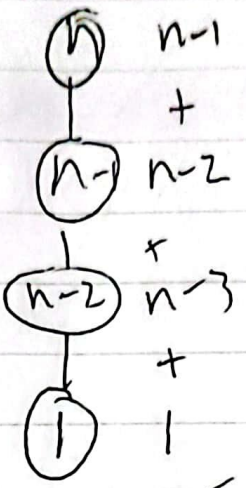
Hw4

1. 2) 3)

for sum_lst1;

The first call will need to copy the length of the list -1, then the next one less, etc. until it copies one.

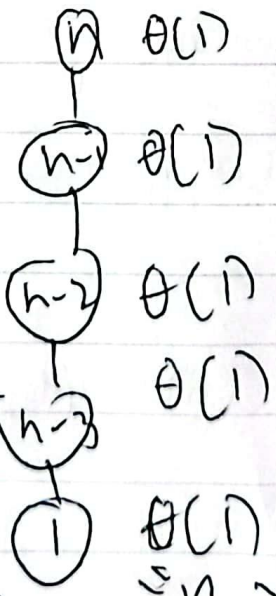


$$\frac{n^2 - n}{2} = \boxed{\theta(n^2)}$$

This is a sum of $\frac{n(n-1)}{2}$, which is $\theta(n^2)$

for sum_lst2;

For every recursive call the same list is passed so no copying is necessary, this means every call runs in constant time, giving an overall runtime of theta(n)

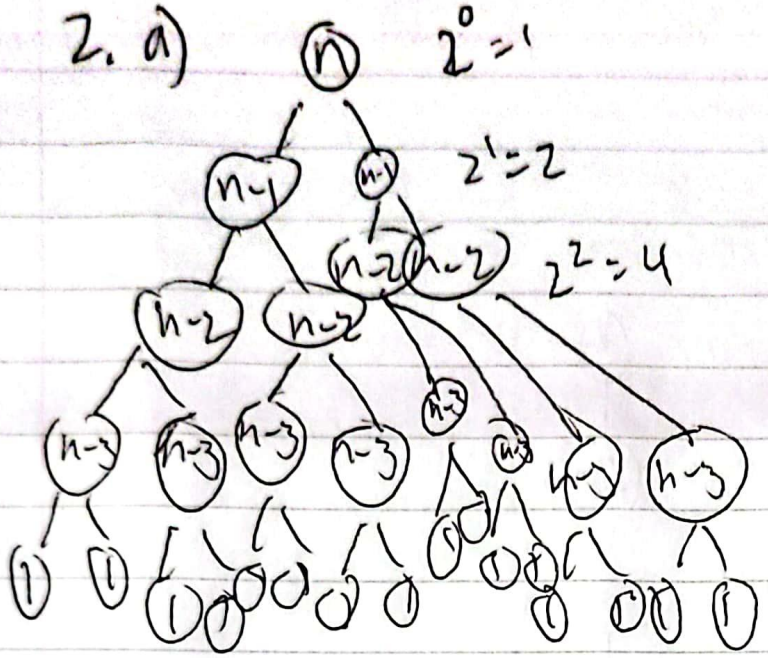


$= n \rightarrow \theta(n)$

3) The second function is asymptotically faster because it runs in linear time whereas the first runs in n^2 time.

n is a power of 2

2. a)



$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

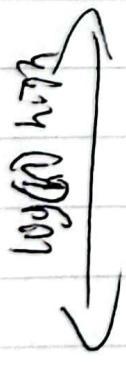
2^n on bottom layer

$2^0 + 2^1 + 2^2 + 2^k = 2^{k+1} - 1$

Each call other than the recursion runs in constant time, so the total runtime is

$1 \cdot (2^{n+1} - 1) = 2^{n+1} - 1 = \Theta(2^n)$

b)



$\Theta(1)$
$\Theta(1)$
$\Theta(1)$
$\Theta(1)$

Each call runs in $\Theta(1)$ time, and it runs $\log(n)$ times, so the total runtime is

$1 \cdot \log_2(n) = \Theta(\log_2(n))$

c)

$\Theta(n)$
$\Theta(n/2)$
$n/4$
1

The function will be called $\log(n)$ times, and each time the cost of the call will cut in half.

$n + \frac{n}{2} + \frac{n}{4} + 1 = 2n - 1 = \boxed{\Theta(n)}$