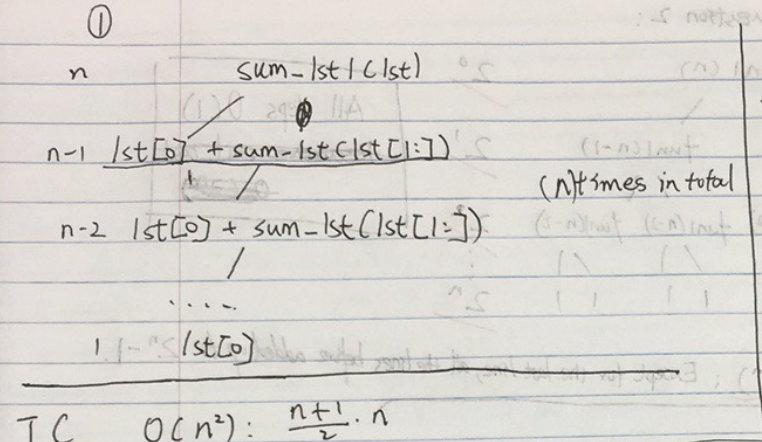
Question 1

①




```
if (len(lst) == 1):        O(1)
    return lst[0]          O(1)
else:                      O(1)
    rest = sum_lst1(lst[1:])   O(n)
    sum = lst[0] + rest        O(1)
    return sum             O(1)
```

TC $O(n^2)$: $\frac{n+1}{2} \cdot n$

②

sum_lst2(lst, low, high):

1 lst[low] + sum_lst2(lst, low+1, high)

1 lst[low+1] + sum_lst2(lst, low+2, high)

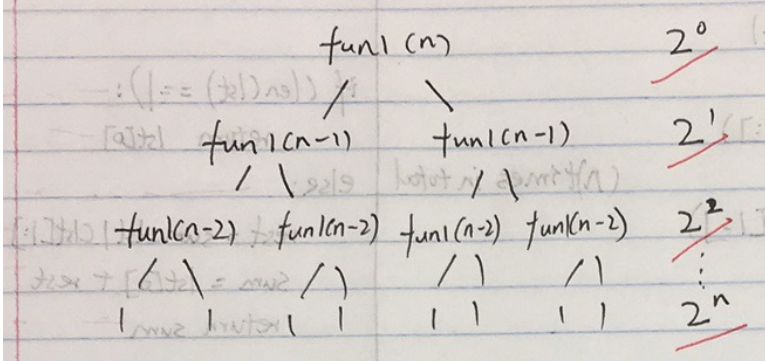
1 lst[high] + sum_lst2(lst, high, high)

n times in total

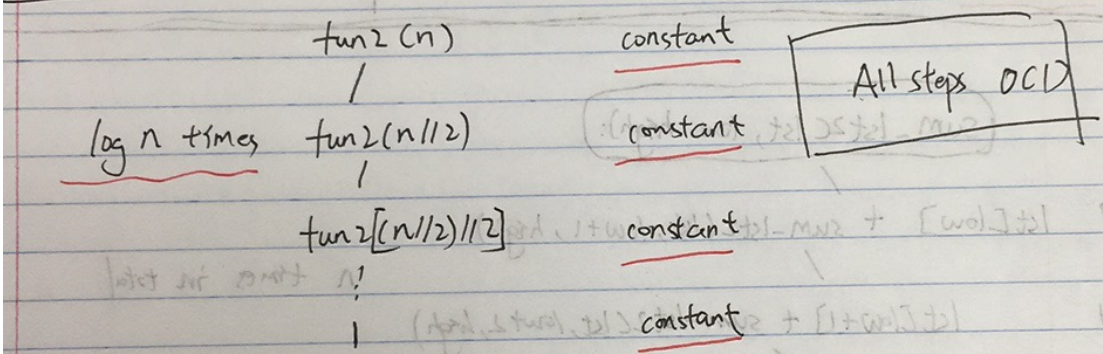
Time Complexity: O(n)

All steps called cost O(1)

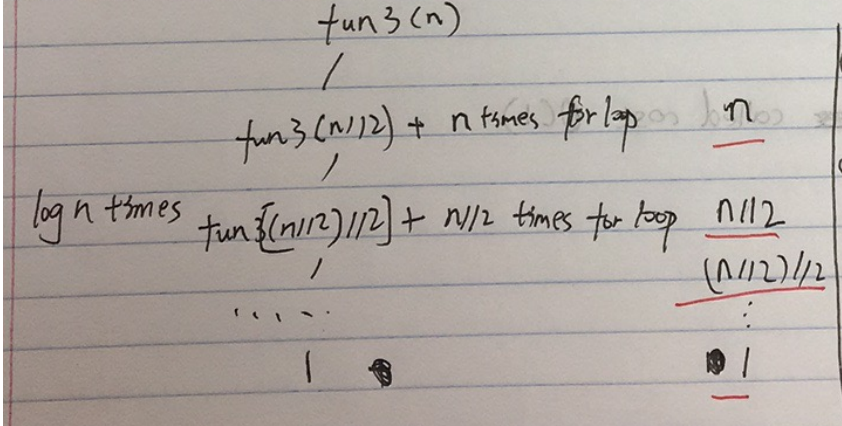
Question 2:



O(2^n): Except for the last line, all the lines before added up to $2^n - 1$.



O(log n)



run times: $\frac{1 + n \times 2}{1-2} = 2n = O(n)$

```
O(1) def fun3(n):
O(1)     if(n==0):
O(1)         return 1
O(1)     else:
O(1)         res=fun3(n//2)
O(n)         for i in range(1,n+1):
O(1)             res=i
```