**SPOILER: Find Answers to all your doubts here. Ask if you still have questions.**

discussion posted a day ago by [**wingsap1**](https://courses.edx.org/courses/course-v1:MITx+6.00.2x+3T2017/discussion/forum/users/80622)

I will explain on how things change for multiple items. I will explain the significance of i and j and the states of an item depending upon the scenarios. So bear with me for some time.

First lets explain "States" : When you toss a coin the outcome is always 'Head' or 'Tail'. So for each flip you have two states that can possibly occur. Let's say you do 5 flips then you will have 2\*2\*2\*2\*2 states for each flip. We multiply the states cause result of each flip is mutually exclusive event. So in the end we have 2^5 states.

So for our problem we have two bags and a few real items. Real items cause they can occur at one state at one time. By that I mean one item can only be in either bag1 or bag2 but not in both bags simultaneously.

Now we have 3 states for each item :- Not in any Bag, In Bag1, or In Bag2. Let's say we have 3 items so we will have 3\*3\*3 states = 3^3 states in total. By total states I mean the total possible permutations we can arrange the items where order matters. That means Item 1 in Bag1 is a different scenario from item in Bag2.

Now coming to significance of "i" which denotes exactly what I explained above that is all possible states. And "j" = Length(of the list) = total no. of items.

So when we iterate through every possible permutation of states, that is "i", we check for every single "i" which of the "j" items we are taking. For example if we have 3 items with 3 states we have total 3\*\*3 = 27 states for each state we see which of the 3 items we carry or not carry.

Now coming to "bitwise operation" - Bitwise helps us enumerate these possibilities for each state using binary or Ternary bits depending on what is being dealt with. If we have had 5 states for each item instead of 2 or 3, we would be using base 5 number system.

So how do we convert to base 3 operation, it is simple while we were doing binary "x>>y" the operation simply means shifting x by y positions to the right. Mathematically it's the integer division by 2^y.

Extrapolating the idea to base 3, we can represent "x>>y" in base 3 as x//3^y.

Now let's think how and "why" it works. Representing the 27 total states for 3 items in Ternary system :-

000 - State 1

001 - State 2

002

010

011

012

020

021

022

200 - State 9

.

.

.

.

222 - State 27

Where the 3 bits represent the 3 items we have. So for state-1 we have 0 representing all the items and for state-9 we have item1 as 0, item2 as 0 and item 3 as 2.

Why in this order? Because remember we are right shifting the bits.

Why are we right shifting? Because we want to check each item once only for each state else we will have repetitions.

Now why are we taking modulus of the shifted bit? We are doing it to create a logic for our items so that we can determine which ones to carry and which to not. You may be thinking why 1 is bag1 and why 2 remainder is bag 2.

Well it doesn't have to be that way. It is simply for representation purpose we are doing. You can very well put "0" remainders in bag 1 or 2. Whichever you will like as we are checking all the states it does not matter which state does what operation. Only the end result matters.

Here is the code that shows you the shifted values for each item evaluation whether to take or not.

items = [1,2,3]

def yieldAllCombos(items):

"""

Generates all combinations of N items into two bags, whereby each item is in one or zero bags.

Yields a tuple, (bag1, bag2), where each bag is represented as a list of which item(s) are in each bag.

"""

N = len(items)

# Enumerate the 3\*\*N possible combinations

for i in range(3\*\*N):

bag1 = []

bag2 = []

print('i',i)

for j in range(N):

print('right shift',i // (3 \*\* j))

print('remainder',(i // (3 \*\* j))%3)

if (i // (3 \*\* j)) % 3 == 0:

bag1.append(str(items[j]))

elif (i // (3 \*\* j)) % 3 == 2:

bag2.append(str(items[j]))

yield (bag1, bag2)

Change values, changes items and remainders. Play with it and understand at root level.