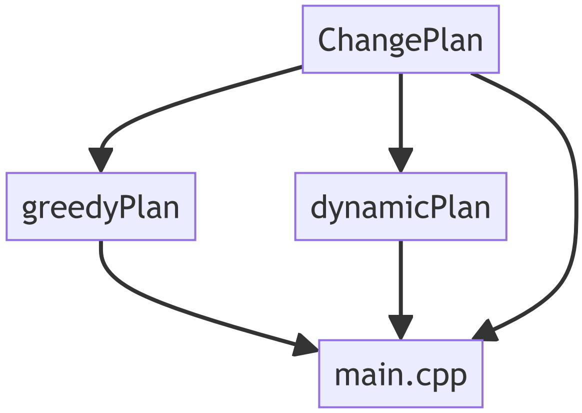
Wheaton College MA, Spring 2024, COMP 318 Algorithms - Project 3

This project uses greedy algorithms and dynamic programming algorithms to find the least amount of changes for a given amount of money. The greedy algorithm tries to separate the money into given bills of each denomination by dividing it starting from the largest to the smallest as much as possible. Dynamic programming uses a recurrent bottom-up approach to solve this problem. We can observe that dynamic programming is always the better option when it comes to finding out the least number of bills. However, dynamic programming also has a higher time complexity and much higher space complexity.

Description

This program is used to find the least amount of changes using a set of denominations. Ozgur Tuna Ozturk '24, implemented this program using a vector data structure and a custom class object that holds the denominations.

Inheritance diagram:



This project contains multiple files that divide the workload.

changemaker.cpp:
This is the main file that contains the implementation of greedy and dynamic programming algorithms. It includes function implementations of greedyPlan and dynamic Plan functions.

changemaker.h:
This is the header file for the changeMaker.cpp. It includes the function definitions for greedyPlan and dynamicPlan functions. It also includes the class declaration of the ChangePlan class.

main.cpp:
This is the file used for implementing and testing out the different algorithms. This file creates a nice user interface wrapper around the algorithms and displays the results.

test.cpp
This file uses Google Tests to do a unit test on the algorithms making sure that all of them work as expected. All the outputs are double checked with different online sources to ensure the quality of the code.

Getting Started

Dependencies

This program uses the following libraries:

- #include <iostream>
- #include <vector>
- #include <string>
- #include <iomanip>
- #include <ctime>
- #include <gtest/gtest.h> (This is for Google Unit Tests)
  To download Google Unit Test library you can run on MacOS terminal

```
% brew install googletest
```

or the command below for linux terminal:

```
$ sudo apt-get install libgtest-dev
```

or the command below for windows command line:

```
git clone https://github.com/microsoft/vcpkg.git
cd vcpkg
bootstrap-vcpkg.bat
vcpkg install gtest
vcpkg integrate install
```

Compiler version

- g++ c++14 -Wall -lgtest -lgtest_main -pthread

Installing /compiling

This project includes a Makefile that makes compiling the codes much easier. In your Terminal or command line Navigate into the directory that contains the repository and run the "make" command. This will create some files that end with ".o" extension. They are the compiled versions of the code files. The executable program is named "program".

Executing program

On UNIX Terminal

```
$ ./changeMaker
```

After you run the code it should be like this

```
Welcome to the IR$ of Banana Republic!
Please enter your name:         Ozgur Tuna Ozturk
What is your tax $$$ due?       $96003

Plan 1: The Greedy Approach
Tax Due:  96003
Greedy Change Plan:
             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 Approach
Dynamic Change Plan:
             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 (Plan 1 or Plan 2)?
2
Current Date is: 5/5/2024 You are late!!
Dear Ozgur Tuna Ozturk, please try to pay your tax ON TIME next time!
---------------------------------------------------------------------------
---------------------
```

Running Unit Tests

The unit tests are stored in test.cpp file. To compile it you can use the makefile too. To compile them use the command below:

```
$ make test
```

To run the test you can use the command below:

```
$ ./test
```

This should output something like this:

```
[==========] Running 2 tests from 2 test suites.
[----------] Global test environment set-up.
[----------] 1 test from GreedyPlanTest
[ RUN      ] GreedyPlanTest.TestGreedyPlan
TestGreedyPlan took 4 microseconds for $123456.
TestGreedyPlan took 1 microseconds for $5000.
TestGreedyPlan took 1 microseconds for $10000.
TestGreedyPlan took 1 microseconds for $15000.
TestGreedyPlan took 1 microseconds for $20000.
TestGreedyPlan took 1 microseconds for $25000.
TestGreedyPlan took 1 microseconds for $96003.
[       OK ] GreedyPlanTest.TestGreedyPlan (0 ms)
[----------] 1 test from GreedyPlanTest (0 ms total)

[----------] 1 test from DynamicPlanTest
[ RUN      ] DynamicPlanTest.TestDynamicPlan
TestDynamicPlan took 94606 microseconds for $123456.
TestDynamicPlan took 4346 microseconds for $5000.
TestDynamicPlan took 7247 microseconds for $10000.
TestDynamicPlan took 11200 microseconds for $15000.
TestDynamicPlan took 13351 microseconds for $20000.
TestDynamicPlan took 17177 microseconds for $25000.
TestDynamicPlan took 66501 microseconds for $96003.
[       OK ] DynamicPlanTest.TestDynamicPlan (214 ms)
[----------] 1 test from DynamicPlanTest (214 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 2 test suites ran. (214 ms total)
[  PASSED  ] 2 tests.
```

Authors

List of authors/contributors' names and contact info

- Ozgur Tuna Ozturk '24 - ozturk_ozgur@wheatoncollege.edu Last edited on April 5th, 2024. On this project, I hereby declare that I have fully abided by the Wheaton College MA Honor Code.

This repo can be found at: https://github.com/Tozturk18/COMP318_Proj3.git