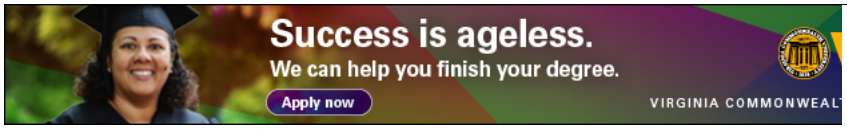

Question: STUDY PROBLEM 5 In coin change problem, the goal is findin...

(1 bookmark)

STUDY PROBLEM 5

In coin change problem, the goal is finding the smallest number of coins that will sum up a change. While there is a greedy algorithm (i.e., continuously selecting the largest coin less than remaining change) for the coins in U.S., there could be a set of coin values in another country for which the greedy algorithm does not always give the smallest number of coins. Give a set of such coins first (and explain why greedy does not work) and define a dynamic programming solution for it?

Show transcribed image text

Expert Answer

anonymous answered this
1,347 answers

Was this answer helpful? 0 0

Set of coins where the greedy algorithm fails:

Consider there are coins of 1, 3, and 4

Now, you need to get the sum 6.

By greedy algorithm gives 3 coins needed i.e. 4+ 1 + 1

But we can get the sum with only two coins of 3. 3+3= 6

Let we want to get the sum N with possible coin denominations in coins list with size M.

Let dp[i] denotes the minimum no. of sum of to a change of i.

Note that ∞ indicates that sum not possible.

I am providing the algorithm in pseudocode. It can be converted to many programming languages easily

```
Algorithm_minimum_coins(N, coins, M){
    // initializing dp array to ∞
    for i= 1 to N
        dp[i]=∞
    // Base case: requires only one coin for the change that equals to some coin
    for i = 1 to M
        dp[coins[i]]= 1
    // filling the remaining values
    for i= 1 to N
    {
        for j= 1 to M
        {
            // checking if coin is not greater than coin
            if(i>coins[j])
```

```
                    }
                }
            }
            // returning the minimum no. of coins required to get given change N
            returning dp[N]
    }
```

Comment

Questions viewed by other students

Q: Andy is trying to put together a holiday gift knapsack (with W=8) for Sarah. He has n items to choose from, each with infinitely many copies (aka. knapsack with repetitions). Item i has weight wi, and value vi. Andy wants to pick some items (possibility with duplicates) so their total weight is exactly W, while minimizing the total value of the items picked. If OPT[w] denotes the...

A: See answer

Q: Part 1: The bottom-up dynamic programming approach fo the Unbounded 0/1 Knapsack Problem given in class (Lecture 23, slide 10) is: int UnboundedBinaryKnapsack(int weights[1,...,n], float values[1,...,n], int W, int n) { z = new int[0,...,W] z[0] = 0 for (w=1; w<=W; w++) { z[w]=0 for (i=1; i<=n; i++) if (weights[i] <= w) { z[w] = max(z[w], z[w-weights[i]] + values[i...

A: See answer 100% (2 ratings)

Show more