the greedy algorithm with backtracking to explore different possibilities when the greedy approach fails to provide the optimal solution. This can help overcome the limitation of the greedy algorithm's optimality and search for alternative combinations.

3. Hybrid Approaches: Combine the greedy algorithm with other techniques, such as dynamic programming or branch and bound, to achieve better results. This can involve using the greedy algorithm as a heuristic to guide the search in a more efficient manner.

**7.3 General Enhancements:**

1. Algorithmic Optimization: Investigate and analyze the problem structure to develop more efficient algorithms specifically tailored for the coin change problem. This may involve designing algorithms with improved time complexity or developing approximation algorithms that provide near-optimal solutions.

2. Machine Learning Techniques: Utilize machine learning algorithms to learn patterns and heuristics from large datasets of coin change instances. This can help in predicting optimal solutions or improving the performance of existing algorithms.

3. Problem Variations: Explore variations of the coin change problem, such as limited coin supplies or introducing constraints on the available denominations. Investigate how the existing algorithms can be modified or extended to handle these variations effectively.

These future enhancements can lead to improved algorithms for the coin change problem, addressing the limitations of the brute force and greedy approaches. By incorporating advanced techniques and exploring problem-specific properties, it is possible to achieve more efficient and optimal solutions.

**REFERENCES**

1. Kleinberg, J., & Tardos, E. (2006). *Algorithm design*. Pearson Education India.

2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

3. Anindhika, Dita. "The Complexity of Algorithms for Solving Coin Exchange Problems with Greedy Algorithm", Informatics Engineering Study Program, Bandung Institute of Technology.

4. Logianto, Albert. "Solving Money Exchange Problems with Dynamic Programs". Program Study Electrical and Information Engineering, Bandung Institute of Technology.

5. https://dosenit.com/ilmu-komputer/pengertian-algoritma-brute-force-dan-greedy