Solving the `reverse` Recurrence

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

$T(n) = (c_0 + c_1 n) + T(n-1)$

$= (c_0 + c_1 n) + (c_0 + c_1(n-1) + T(n-2))$

$= \sum_{i=0}^{n-1} (c_0 + c_1[n-i]) + d_0$

Merge Sort

```
1  sort(L) {
2     if (L.size() < 2) {
3        return L;
4     }
5     else {
6        int mid = L.size() / 2;
7        return merge(
8           sort(L.subList(0, mid)),
9           sort(L.subList(mid, L.size()))
10       );
11    }
12 }
```

First, we need to find the recurrence:

$T(n) =$ $2T(n/2) + 1$

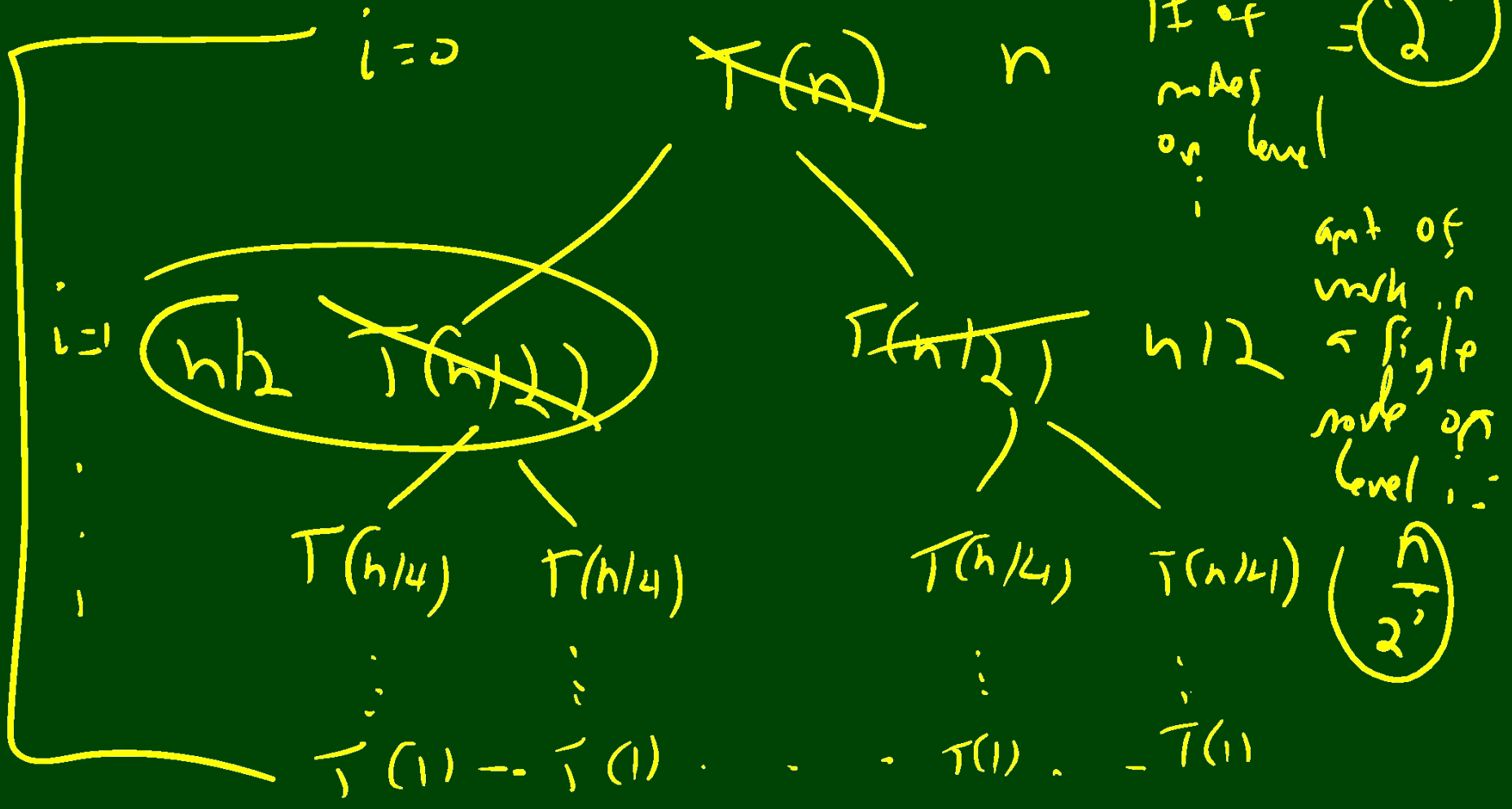
NR $= O(1) + O(n) = O(n)$

R $= 2T(n/2)$

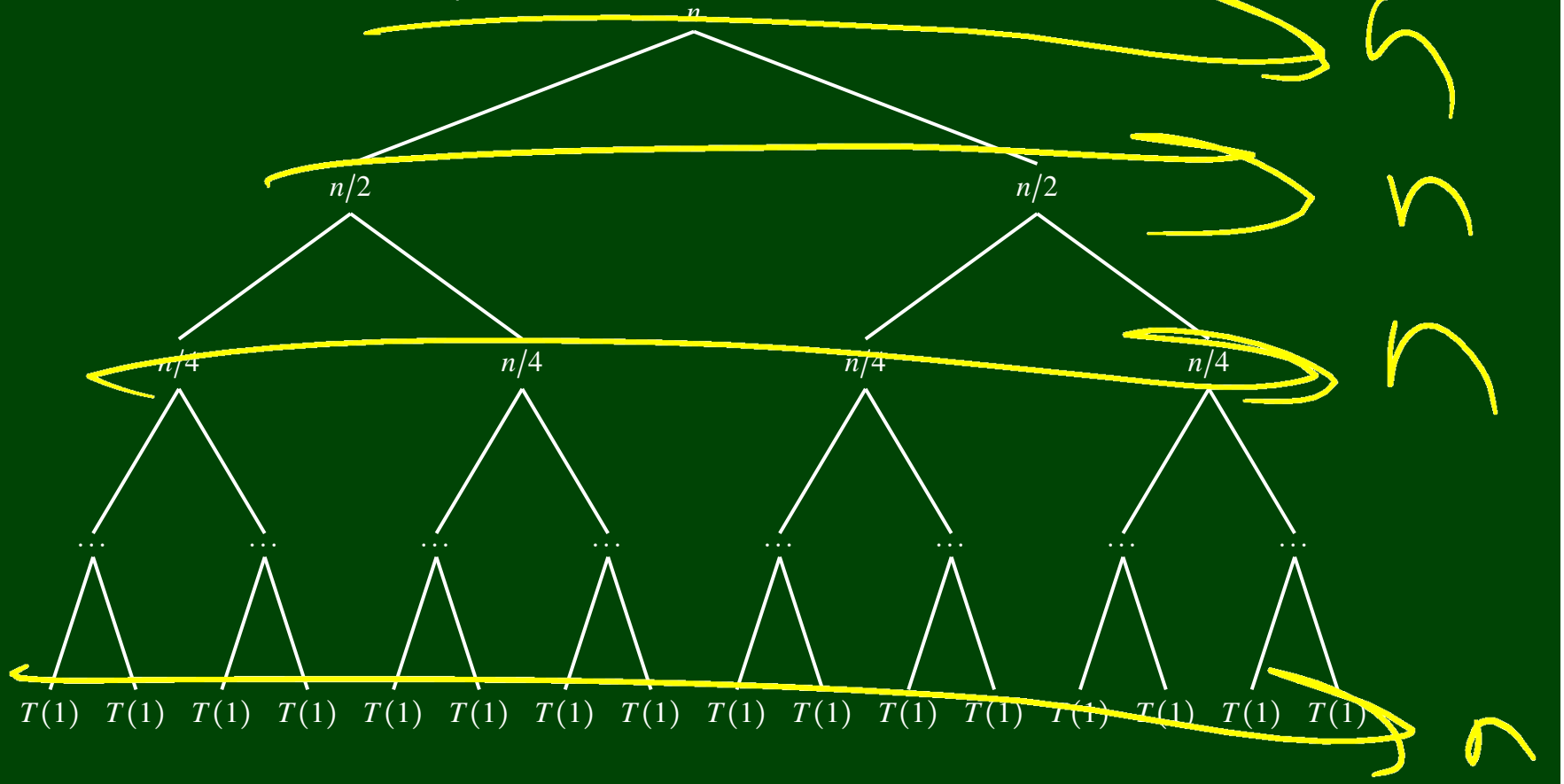
BC $= O(1)$

height $= \lg(n)$

$$T(n) = \begin{cases} d_0 & \text{if } n = 0 \\ d_1 & \text{if } n = 1 \\ \underbrace{c_0 + c_1 n + 2T(n/2)}_{T(n)} & \text{otherwise} \end{cases}$$



$$T(n) = \begin{cases} d_0 & \text{if } n = 0 \\ d_1 & \text{if } n = 1 \\ c_0 + c_1 n + 2T(n/2) & \text{otherwise} \end{cases}$$



Find A Big-Oh Bound For The Worst Case Runtime

```
1 sum(n) {
2     if (n < 2) {
3         return n;
4     }
5     return 2 + sum(n - 2);
6 }
```

BC

NR NR

R

$$T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 1 + T(n-2) & \text{otherwise} \end{cases}$$

$$\underbrace{1 + 1 + 1 + \cdots + 1}_{n/2}$$

Find A Big-Oh Bound For The Worst Case Runtime

```
1  binarysearch(L, value) {
2     if (L.size() == 0) {
3        return false;
4     }
5     else if (L.size() == 1) {
6        return L[0] == value;
7     }
8     else {
9        int mid = L.size() / 2;
10       if (L[mid] < value) {
11          return binarysearch(L.subList(mid + 1, L.size()), value);
12       }
13       else {
14          return binarysearch(L.subList(0, mid), value);
15       }
16    }
17 }
```

$T(n) = 1 + \max(T(n/2), T(n/2))$

Master Theorem

Consider a recurrence of the form:

$$T(n) = \begin{cases} d & \text{if } n = 1 \\ aT\left(\frac{n}{b}\right) + n^c & \text{otherwise} \end{cases}$$

Then,

- If $\log_b(a) < c$, then $T(n) = \Theta(n^c)$.
- If $\log_b(a) = c$, then $T(n) = \Theta(n^c \lg(n))$.
- If $\log_b(a) > c$, then $T(n) = \Theta(n^{\log_b(a)})$.