Project 2: Coin change problem

dr Szymon Murawski

Task 1: *Coin change for canonical coin systems*

1. Using greedy approach write an algorithm, that for a given input integer X returns a string of how many coins of specific value should be returned.
2. Pseudocode for this algorithm was given during lecture, use it!
3. Assume that we have at our disposal set of coins with values: $C = \langle 1, 2, 5, 10, 20, 50, 100 \rangle$
4. For each input algorithm should return the optimal number of coins
5. Example test cases:
   - If x is not a positive integer, throw exception *Input should be a positive integer*
   - For $x = 0$, return empty string
   - For $x = 3$, return $1 \times 2, 1 \times 1$
   - For $x = 47$, return $2 \times 20, 1 \times 5, 1 \times 2$

Task 2: *Greedy algorithm for general case*

1. Expand previous algorithm, allowing it to also take as input set of coins C
2. For each input set of coins C and change to be made x algorithm should return string of how that change should be made
3. Example test cases:
   (a) If x is not a positive integer, throw exception *Input should be a positive integer*
   (b) If any coin c_x is not a positive integer, throw exception *Coin should be positive integer*
   (c) If there are coins c_i, c_j such that $c_i = c_j$, throw exception *Coins should have unique values*
   (d) If $1 \notin C$, throw exception *One coin must have value of 1*
   (e) For $C = \langle 1, 3, 4 \rangle$, $x = 6$, return $1 \times 4, 2 \times 1$
   (f) For $C = \langle 1, 3, 6, 8 \rangle$, $x = 12$, return $1 \times 8, 1 \times 3, 1 \times 1$

Task 3: *Coin change for general case*

Previous algorithm will produce sub-optimal results for non-canonical coin systems. For example given coins 1,3,4 and input x=6 greedy approach would produce $1 \times 4, 2 \times 1$, while the optimal answer is 2×3. Optimal answer in general case can be found using dynamic programming

1. Using dynamic programming write an algorithm, that for a given set of coins $C = \langle c_1, c_2, c_3, \ldots, c_n \rangle$ and positive integer X returns a string of how many coins of specific value should be returned.

2. Pseudocode for this algorithm was given during lecture, use it!
3. Set of coins C should be taken as an input of a program.
4. For each input algorithm should return the optimal number of coins.
5. Example test cases:
    - If x is not a positive integer, throw exception *Input should be a positive integer*
    - If any coin c_x is not a positive integer, throw exception *Coin should be positive integer*
    - If there are coins c_i, c_j such that $c_i = c_j$, throw exception *Coins should have unique values*
    - If $1 \notin C$, throw exception *One coin must have value of 1*
    - For $C = \langle 1, 3, 4 \rangle$, $x = 6$, return 2×3
    - For $C = \langle 1, 3, 6, 8 \rangle$, $x = 12$, return 2×6

Task 4: *Analysis*

Now that we have two algorithms we can check how they perform for different number of coins and different value of change to be made.

1. Run your programs for increasing number of coins in set C and measure the time it takes for both greedy and dynamic algorithm to complete the task
2. Run your program for increasing value of change to be made x and measure the time it takes for both greedy and dynamic algorithm to complete the task
3. Plot your results and write short report on your findings - how do those two algorithms behave when you increase size of coins array of change to be made?