ADSA Assignment 6 Report

S20190010029

Problem 1: Coin Denomination

The approach is to use Dynammic Programming to find minimum number of coins. While finding the minimum number of coins, Backtracking can be used to track the coins needed to make thei sum equals to Value. Follow the steps to solve the problem:

1. Initialize the auxiliary array DP[]. This array will store the minimum number os coins needed to make sum equals to i.
2. Find the minimum number of coins needed to make their sum equal to Value using the CountMin() procedure.
3. After Finding the minimum number of coins, We use the Backtracking Technique (Done in Denomination() procedure) to track down the coins used, to make the sum equal Value.
4. In this Backtracking technique, traverse the array and choose a coin which is smaller than the current sum such that DP[Current_Sum] equals to DP[CurrentSum-ChoosenCoin]+1 (This is Optimal Substructure). Store the chosen coin in an array.
5. After completing the above step, Backtrack again by passing the CurrentSum as (CurrentSum - ChosenCoin).
6. After finding the solution, print the array of chosen coins.

Time Complexity: O(N*Value), Where N is the length of the given Denominations and Value is the total sum to add up to.

Auxiliary Space: O(N) [Used for the array to store Denominations that add up to the value].

Problem 2: Travel and Stay Planner

Subproblem Definition:
Let Penalty[i] be the minimum total penality to get to Hotel i.

Recursive Formulation:
To get Penalty[i], We consider all possible Hotels j we can stay at night before reaching Hotel i. For each of these possibilities, the minimum penalty to reach i is the sum of:

- The minimum penalty Penalty[j] to reach j.
- And the cost $(200 - (Hotels[j] - Hotels[i]))^2$ of a one-day trip from Hotel j to Hotel i.

Because we are interested in the minimum penalty to reach i, We take the minimum of these values over all the j:

$$Penalty[i] = \min(Penalty[j]+(200 -(Hotels[j]-Hotels[i]))^2)$$

And the base case is Penalty[0] = 0.

Pseudo-Code:

```
// Base Case
// n = Total Hotels
Penalty[0] = 0
//Main Loop
for i = 1...(n-1):
    Penalty[i] = min(Penalty[j] +(200 - ( Hotels[j] - Hotels[i] ) )^2 for j=0...i-1 )
// Final Result
return Penalty[n]
```

Time Complexity:
The time complexity would be $O(n^2)$, Where n is the Total number of Hotels.

Space Complexity:
The auxiliary space would be O(n) required for storing Penalty.