# Task 4

A “machine” consists of a row of boxes. To start, one places n pennies in the leftmost box. The machine then redistributes the pennies as follows.

On each iteration, it replaces a pair of pennies in one box with a single penny in the next box to the right. The iterations stop when there is no box with more than one coin. For example, see the figure that shows the work of the machine in distributing six pennies by always selecting a pair of pennies in the leftmost box with at least two coins.

Design an algorithm using greedy method automate the machine, then answer the following questions.

(a) Does the final distribution of pennies depend on the order in which the machine processes the coin pairs?

(b) What is the minimum number of boxes needed to distribute n pennies?

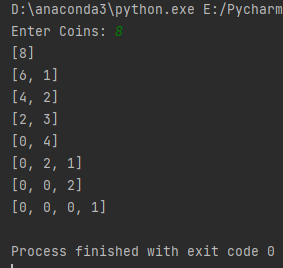
(c) How many iterations does the machine make before stopping?

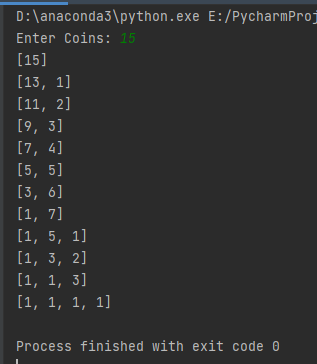
**Solution:**

* **Greedy Algorithm (written in Python Programming Language)**

if \_\_name\_\_ == "\_\_main\_\_":  
 # asking user to input coins in the leftmost box to process  
 initCoins = int(input("Enter Coins: "))  
  
 # resultArr is the array where we will save our final distribution  
 # resultArr will change each iteration to reflect new distributions  
 resultArr = [initCoins]  
 print(resultArr)  
 ind = 0  
  
 # checking whether there still an element not equal to 0 or 1  
 while resultArr[ind] > 1:  
 # Add a zero to right of this element to increment it later  
 resultArr.append(0)  
  
 # Here comes the greedy concept  
 # loop fetches the element with value > 1  
 # the element with value > 1 is considered the local region  
 while resultArr[ind] > 1:  
 # deducting 2 coins from current box  
 resultArr[ind] -= 2  
 # adding 1 coin to the next right boxx  
 resultArr[ind + 1] += 1  
   
 # show coin distribution each time  
 print(resultArr)  
 # the local region is solved completely, now we can move to the next local region (element)  
 ind += 1

* Sample input and Corresponding Output:





* Pseudo-code of the Second Algorithm:

MachineDist(n):

INTEGER myVALUE 🡨 0, Iters 🡨 0

INTEGER Len 🡨 log2(n)

INTEGER arr[Len]

INTEGER\_VECTOR pennArr[Len]

pennArr[0] 🡨 0

**FOR** i 🡨 0 to Len **DO**

**WHILE** (pennArr[i] > 1)

pennArr[i] 🡨 boxes[i] – 2

myVALUE 🡨 myVALUE + 1

pennArr[i+1] 🡨 myVALUE

**END WHILE**

**END LOOP**

myVal 🡨 0

1. No, the final distribution doesn’t depend on the order in which the machine processes the coin pairs

Because,

If we consider an arbitrary string (S1 S2 … Sk) showing the distribution of n coins. Now, consider i such that **0 ≤ i ≤ k,**and that Si may be either 1 or 0. Let Sk = 1, where k is index of element the rightmost box (last element), the value of Sk is evaluated by subtracting 2 from position k-1 and thus 4 has been subtracted from position k-2 and same goes for any selected box. From the above we can deduce the following:

Which means that the final distribution is the reverse of the binary representation of the coins value initially entered by the user, thus the final distribution doesn’t depend on the order in which the machine processes the coin pairs

1. Since the final distribution is the reverse of the binary representation of the coins value, so the minimum number of boxes will be equal to minimum number of bits needed to represent the binary value of the coins entered by the user. n is the number of coins entered by the user.
2. We will be using recurrence relation to solve this and find the iterations that the machine make before stopping

At a give iteration i such that **0 ≤ i ≤ k** we will need two steps for the previous iteration (i-1) and one step for the current one (i)

f(i) = 2 \* f(i − 1) + 1, f(0) = 0

*f*(*i*) = 2 \* *f*(*i* − 1) + 1

= 2 \* (2 \* *f*(*i* − 2) + 1) + 1 = 22\* *f*(*i* − 2) + 2 + 1

= 22 \* (2 \* *f*(*i* − 3) + 1) + 2 + 1 = 23 *\* f*(*i* − 3) + 22+ 2 + 1

= 2i *\* f*(*i* − *i*) + 2i-1 + 2i-2 + · · · + 1 = 2i· 0 + (2i− 1) = 2i− 1*.*

From the rule we found above we can say that the total number of iterations will be:

And that is simply the difference between the number of coins initially entered by the user and the number of ones in its binary expansion.