Coin Changes Report

Kexin Ding

1. Source Code

Please check coin_changes.py

2.Excution Results

```
[Test 1]
coin changes for:  5
[Dynamic Programming] the change for the fewest number of coin is: 1
coins:  [5]
[Greedy algorithm] the change for the fewest number of coin is: 1
coins:  [5]

[Test 2]
coin changes for:  30
[Dynamic Programming] the change for the fewest number of coin is: 2
coins:  [5, 25]
[Greedy algorithm] the change for the fewest number of coin is: 2
coins:  [5, 25]

[Test 3]
coin changes for:  82
[Dynamic Programming] the change for the fewest number of coin is: 6
coins:  [1, 1, 5, 25, 25, 25]
[Greedy algorithm] the change for the fewest number of coin is: 6
coins:  [1, 1, 5, 25, 25, 25]

[Test 4]
coin changes for:  365
[Dynamic Programming] the change for the fewest number of coin is: 16
coins:  [5, 10, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25]
[Greedy algorithm] the change for the fewest number of coin is: 16
coins:  [5, 10, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25]
```

3. Time Complexity Analysis

For dynamic programming version, I use the method as the Bottom to Up method. Time complexity: T(n) = O(4n) + O(n) = O(4n) = O(n). 4 is the number of coin types: penny, nickel, dime, quarter.

For the greedy algorithm version, time complexity is T(n) = O(n). Because there is only one for loop.

4. Answer of c

a. We assume that we only use the first two denominations $c^0 = 1$ and $c^1 = c$. At most, we can use (c-1) $*c^0$ because any value larger than c would be replaced by at least on c. This will reduce the total coin number by c-1. It means the remainder is greater than c and we can use only c^0 and c^1. To satisfy the condition of fewer changes, we can use as many c^1 as we can before use c^0.

b. We assume that we only use the first two denominations $c^{n-1} = 1$ and c^n. At most, we can use (c-1) $*c^{n-1}$ because any value larger than c^n would be replaced by at least on c^n. This will reduce the total coin number by c-1. It means the remainder is greater than c^n and we can use only c^{n-1} and c^n. To satisfy the condition of fewer changes, we can use as many c^nas we can before use c^{n-1}.

c. Consider a and b together, the greedy algorithm applies to all situations of this coin set. Hence greedy algorithm is optimal.