---------- *Wheaton College, MA - Spring 2024* ----------

COMP 318 Algorithms - Project 3

Description - *Greedy & Dynamic Programming*

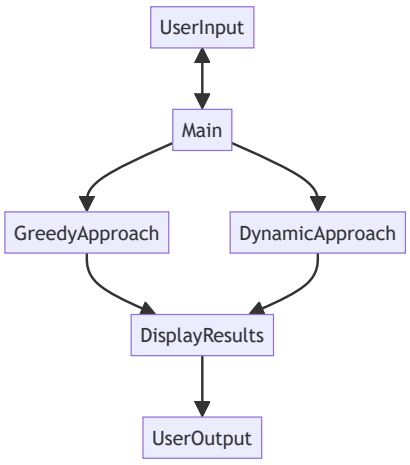
This project explores the design and implementation of algorithms to manage a Tax Amount distribution effectively, specifically focusing on the greedy and dynamic programming approaches to solve the change-making problem. The project aims to find the most efficient way to combine different denominations; to reach a specified total with the minimum number of bills, illustrating the concepts of algorithm optimization and complexity.

Highlights of the project:

- **Interactive User Input:** Allows users to enter a specific amount and observe how each algorithm tackles the change-making problem.
- **Change-Making Problem:** Calculates the minimum number of bills needed to reach a specific amount using different denominations.
- **Greedy Algorithm:** Quickly computes a solution by always taking the best immediate or local solution, ensuring rapid execution but not always guaranteeing the best global solution.
- **Dynamic Programming Algorithm:** Breaks down the problem into simpler subproblems and solves them once, storing their solutions to avoid redundant computations, ensuring an optimal solution is found.
- **Comparison and Analysis:** Evaluates and compares the efficiency of both algorithms in terms of time and space complexities.
- **Result Visualization:** Displays detailed steps of both algorithms, showing the number of each denomination used and the progress towards the total amount.

Developed by Sam Hammami '25

System Architecture Diagram:



This project contains multiple files that divide the workload.

- **main.cpp:**

This is the primary driver file for the project. It handles user input for specifying the amount of Tax Due for which invokes both the greedy and dynamic programming algorithms. It is designed to showcase how each algorithm approaches the calculation of minimum bills required to reach the specified total,

- **changeMaker.cpp:**

Implements the core logic of the greedy and dynamic programming algorithms. This file contains the functions that directly manage the calculation of minimum bills required to achieve the specified total using both methods. It is crucial for demonstrating the practical application of these algorithms in a real-world financial context.

- **changeMaker.h:**

This header file declares the functions and structures used in `changeMaker.cpp`. It includes the prototypes for the greedy and dynamic algorithms, ensuring that `main.cpp` can call these functions appropriately. The file helps in maintaining a clean and organized code structure, promoting good software development practices.

Getting Started

This program uses the following libraries:

- #include // For input-output
- #include // For Greedy Programming
- #include // For Dynamic Programming
- #include // For INT_MAX
- #include // For string manipulation
- #include // For setw

CMake Minimum version

- cmake_minimum_required(VERSION 3.27)

Compiler version

- Clang C++ compiler

Installing /compiling

On CLion

After downloading the code, please create a new project on CLion and select the folder containing the program as the project folder. It will ask you to either use the files existing inside the folder or create a blank design, please select to use the files inside the folder. The CMakeList.txt file and cmake-build-debug folder should include all the necessary files needed to run this program on CLion.

Executing program

On CLion

Click the run button - or - Shift + F10

Run the code - It should be like:

This section provides 3 three sample runs to demonstrate the behavior of both the greedy and dynamic programming algorithms under various scenarios, illustrating their efficiency in handling different tax amounts.

_ Sample 1 _

```
***     Welcome to the IR$ of Banana Republic!     ***

Please enter your name below!
My Name is Poor Patrick

What is your tax $$$ due?
My Tax Amount Due = $96003

---------------------------------------------------------------
Plan 1: The Greedy Approach
Tax Due: $96003
                Bills         Num                   Subtotal
                $95000   x    1     = $95000        $95000
                $32001   x    0     = $0            $95000
                $5001    x    0     = $0            $95000
                $701     x    1     = $701          $95701
                $101     x    2     = $202          $95903
                $1       x    100   = $100          $96003

Total number of bills needed: 1 + 0 + 0 + 1 + 2 + 100 = 104
*-----------------------------*****--------------------------*
Plan 2: The Dynamic Programming Approach
Tax Due: $96003
                Bills         Num                   Subtotal
                $95000   x    0     = $0            $0
                $32001   x    3     = $96003        $96003
                $5001    x    0     = $0            $96003
                $701     x    0     = $0            $96003
                $101     x    0     = $0            $96003
                $1       x    0     = $0            $96003

Total number of bills needed: 0 + 3 + 0 + 0 + 0 + 0 = 3
---------------------------------------------------------------
Which payment plan do you want to choose (1 or 2)?
Plan:2

---------------------------------------------------------------
Dear  Poor Patrick,

Thank you for paying your tax ON TIME, you have chosen Plan 2!
Remember that IR$ of Banana Republic is always here for you.

Best Regards,
IR$ Support Team
---------------------------------------------------------------
```

__ Sample 2 __

```
***     Welcome to the IR$ of Banana Republic!     ***

Please enter your name below!
My Name is Houssem .H

What is your tax $$$ due?
My Tax Amount Due = $35000

---------------------------------------------------------------
Plan 1: The Greedy Approach
Tax Due: $35000
                Bills         Num                   Subtotal
                $95000   x    0     = $0            $0
                $32001   x    1     = $32001        $32001
                $5001    x    0     = $0            $32001
                $701     x    4     = $2804         $34805
```

```
                $101     x    1     = $101          $34906
                $1       x    94    = $94           $35000

Total number of bills needed: 0 + 1 + 0 + 4 + 1 + 94 = 100
* ------------------------- ***** ----------------------- *
Plan 2: The Dynamic Programming Approach
Tax Due: $35000
                Bills         Num                   Subtotal
                $95000   x    0     = $0            $0
                $32001   x    1     = $32001        $32001
                $5001    x    0     = $0            $32001
                $701     x    4     = $2804         $34805
                $101     x    1     = $101          $34906
                $1       x    94    = $94           $35000

Total number of bills needed: 0 + 1 + 0 + 4 + 1 + 94 = 100
-----------------------------------------------------------------
Which payment plan do you want to choose (1 or 2)?
Plan:1

-----------------------------------------------------------------
Dear  Houssem .H,

Thank you for paying your tax ON TIME, you have chosen Plan 1!
Remember that IR$ of Banana Republic is always here for you.

Best Regards,
IR$ Support Team
-----------------------------------------------------------------
```

___ Sample 3 ___

```
***     Welcome to the IR$ of Banana Republic!     ***

Please enter your name below!
My Name is Rich Richard

What is your tax $$$ due?
My Tax Amount Due = $296005

-----------------------------------------------------------------
Plan 1: The Greedy Approach
Tax Due: $296005
                Bills         Num                   Subtotal
                $95000   x    3     = $285000       $285000
                $32001   x    0     = $0            $285000
                $5001    x    2     = $10002        $295002
                $701     x    1     = $701          $295703
                $101     x    2     = $202          $295905
                $1       x    100   = $100          $296005

Total number of bills needed: 3 + 0 + 2 + 1 + 2 + 100 = 108
* ------------------------- ***** ----------------------- *
Plan 2: The Dynamic Programming Approach
Tax Due: $296005
                Bills         Num                   Subtotal
                $95000   x    2     = $190000       $190000
                $32001   x    3     = $96003        $286003
                $5001    x    2     = $10002        $296005
                $701     x    0     = $0            $296005
                $101     x    0     = $0            $296005
                $1       x    0     = $0            $296005

Total number of bills needed: 2 + 3 + 2 + 0 + 0 + 0 = 7
-----------------------------------------------------------------
Which payment plan do you want to choose (1 or 2)?
Plan:2

-----------------------------------------------------------------
Dear  Rich Richard,
```

```
Thank you for paying your tax ON TIME, you have chosen Plan 2!
Remember that IR$ of Banana Republic is always here for you.

Best Regards,
IR$ Support Team
---------------------------------------------------------------
```

1) My interpretation:

For large amounts, the DP approach consistently outperforms Greedy (see sample 1 and 3) in terms of the number of bills, highlighting its scalability and effectiveness in handling larger and more complex scenarios.

2) My analysis:

__ **Time Efficiency** __ :
Dynamic Programming tends to be more time-efficient as the amount increases, due to its methodical reuse of computed solutions.

- ***Greedy:*** $O(n)$ The greedy algorithm processes each denomination once, determining how many times each denomination can fit into the remaining amount. Basically, here professor the iterations depend linearly on the number of denominations, the time complexity is linear.
- ***Dynamic Programming:*** $O(m \text{ x } n)$ Dynamic Programming for the change-making problem uses a two-dimensional array where m is the amount to be changed, and n is the number of different denominations. The algorithm iterates over each denomination for each value from 1 to m, leading to a time complexity that is proportional to the product of the amount and the number of denominations.

__ **Space Efficiency** __ :
Although DP requires more memory to store sub-solutions, its space cost can be justified by the significantly better outcomes it delivers.

- ***Greedy:*** $O(1)$ Greedy does not require additional space to store sub-solutions, as it only needs to keep track of the number of each denomination used to reach the total amount.
- ***Dynamic Programming:*** $O(m \text{ x } n)$
  - Worst-case scenario picks the largest denomination m or n (As we saw in the examples in class)
  - Dynamic Programming uses a two-dimensional array to store the sub-solutions, which can be space-intensive for larger amounts. The space complexity is proportional to the amount to be changed, m, as it needs to store the solutions for each value from 1 to m.

3) My conclusion:

- **Optimal Approach:** Dynamic Programming is generally better because it guarantees the optimal solution by considering all possible combinations, unlike Greedy, which can fail in complex scenarios.
- **Recommendation:**
  For smaller amounts (Sample 2):
  - The **greedy approach** excels in scenarios where simplicity and speed are paramount (e.g., small amounts), offering $O(n)$time complexity and $O(1)$ space complexity.

  => This makes it ideal for quick calculations and situations where an approximate solution is acceptable.

  For larger amounts (Sample 1 and 3):
  - The **dynamic programming approach** is preferable for high-stake, large amount scenarios where precision is crucial and the minimal use of bills is essential. But it has a higher time complexity of $O(m \text{ x } n)$ and space complexity of $O(m \text{ x } n)$

  => Despite its higher computational and memory demands, it ensures the optimal solution is found, making it the best choice.

Authors

List of authors/contributors' names and contact info:

- Sam Hammami '25 - hammami_houssem@wheatoncollege.edu

Version History

- Starter Code Uploaded - April 27, 2024
- CLion Set up - April 27, 2024
- Adjusted changeMaker.h - April 28, 2024
- Completed Greedy Approach - April 30, 2024
- Some work on Main.cpp - April 30, 2024
- Planning Dynamic Approach (Book. Chap 8) - May 01, 2024
- Discussing the project with Prof. Tony - May 02, 2024
- Developed Dynamic Approach - May 02, 2024
- Solved (DP) Approach Issue - May 03, 2024
- Testing Main.cpp (Specific Cases, Display, Delete) - May 03, 2024
- Checking-In with Professor **Tony** - May 03, 2024
- Commenting the code (Sam style) - May 04, 2024
- Final Testing - May 04, 2024
- Uploaded_Final Version - May 04, 2024

License

Acknowledgments

- We acknowledge all the programming input Prof. **Martin Gagné** put into this program.
- We also acknowledge all modifications and updates from Prof. **Tony Tong**.
- Last, we acknowledge **Sam Hammami** for his hard work and dedication to this project.