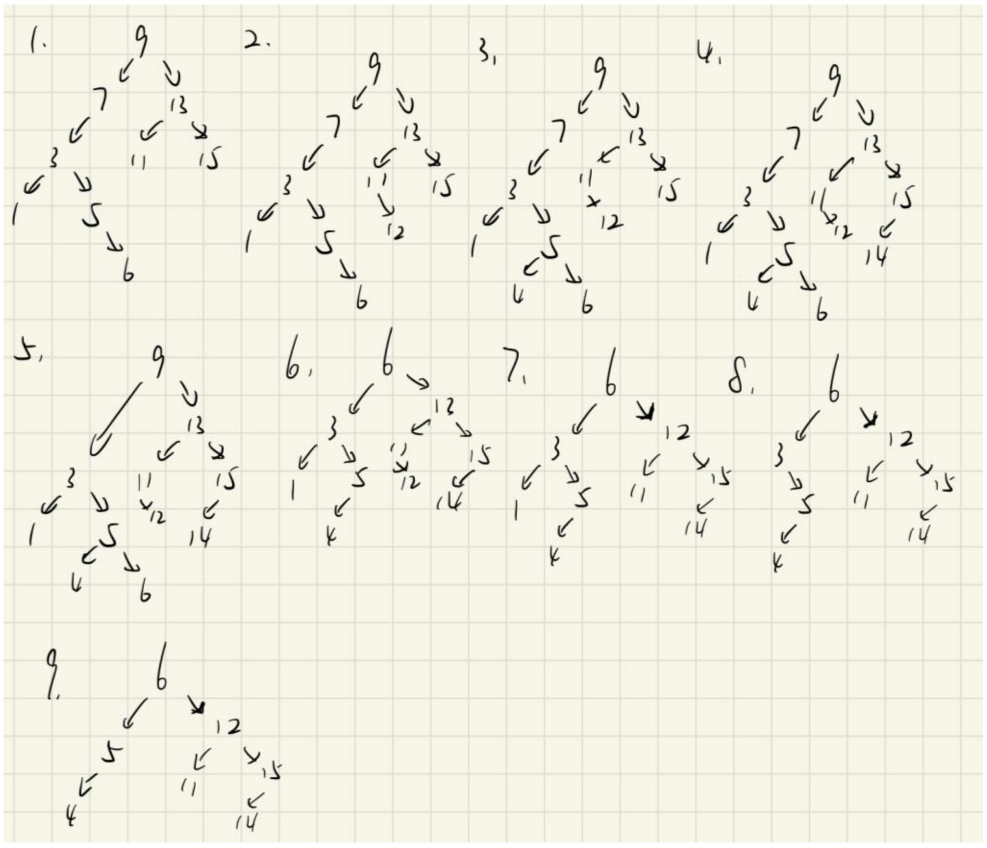
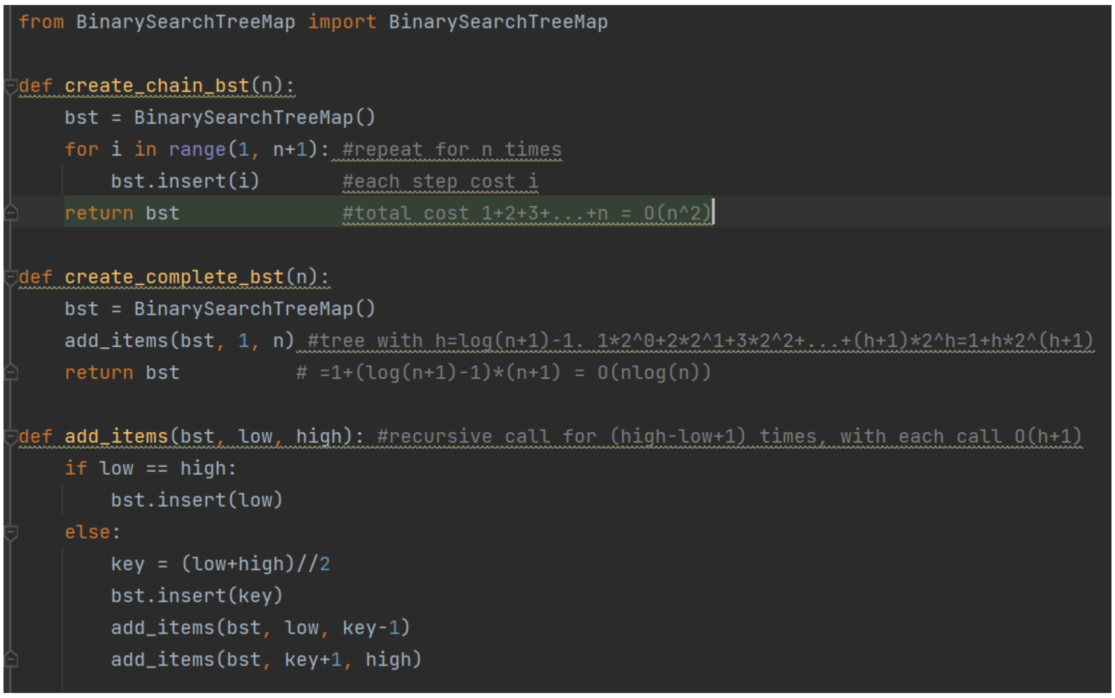
Q1:



Q2(c):



```
from BinarySearchTreeMap import BinarySearchTreeMap

def create_chain_bst(n):
    bst = BinarySearchTreeMap()
    for i in range(1, n+1): #repeat for n times
        bst.insert(i)       #each step cost i
    return bst              #total cost 1+2+3+...+n = O(n^2)

def create_complete_bst(n):
    bst = BinarySearchTreeMap()
    add_items(bst, 1, n) #tree with h=log(n+1)-1. 1*2^0+2*2^1+3*2^2+...+(h+1)*2^h=1+h*2^(h+1)
    return bst          # =1+(log(n+1)-1)*(n+1) = O(nlog(n))

def add_items(bst, low, high): #recursive call for (high-low+1) times, with each call O(h+1)
    if low == high:
        bst.insert(low)
    else:
        key = (low+high)//2
        bst.insert(key)
        add_items(bst, low, key-1)
        add_items(bst, key+1, high)
```