August 18, 2015

Launchpad

Recursion++

Aman Bahl

CODING BLOCKS

Let learn how to

think

recursively

What to look for:

Given any problem look for following:

- Is there a simple case which can be done by inspection?
- Can it be broken into similar but smaller/ simpler subproblems?

Example: coin change problem

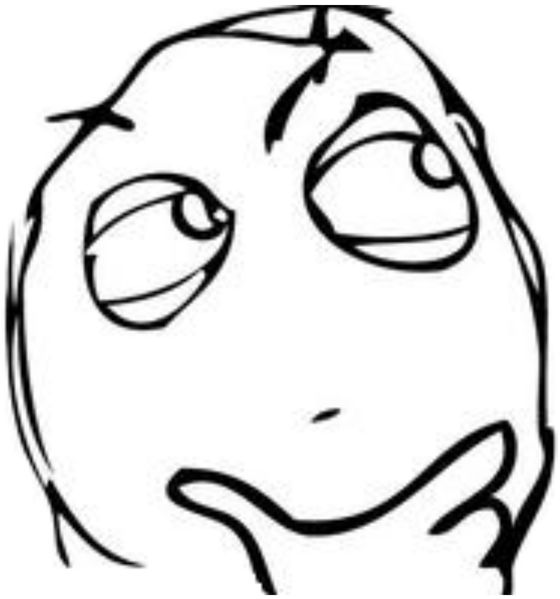
Problem: Given a value SUM and infinite supply of N types of coins. What is the minimum number of coins required to make up the given SUM. Output -1 if it is not possible.

Example:

SUM : 6

Coins : 1,2,3

Is there a simple case which can be done by inspection?



Is there a simple case which can be done by inspection?

- What if I don't have any coin?
  - Answer would be -1.
- What if sum is negative?
  - Answer would be -1.
- What if sum is zero
  - Answer would be zero

CODING BLOCKS

Can it be broken into similar but smaller subproblems?

Can it be broken into similar but smaller subproblems?

1. Give your tool/function a name.

   find_min_coins()

2. write what it need

   *find_min_coins(**all coin types**, **sum**)*

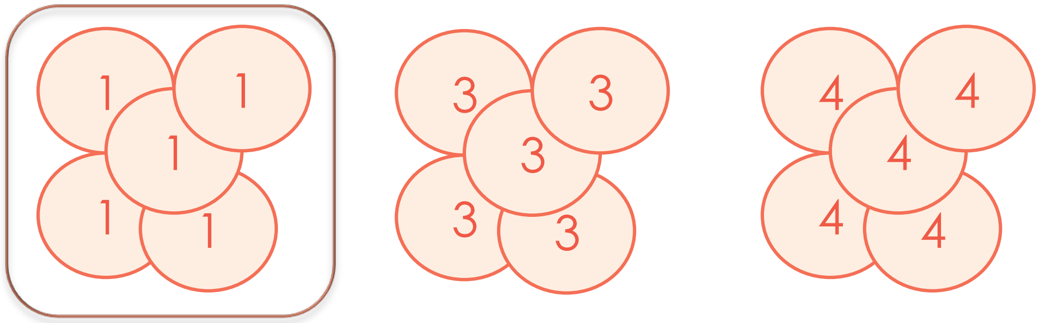
3. Write in comments above what it would return.

   // given various types of coins and a number this
   // tool will compute minimum number of coins
   // needed to make that number
   *min_coins find_min_coins(**all coin types**, **sum**)*

Literally write these three steps(in notebook/comments) until you get comfortable in recursion, these steps will reinforce something you need for next step. ☺

CODING BLOCKS

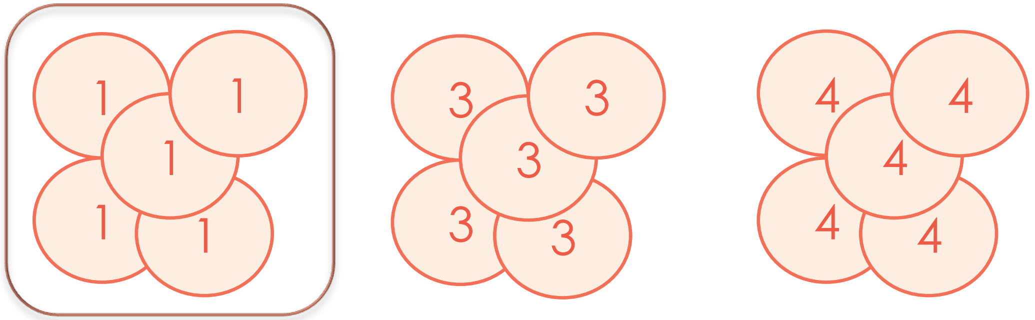
Can it be broken into similar but smaller subproblems?



Lets just look at the first type of coin we have, think the **basics**

Either we will use the coins of this type OR we will skip this type.

Can it be broken into similar but smaller subproblems?



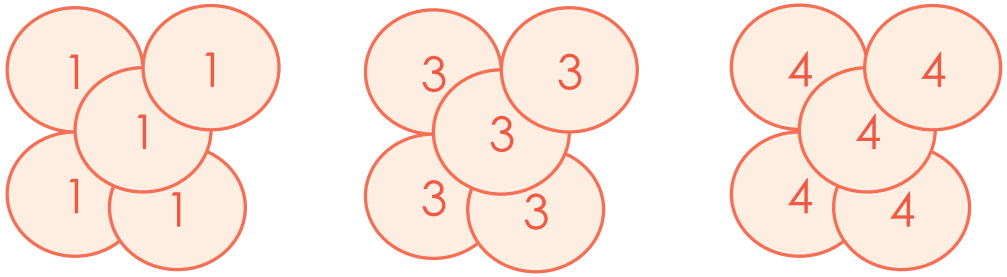
Option 1: Lets choose coin of first type to include in our solution.

We have picked one coin, so sum would be reduced by value of that coin.
SUM = SUM – V_1 *(6 - 1 = 5)*
Lets look at the what problem we have left with:
*given all types of coins and a number **(SUM – V_1)** compute minimum number of coins needed to make that number. (sounds familiar ☺)*

*Is this similar – **YES***
*Is this smaller/simpler – **YES**(the SUM needed is less now)*
*find_min_coins(all coin types, **sum-V_1**)*

CODING BLOCKS

Can it be broken into similar but smaller subproblems?



Option 2: Lets not pick coin of first type.

We have not picked any coin, so sum would be same.
Lets look at the what problem we have left with:
*given **fewer** types of coins and a number compute minimum number of coins needed to make that number. (sounds familiar ☺)*

*Is this similar – **YES***
*Is this smaller/simpler – **YES**(the types of coins are lesser now)*

find_min_coins(all coin types – first coin type, sum)

Hence solution would be:

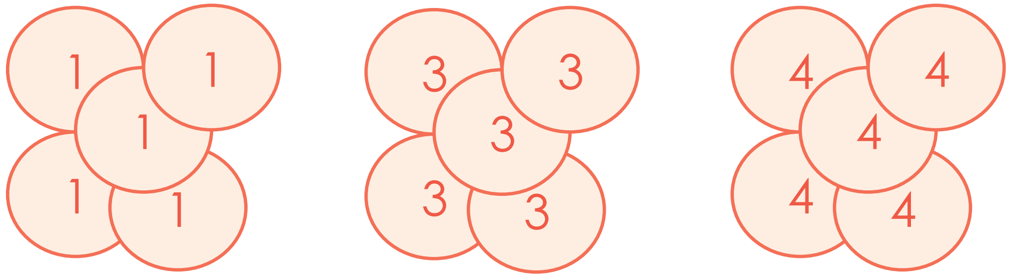
```
on_including = find_min_coins(all_coins, SUM-$V_1$)
on_excluding = find_min_coins(all_coins – first coin, SUM)

If on_including = -1:
        return on_excluding
If on_excluding = -1:
        return on_including + 1
return min(on_including+1, on_excluding)
```

Can it be broken into similar but smaller subproblems? -- another approach

CODING BLOCKS

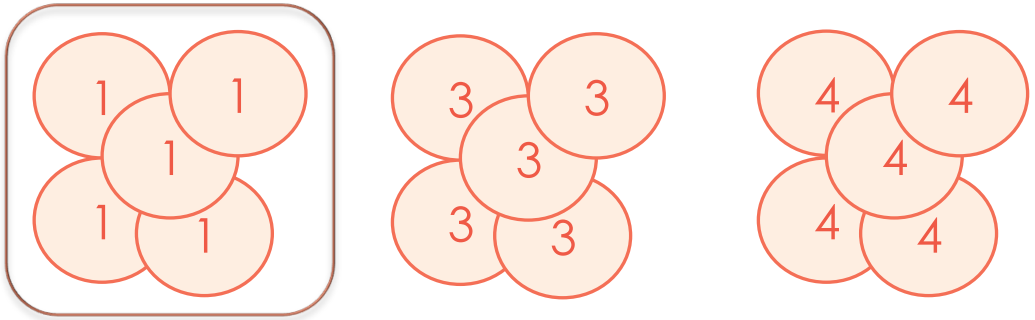
Can it be broken into similar but smaller subproblems?



Lets just try to find what can we choose for the first coin

Either we will choose first coin or second coin or third coin or …..

Can it be broken into similar but smaller subproblems?



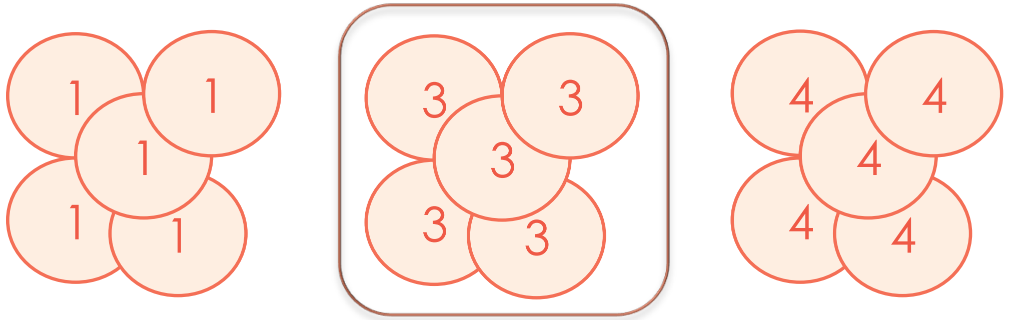
Option 2: Lets choose the coin of first type

We have picked one coin, so sum would be reduced by value of that coin.
SUM = SUM – V_1 *(6 - 1 = 5)*
Lets look at the what problem we have left with:
*given all types of coins and a number **(SUM – V_1)** compute minimum number of coins needed to make that number. (sounds familiar ☺)*

*Is this similar – **YES***
*Is this smaller/simpler – **YES**(the SUM needed is less now)*
*find_min_coins(all coin types, **sum-V_1**)*

CODING BLOCKS

Can it be broken into similar but smaller subproblems?



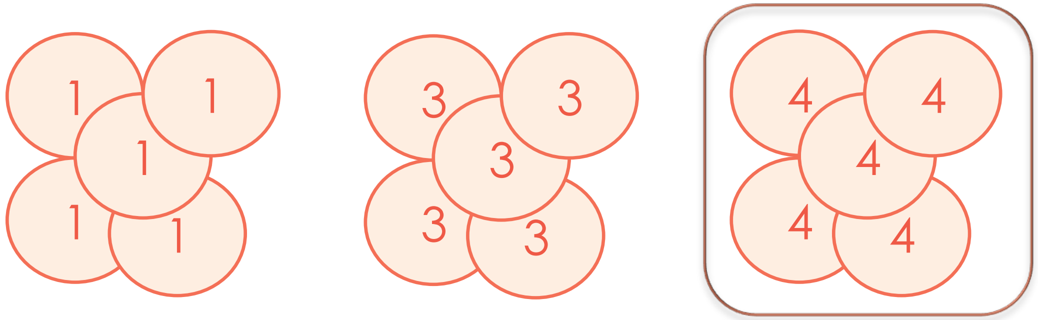
Option 1: Lets choose the coin of second type

We have picked one coin, so sum would be reduced by value of that coin.
SUM = SUM – V_2 *(6 - 3 = 3)*
Lets look at the what problem we have left with:
given all types of coins and a number ***(SUM – V_2)*** *compute minimum number of coins needed to make that number. (sounds familiar ☺)*

Is this similar – ***YES***
Is this smaller/simpler – ***YES****(the SUM needed is less now)*
find_min_coins(all coin types, ***sum-V_2****)*

CODING BLOCKS

Can it be broken into similar but smaller subproblems?



Option 3: Lets choose the coin of third type

We have picked one coin, so sum would be reduced by value of that coin.

SUM = SUM – V_3 *(6 - 4 = 2)*

Lets look at the what problem we have left with:

given all types of coins and a number ***(SUM – V_3)*** *compute minimum number of coins needed to make that number. (sounds familiar ☺)*

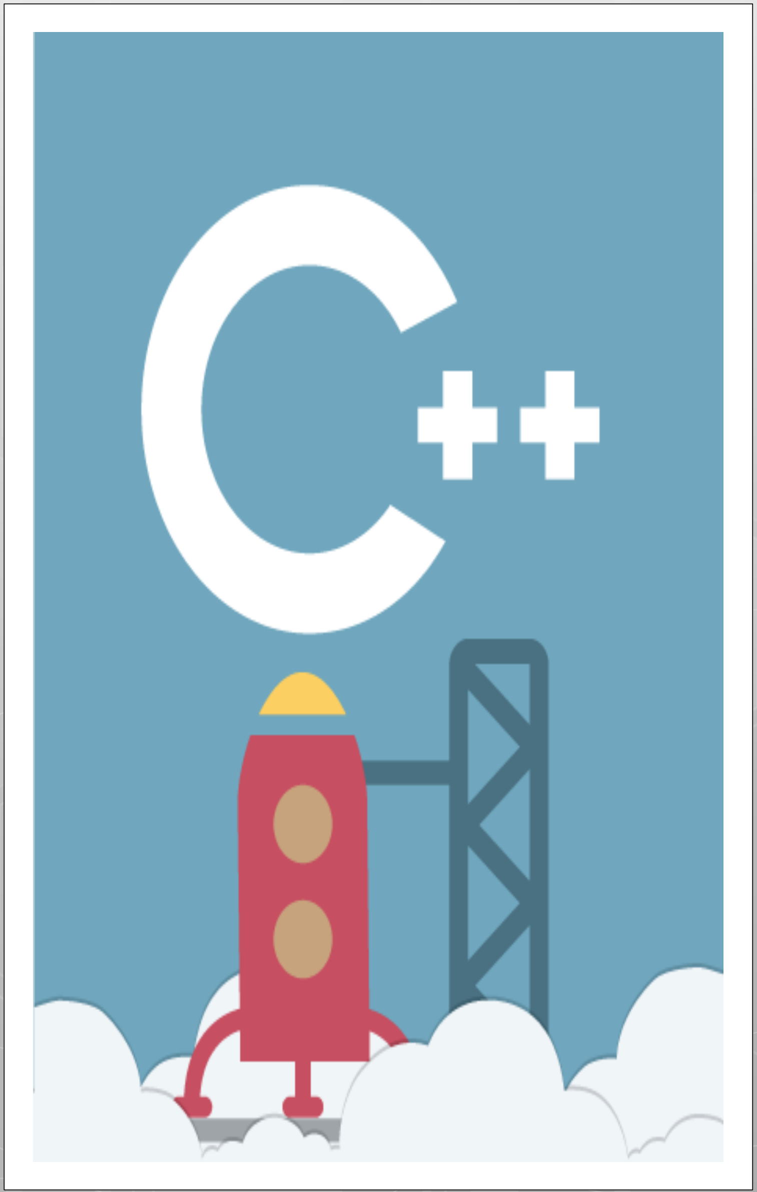
Is this similar – ***YES***

Is this smaller/simpler – ***YES****(the SUM needed is less now)*

find_min_coins(all coin types, ***sum-V_3****)*

Hence solution would be:

```
min_coins = infinity
For (each coin type i):
        on_inc = find_min_coins(all_coins, SUM-$V_i$) + 1
        min_coins  = min(min_coins, on_inc)
```



Thank You!

Aman Bahl

aman.or.b@gmail.com
+91-9908124628