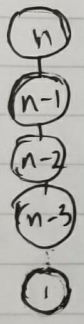
Homework 4

Question 1:

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

def sum_lst2(lst, low, high):
    if (low == high):
        return lst[low]
    else:
        rest = sum_lst2(lst, low+1, high)
        sum = lst[low] + rest
        return sum
```

sum_lst1

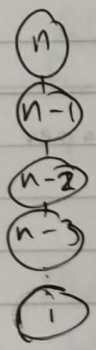


level #	# of calls	cost of local
0	1	n
1	1	n
2	1	n
⋮		
n	1	n

Total Cost = $\Theta(n^2)$

Splicing cost $\Theta(n)$ for n calls therefore $\Theta(n^2)$

sum_lst2



level #	# of calls	cost of local
0	1	1
1	1	1
2	1	1
⋮		
n	1	1

Total Cost = $\Theta(n)$

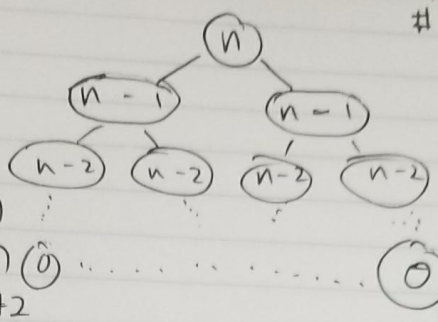
each local cost $\Theta(1)$ for n calls therefore $\Theta(n)$

sum_lst2 is asymptotically faster

Homework 4

Question 2:

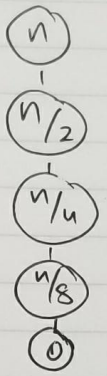
```
def fun1(n):
    if (n==0):
        return 1
    else:
        part1 = fun1(n-1)
        part2 = fun1(n-1)
        res = part1 + part2
        return res
```



# levels	local	# calls
0	1	1
1	1	2
2	1	4
3	1	8
⋮	⋮	⋮
n	1	2^k

$T(n) = \theta(2^n)$

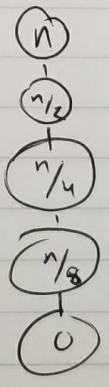
```
def fun2(n):
    if (n==0):
        return 1
    else:
        res = fun2(n//2)
        res += n
        return res
```



# levels	local cost	# calls
0	1	1
1	1	1
2	1	1
3	1	1
⋮	1	1
log(n)	1	1

$T(n) = \theta(\log(n))$

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



# levels	local cost	# calls
0	n	1
1	n	1
2	n	1
3	n	1
⋮	n	1
log(n)	n	1

$T(n) = \theta(n\log(n))$