Greedy Algorithm to find Minimum number of Coins

Given a value V, if we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change?

Examples:

```
Input: V = 70
Output: 2
We need a 50 Rs note and a 20 Rs note.

Input: V = 121
Output: 3
We need a 100 Rs note, a 20 Rs note and a
1 Rs coin.
```

We strongly recommend you to minimize your browser and try this yourself first.
The idea is simple Greedy Algorithm. Start from largest possible denomination and keep adding denominations while remaining value is greater than 0. Below is complete algorithm.

```
1) Initialize result as empty.
2) find the largest denomination that is
   smaller than V.
3) Add found denomination to result. Subtract
   value of found denomination from V.
4) If V becomes 0, then print result.
   Else repeat steps 2 and 3 for new value of V
```

Below is C++ implementation of above algorithm.

```
// C++ program to find minimum number of denominations#include <bits/stdc++.h>
using namespace std;
// All denominations of Indian Currency
int deno[] = {1, 2, 5, 10, 20, 50, 100, 500, 1000};
int n = sizeof(deno)/sizeof(deno[0]);
// Driver program
void findMin(int V)
{
    // Initialize result
    vector<int> ans;
    // Traverse through all denomination
    for (int i=n-1; i>=0; i--)
    {
        // Find denominations
        while (V >= deno[i])
        {
           V -= deno[i];
           ans.push_back(deno[i]);
        }
    }
    // Print result
    for (int i = 0; i < ans.size(); i++)
           cout << ans[i] << "  ";
}// Driver program
int main()
{
   int n = 93;
   cout << "Following is minimal number of change for " << n << " is ";
   findMin(n);
   return 0;
}
```

Output:

```
Following is minimal number of change for 93 is 50  20  20  2  1
```

Note that above approach may not work for all denominations. For example, it doesn't work for denominations {9, 6, 5, 1} and V = 11. The above approach would print 9, 1 and 1. But we can use 2 denominations 5 and 6.
For general input, we use below dynamic programming approach.

Thanks to Utkarsh for providing above solution here.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

See GATE Corner for all information about GATE CS and Quiz Corner for all Quizzes on GeeksQuiz.