Home(/)

# 0x08. Making Change

Algorithm Python

My Planning(/planning/me)

Weight: 1

Projects(/projects/current)

Project over - took place from Aug 19, 2024 6:00 AM to Aug 23, 2024 6:00 AM

An auto QA review will be launched at the deadline

QA Reviews I can make(/corrections/to_review)

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

## In a nutshell…

- **Auto QA review:** 16.0/16 mandatory
- **Altogether: 100.0%**
  - Mandatory: 100.0%
  - Optional: no optional tasks

Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Set vars(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

For the "0. Change comes from within" project, you will tackle a classic problem from the domain of dynamic programming and greedy algorithms: the coin change problem. The objective is to find the minimum number of coins required to make up a given total amount, given a list of coin denominations. This project challenges you to apply your understanding of algorithms to devise a solution that is not only correct but also efficient. Below are the key concepts and resources necessary to complete this project successfully.

## Concepts Needed:

1. **Greedy Algorithms**:
   - Understanding how greedy algorithms work and why they are suitable for the coin change problem.
   - Recognizing the limitations of greedy algorithms and scenarios where they might not provide the optimal solution.
2. **Dynamic Programming**:
   - Basic principles of dynamic programming as a method to solve optimization problems.
   - The concept of overlapping subproblems and optimal substructure in the context of the coin change problem.
3. **Algorithmic Complexity**:
   - Analyzing the time and space complexity of algorithms.
   - Striving for solutions with lower complexity to meet runtime constraints.
4. **Problem-Solving Strategies**:
   - Breaking down the problem into smaller, manageable sub-problems.
   - Iterative vs recursive approaches to dynamic programming.

5. **Python Programming**:
    - Manipulating lists and using list comprehensions.
    - Implementing functions with efficient looping and conditional statements.

## Resources:

- **Python Official Documentation**:
    - More Control Flow Tools (for loops, if statements) (/rltoken/oVyaCk8erLwLPj96P-qlCw)
- **GeeksforGeeks Articles**:
    - Coin Change | DP-7 (/rltoken/iQPaO5JhI-BtuZdm6HIVCQ)
    - Greedy Algorithm to find Minimum number of Coins (/rltoken/FsBN0oeRp0FpyU8sMd4UiA)
- **YouTube Tutorials**:
    - Dynamic Programming - Coin Change Problem (/rltoken/qFEdwwtAVyJr9NLHDZDsUQ) for a visual and step-by-step explanation of the dynamic programming approach.

By thoroughly understanding these concepts and utilizing the provided resources, you will be well-prepared to tackle the coin change problem. You will need to decide whether a greedy algorithm suffices for your particular set of coin denominations or if a more comprehensive dynamic programming approach is necessary to ensure correctness and efficiency. This project not only tests algorithmic skills but also reinforces the importance of choosing the right strategy based on problem constraints.

# Additional Resources

- Mock Technical Interview (/rltoken/ktLaKIVRkq_-byFO-_-aGg)

# Requirements

## General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using `python3` (version 3.4.3)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `PEP 8` style (version 1.7.x)
- All your files must be executable

# Tasks

## 0. Change comes from within

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Given a pile of coins of different values, determine the fewest number of coins needed to meet a given amount `total`.

- Prototype: `def makeChange(coins, total)`
- Return: fewest number of coins needed to meet `total`
  - If `total` is `0` or less, return `0`
  - If `total` cannot be met by any number of coins you have, return `-1`
- `coins` is a list of the values of the coins in your possession
- The value of a coin will always be an integer greater than `0`
- You can assume you have an infinite number of each denomination of coin in the list
- Your solution's runtime will be evaluated in this task

```
carrie@ubuntu:~/0x08-making_change$ cat 0-main.py
#!/usr/bin/python3
"""
Main file for testing
"""

makeChange = __import__('0-making_change').makeChange

print(makeChange([1, 2, 25], 37))

print(makeChange([1256, 54, 48, 16, 102], 1453))

carrie@ubuntu:~/0x08-making_change$
```

```
carrie@ubuntu:~/0x08-making_change$ ./0-main.py
7
-1
carrie@ubuntu:~/0x08-making_change$
```

**Repo:**

- GitHub repository: `alx-interview`
- Directory: `0x08-making_change/`
- File: `0-making_change.py`

Check submission | View results