1)

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

$\theta(n)$

---

n	$\theta(k)$
$n-1$	$\theta(k)$
$n-2$	$\theta(k)$
$n-3$	$\theta(k)$
$\vdots$	
$n-(n)$	$\theta(k)$

k = slice size

n levels of $\theta(n)$

Time Complexity:

$\theta(n^2)$

```
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
```

$\Theta(1)$ ←

n	$\Theta(1)$
1	
$n-1$	$\Theta(1)$
1	
$n-2$	$\Theta(1)$
⋮	
$n-(n)$	$\Theta(1)$

Also n levels of $\Theta(1)$ each

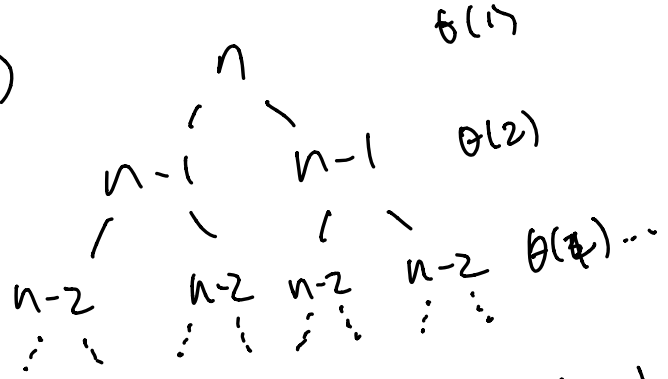
Time Complexity

$\Theta(n)$

Implementation 2 is faster.

2)

a)



h levels of constant time

$\Rightarrow \theta(n)$

b)

$\log_2(n)$ levels of constant time

$\Rightarrow \theta(\log(n))$

C)

n	$\theta(n)$
$\vert$	
$n/2$	$\theta(k)$
$\vert$	
$n/4$	$\theta(k)$
$\vert$	
$n/8$	$\theta(k)$

$\log_2(n)$ levels of

$\theta(k)$: $n + n/2 + n/4 + \dots$

$\Rightarrow \theta(n\log(n))$