Coin Changing Problem

q:) Describe a greedy algorithm to make change consisting of quarters and dimes, nickels and pennies. Prove that your algorithm yields an optimal solution.

→ We define $C[j]$ to be the minimum number of coins. Need to make change for j cents. Let the coin denominations be $d_1, d_2 \ldots\ldots d_k$. Since one of coins is a penny, there is a way to make change for any amount $j \geq 1$.

Because of optimal substructure, if we knew that an optimal solution for the problem of making change for j cents used a coin of denomination d_i, we would have

$$C[j] = 1 + C[j - d_i]$$

As base cases, we have that $C[j] = 0$ for all $j \leq 0$

To develop a recursive formulation, we check all denominations giving

$$C[j] = \begin{cases} 0 & \text{if } j \leq 0 \\ 1 + \min\limits_{1 \leq i \leq k} \{C[j - d_i]\} & \text{if } j > 1 \end{cases}$$

we can compute the c[j] values in order of increasing j by using a table. The following procedure does so, producing a table c[1.....n]. It avoids even examining C[j] for $j \le 0$ by ensuring that $j \ge d_i$ before looking up $c[j-d_i]$. The procedure also produces a table denom[1......n] where denom[j] is denomination of a coin used in an optimal solution to the problem of making change for j cents.

```
COMPUTE CHANGE (n,d,k)
let c[1.....n] and denom[1....n] be new arrays
for j=1 to: n
  c[j] = ∞
  for i=1 to k
    if j ≥ d[i] and 1 + c[j-d[i]) < c[j]
      c[j] = 1+ c[j - d[i]]
      denom[j] = d[i]
return c and denom.
```

Time Complexity : O(nK)

b.) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels and pennies. Prove that this algorithm yields an optimal solution.

→ Suppose we have an optimal solution for a problem of making change for n cents and we know that this optimal solution uses a coin whose value is c cents, let this optimal solution uses k coins. We claim that this optimal solution for problem of n cents must contain within it, an optimal solution for the problem of n-c cents. Clearly there are K-1 coins in the solution to n-c cents problem used within our optimal solution to the n cents problem. If we had a solution to n-c cents problem that used fewer than K-1 coins, then we could use this solution to produce a solution to the n cents problem that uses fewer than K coins, which contradicts the optimality of our solution

A greedy algorithm to make change using quarters dimes, nickels and pennies works as follow:

1. Give $q = \lfloor n/25 \rfloor$ quarters. That leaves $n_q = n \bmod 25$ cents to make change.

2. Then give $d = \lfloor n_q / 10 \rfloor$ dimes. That leaves $n_d = n_q \bmod 10$ cents to make change.

3. Then give $k = \lfloor n_d / 5 \rfloor$ nickels. That leaves $n_k = n_d \bmod 5$ cents to make change.

4. Finally give $p = n_k$ pennies.

The problem we wish to solve is making change for ~~no dimes~~ n cents. If $n=0$, the optimal solution is to give no coins. If $n>0$, determine the largest coin whose value is less than or equal to n. Let this coin have value C. Give one such coin and then recursively solve the subproblem of making change for $n-c$ cents.

To prove that this algorithm yields an optimal solution, we first need to show that the greedy choice property holds, that is, that some optimal solution to making change for n cents includes one coin of value c, where C is largest coin value such that $c \leq n$. Consider some optimal solution. If this optimal solution includes a coin of value c, then we are done. Otherwise, this optimal solution does not includes a coin of value c, then

We have four cases consider.

1. If $1 \leq n < 5$, then $c = 1$, A solution may consist only of pennies, and so it must contain the greedy choice

2. If $5 \leq n < 10$, the $c = 5$. By supposition, this optimal solution doesnot contain a nickel, and so it contains only pennies. Replace five pennies by one nickel to give the solution with 4 fewer coins.

3. If $10 \leq n < 25$, then $c = 10$. By supposition, this optimal solution doesnot contain a dime and, so it contains only nickels and pennies. Some subset of nickels & pennies in this solution adds up to 10 cents, so we can replace these nickels & pennies in this solution adds up to 10 cents, and so we can replace these nickels and pennies by a dime to give a solution with fewer coins.

4. If $25 \leq n$, then $c = 25$. By supposition, this optimal solution doesnot contain a quarter, and so it contains only dimes, nickels and pennies. If it contains 3 dimes, we can replace these 3 dimes by a quarter and a nickel, giving a solution with few coins.

Thus, we have shown that there is always an optimal solution that includes greedy choice, and that we can combine the greedy choice with an optimal solution to the remaining subproblem to produce an optimal solution to the original problem. Therefore, greedy algorithm produces optimal solution.

Time Complexity = $O(n)$

c. Suppose that available coins are in denominations that are powers of c, ie denominations are $c^0, c^1 \ldots c^k$ for some integers $c > 1$ and $k >= 0$. Show that the greedy algorithm always yields optimal solution.

⇒ let us assume that c^k is the highest denomination, then we compare the input with this value, if the input is greater then, we check the input $- c^k$ with the next highest denomination c^{k-1}. This is again best approach as the large values make a smaller number of coins - we proceed in this way until we reach c^0. Since greedy algorithm chooses the highest value first, it always gives us an optimal solution for this problem.