**DESIGN AND ANALYSIS OF ALGORITHMS**

**PROJECT 2B**

**CS - 5120**

**Project Report**

**Name of Student: Sakshi Shastri**

**Professor Name: Dr. Sankardas Roy**

**Fall 2023**

**Department of Computer Science**

**Bowling Green State University, Ohio**

**TABLE OF CONTENTS**

1. THEORITICAL ANALYSIS

* WHAT IS COIN-CHANGE PROBLEM
* APPROACHES

a. RECURSIVE APPROACH

b. GREEDY APPROACH

2. COMPARISION

* TABLES AND GRAPHS

3. ANALYSIS AND DISCUSSION

4. OUTPUT GENERATED

**Theoretical Analysis**

* **What is a Coin Change Problem?**

The main objective of this problem is to find the minimum number of coins as change for a given value(n) from a set of denominations (k-size integer array d (i.e., d[1, 2, …, k])).

There are 2 approaches for Coin Change Problem:

1. Recursive

2. Greedy

3. Dynamic

Using these two approaches we have to test 2 types of data.

**I. US Coin System**

Test for different value of n = 11, 23, 31, 51, 73, 83, 91, 99

Denomination -> k = 4 and d[1] = 1 (penny), d[2] = 5 (nickel), d[3] = 10 (dime), and d[4] = 25 (quarter).

**II. Weird Coin System**

Test for n = 69

Denomination -> k = 5, and d [1] = 1, d[2] = 5, d[3] = 10, d[4] = 23,

d [5] = 25

* **Approach**

**1. Recursive**

We can solve coin change problems using recursion. So, basically, we have two options for each coin- either to include it in solution or to exclude it from solution. But it is extremely inefficient because there is one problem that this approach solves the sub-problems again and again. A lot of time is wasted in calculating the old result.

Time complexity: O(2^(m+n)) (Exponential time complexity due to overlapping subproblems.)

**2. Greedy**

This approach comes up with solutions piece by piece. It makes greedy choice at each step to ensure optimality. The main idea of Greedy approach is to make locally optimal choices in order to get globally optimal solutions. Greedy approach may not give the optimal solution.

Time complexity: O(mlogn)

**3. Dynamic**

This approach divides the problem into smaller pieces. Dynamic programming solves the recursive approach in more efficient manner as it stores the result of sub-problems for future use instead of re-computing Dynamic Programming always makes right decision as it always solves the sub-problem before making choice. Dynamic approach gives efficient solution with optimal substructure and avoids redundant calculations.

Time complexity: O(m\*n)

**COMPARISION**

* **Tables and Graphs**

**US COIN SYSTEM (GREEDY VS RECURSIVE APPROACH)**

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | GREEDY APPROACH | RECURSIVE APPROACH | DYNAMIC APPROACH |
| N=11 | 7000.053301 nanoseconds | 44598.00123 nanoseconds | 25999.7323155403 nanoseconds |
| N=23 | 7399.823517 nanoseconds | 134298.66345 nanoseconds | 39800.0702261924 nanoseconds |
| N=31 | 7799.826562 nanoseconds | 123342.88753 nanoseconds | 55699.6092200279 nanoseconds |
| N=51 | 7000.053301 nanoseconds | 754297.0017 nanoseconds | 63499.901443719 nanoseconds |
| N=73 | 6800.051779 nanoseconds | 1245899.014 nanoseconds | 81700.2728581428 nanoseconds |
| N=83 | 10699.96506 nanoseconds | 1914300.12 nanoseconds | 100700.184702873 nanoseconds |
| N=91 | 10000.07614 nanoseconds | 3679198.9932 nanoseconds | 114700.29130578 nanoseconds |
| N=99 | 9600.073099 nanoseconds | 3806989.0025 nanoseconds | 132199.842482805 nanoseconds |

**GRAPHS**

**WEIRD COIN SYSTEM (GREEDY VS RECURSIVE APPROACH)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INPUT | GREEDY APPROACH | RECURSIVE APPROACH | | DYNAMIC APPROACH |
| N= 69 | 7600.05787014961 nanoseconds | 1280999.27678704 nanoseconds |  | 112100.038677454 nanoseconds |

**ANALYSIS AND DISCUSSION**

In this project, I have tried three approaches (recursive, greedy and dynamic) for the US coin system and weird coin system. For recursive approach I analyzed that it does take more time in comparison to the greedy approach because in recursive approach, subproblems are computed again and again instead of storing them. On the other hand, Greedy Approach works fine for US Coin System but for the weird coin System it does not generate an optimal solution. Hence, Greedy is not always optimal.

The difference between greedy and recursive approaches is that the recursive approach finds the optimal solution but is very slow and requires some memory for recursive calls whereas the greedy approach does not always find the optimal solution, but it is fast and requires almost no memory. After performing these two algorithms when I used dynamic programming to solve this problem, I analyzed that dynamic programming always gives the optimal solution by taking O(m\*n) time which is less in comparison to recursive because recursive takes exponential time to perform the coin change problem. The difference between recursion and dynamic programming is that dynamic programming is a little bit intelligent version of recursion. Recursion does not store solution and output of one part is dependent on the output of the second part of subproblems whereas dynamic approach splits the problem into subproblems, stores their solutions and uses them if needed in the future. The difference between dynamic programming and greedy programming is that a dynamic approach makes the decision at every stage and is a more efficient and greedy approach makes the decision based on the first stage and is less efficient. The greedy approach first makes the best choice and then solves the subproblem whereas the dynamic approach solves all the subproblems and then selects which will find the optimal solution.

Greedy is much faster in comparison to the other two approaches but it does not always provide the optimal solution. The results from the Greedy, Recursive, and Dynamic Programming approaches for the Coin Change problem can differ in terms of the number of coins used to reach the target amount. Greedy and Recursive approaches may not always give the optimal solution, while Dynamic Programming ensures the optimal solution efficiently by avoiding redundant calculations.

**OUTPUT GENERATED**

|  |  |  |
| --- | --- | --- |
|  | **\*US COIN SYSTEM GREEDY APPROACH\*** |  |
| INPUT | GREEDY APPROACH FOR US COIN SYSTEM MINIMUM COINS | EXECUTION TIME |
| 11 | [10, 1] | 7000.053301 nanoseconds |
| 23 | [10, 10, 1, 1, 1] | 7399.823517 nanoseconds |
| 31 | [25, 5, 1] | 7799.826562 nanoseconds |
| 51 | [25, 25, 1] | 7000.053301 nanoseconds |
| 73 | [25, 25, 10, 10, 1, 1, 1] | 6800.051779 nanoseconds |
| 83 | [25, 25, 25, 5, 1, 1, 1] | 10699.96506 nanoseconds |
| 91 | [25, 25, 25, 10, 5, 1] | 10000.07614 nanoseconds |
| 99 | [25, 25, 25, 10, 10, 1, 1, 1, 1] | 9600.073099 nanoseconds |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | **\*US COIN SYSTEM RECURSIVE APPROACH\*** |  |
| INPUT | RECURSIVE APPROACH FOR US COIN SYSTEM MINIMUM COINS AND COUNT | EXECUTION TIME |
| 11 | Coins: [1, 10], Count:2 | 44598.00123 nanoseconds |
| 23 | Coins: [1, 1, 1, 10, 10], Count: 5 | 134298.66345 nanoseconds |
| 31 | Coins: [1, 5, 25], Count: 3 | 123342.88753 nanoseconds |
| 51 | Coins: [1, 25, 25], Count: 3 | 754297.0017 nanoseconds |
| 73 | Coins: [1, 1, 1, 10, 10, 25, 25], Count: 7 | 1245899.014 nanoseconds |
| 83 | Coins: [1, 1, 1, 5, 25, 25, 25], Count: 7 | 1914300.12 nanoseconds |
| 91 | Coins: [1, 5, 10, 25, 25, 25], Count: 6 | 3679198.9932 nanoseconds |
| 99 | Coins: [1, 1, 1, 1, 10, 10, 25, 25, 25], Count: 9 | 3806989.0025 nanoseconds |
|  |  |  |

**\*US COIN SYSTEM DYNAMIC APPROACH\***

|  |  |  |
| --- | --- | --- |
| INPUT | DYNAMIC APPROACH FOR US COIN SYSTEM MINIMUM COINS AND COUNT | EXECUTION TIME |
| N=11 | Coins:[10, 1] | 25999.7323155403 nanoseconds |
| N=23 | Coins:[10, 10, 1, 1, 1] | 39800.0702261924 nanoseconds |
| N=31 | Coins:[25, 5, 1] | 55699.6092200279 nanoseconds |
| N=51 | Coins:[25, 25, 1] | 63499.901443719 nanoseconds |
| N=73 | Coins:[25, 25, 10, 10, 1, 1, 1] | 81700.2728581428 nanoseconds |
| N=83 | Coins: [25, 25, 25, 5, 1, 1, 1] | 100700.184702873 nanoseconds |
| N=91 | Coins: [25, 25, 25, 10, 5, 1] | 114700.29130578 nanoseconds |
| N=99 | Coins: [25, 25, 25, 10, 10, 1, 1, 1, 1] | 132199.842482805 nanoseconds |

**\*WEIRD COIN SYSTEM \***

|  |  |  |  |
| --- | --- | --- | --- |
| DYNAMIC APPROACH FOR WEIRD COIN SYSTEM | GREEDY APPROACH FOR WEIRD COIN SYSTEM | RECURSIVE APPROACH FOR WEIRD COIN SYSTEM |  |
| [23, 23, 23] | [25, 25, 10, 5, 1, 1, 1, 1] | Coins: [1, 1, 1, 1, 5, 10, 25, 25], Count: 8 |  |
| 112100.038677454 nanoseconds | 7600.05787014961 nanoseconds | 1280999.27678704 nanoseconds |  |