Let x and L be LinkedList Nodes.

Analyzing append

```
1 append(x, L) {
2    Node curr = L;
3    while (curr != null && curr.next != null) {
4       curr = curr.next;
5    }
6    curr.next = x;
7 }
```

$O(n)$

LinkedList Reversal

```
1  reverse(L) {
2     if (L == null) {
3        return null;
4     }
5  } else {
6        Node front = L;
7        Node rest = L.next;
8        L.next = null;
9
10       Node restReversed = reverse(rest);
11       append(front, restReversed);
12    }
13 }
```

else if (L.next == null) { return L;

$$T(n) = \begin{cases} d_0 & \text{if } n = 0 \text{ or } 1 \\ c_0 + c_1 n + T(n-1) & \text{otherwise} \end{cases}$$

$$\begin{aligned} T(n) &= (c_0 + c_1 n) + T(n-1) \\ &= (c_0 + c_1 n) + (c_0 + c_1(n-1)) + T(n-2) \\ &= (c_0 + c_1 n) + (c_0 + c_1(n-1)) + (c_0 + c_1(n-2)) + \ldots + (c_0 + c_1(1)) + d_0 \end{aligned}$$

$$= \sum_{i=1}^{n} (c_0 + c_1 i) + d_0$$

$$= n \cdot c_0 + c_1 \sum_{i=1}^{n} i + d_0$$

$$= n \cdot c_0 + c_1 \left(\frac{n(n+1)}{2} \right) + d_0 = O(n^2)$$

```
int sum(int[] arr){
    return help(arr,0,arr.length);
}
int help(int[] arr, int lo, int hi) {
    if(lo==hi)    return 0;
    if(lo==hi-1)  return arr[lo];
    int mid = (hi+lo)/2;
    return help(arr,lo,mid) + help(arr,mid,hi);
}
```

C_3

$T(0) = c_1$

$T(1) = c_2$

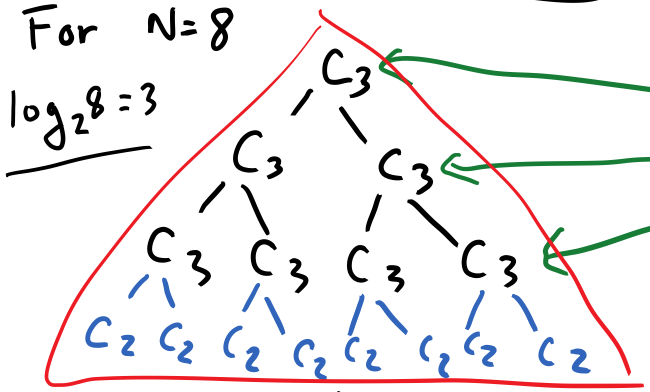
$T(N) = c_3 + 2 \cdot T\left(\frac{N}{2}\right)$

C_3

$T\left(\frac{N}{2}\right)$ $T\left(\frac{N}{2}\right)$

For $N = 8$

$\log_2 8 = 3$



"level"	# nodes at this level	Total Work at this level
0	1	$1 \cdot c_3$
1	2^1	$2\ c_3$
2	2^2	$4 \cdot c_3$
⋮	⋮	⋮
λ	2^λ	$2^\lambda \cdot c_3$

$$T(N) = \sum_{\lambda=0}^{(\log_2 N)-1} 2^\lambda c_3 + c_2 \cdot N$$

$$= c_3 \sum_{\lambda=0}^{(\log_2 N)-1} 2^\lambda + c_2 \cdot N$$

$$= c_3 \cdot \left(2^{\log_2 N} - 1\right) + c_2 \cdot N$$

$$= c_3 \cdot (N-1) + c_2 \cdot N = c_3 \cdot N - c_3 + c_2 \cdot N$$

$$= O(N)$$

$$\sum_{\lambda=0}^{k} 2^\lambda = 2^{k+1} - 1$$