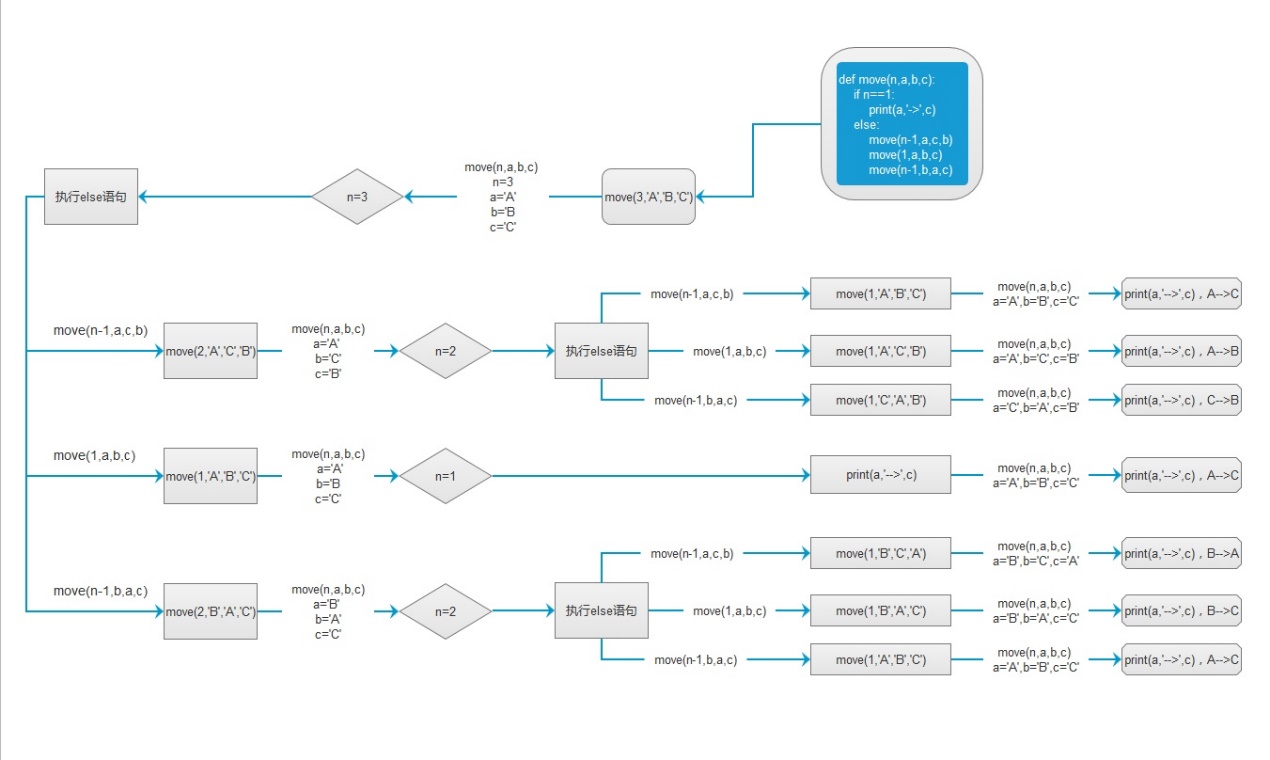
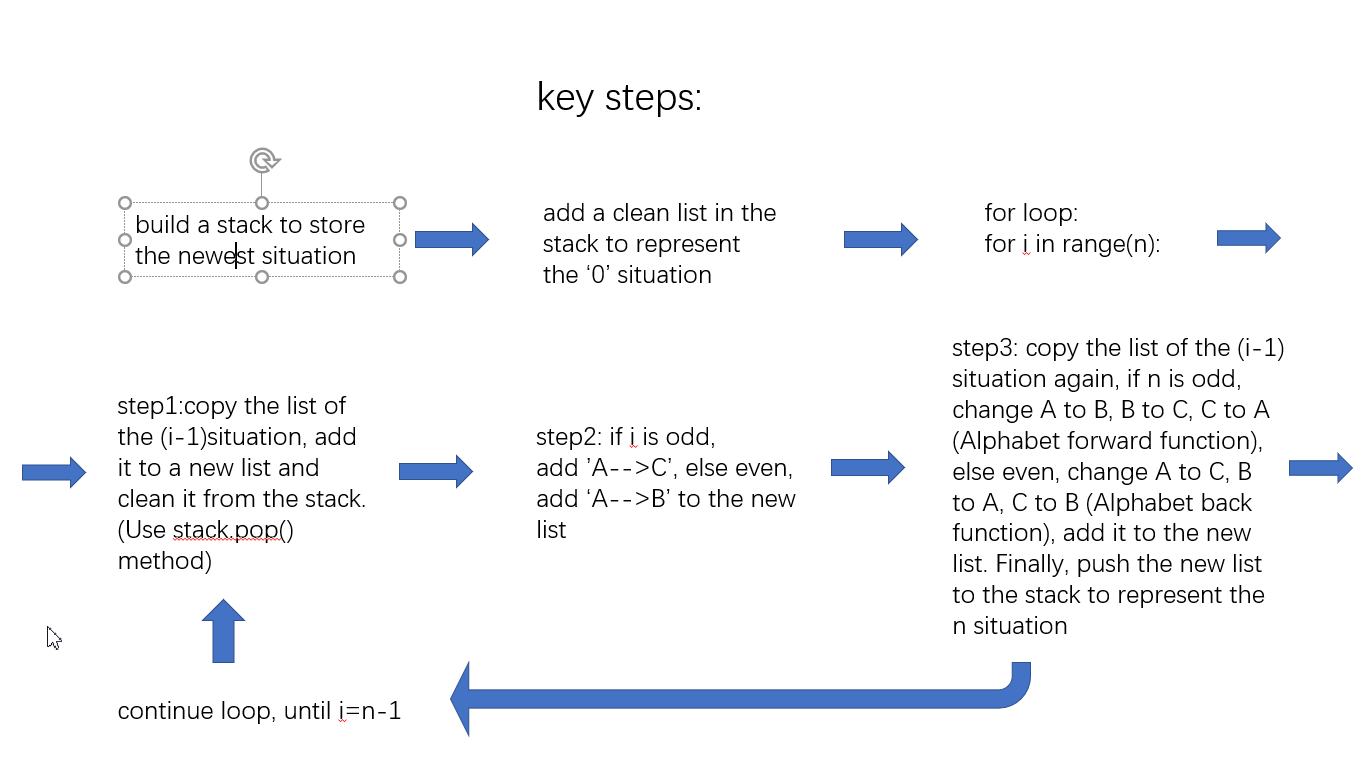
The recursive algorithm for the Tower of Hanoi is very simple, the thought and the codes are shown on the following picture:



My non-recursive function is to transfer the recursive function into non-recursive way. Therefore, I choose to use stack.

My main idea is quite same with the recursive function. To do 4, you firstly need to do 3. To do 3, you need to do 2 at first. Therefore, all the situation come from 1. We need to start with 1 every time. I prepare a stack to store the situation before the number we need. To show it clearly, I decide to show you some picture.



def Alphabet\_forward(n):

Alphabet\_list=list(n)

new\_list=list()

for i in Alphabet\_list:

if i=='A':

i='B'

elif i=='B':

i='C'

elif i=='C':

i='A'

new\_list.append(i)

new\_string=''.join(new\_list)

return new\_string

def Alphabet\_back(n):

Alphabet\_list=list(n)

new\_list=list()

for i in Alphabet\_list:

if i=='A':

i='C'

elif i=='B':

i='A'

elif i=='C':

i='B'

new\_list.append(i)

new\_string=''.join(new\_list)

return new\_string

def HanoiTower(n):

Situation\_Stack=listStack()

step\_list=list()

Situation\_Stack.push(step\_list)

for i in range(n):

step\_list=Situation\_Stack.pop()

if i%2==0:

new\_step\_list=list()

for j in step\_list:

new\_step\_list.append(Alphabet\_back(j))

step\_list.append('A-->B')

step\_list+=new\_step\_list

else:

new\_step\_list=list()

for j in step\_list:

new\_step\_list.append(Alphabet\_forward(j))

step\_list.append('A-->C')

step\_list+=new\_step\_list

Situation\_Stack.push(step\_list)

for i in step\_list:

print(i)

It basically is transferring the recursive algorithm into loop with stack.