

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

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

$$(c_0 + c_1 n) + (c_0 + c_1(n-1)) + \dots + (c_0 + c_1(n-2))$$

$$\text{it} \rightarrow \left(\sum_{i=0}^{n-1} (c_0 + c_1(n-i)) \right) + d_0 + c_1(n-2) + T(n-3)$$

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 }
    
```

Handwritten annotations on the code:

- A blue circle around lines 2-4 is labeled BC .
- A red circle around line 6 is labeled NR .
- A red circle around line 7 is labeled NR .
- A yellow circle around line 8 is labeled R .
- A green circle around line 9 is labeled R .
- A red circle around line 10 is labeled R .

First, we need to find the recurrence:

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

$$BC = \Theta(1)$$

$$NR = \Theta(n)$$

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

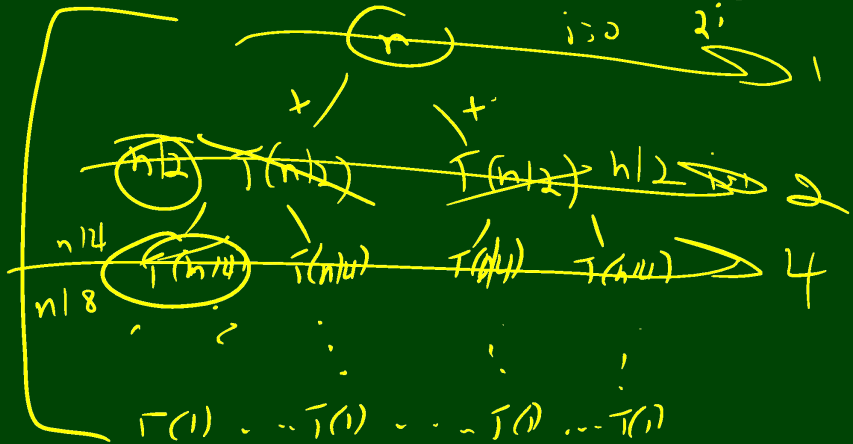
Merge Sort: Solving the Recurrence

height = $\lg(n)$

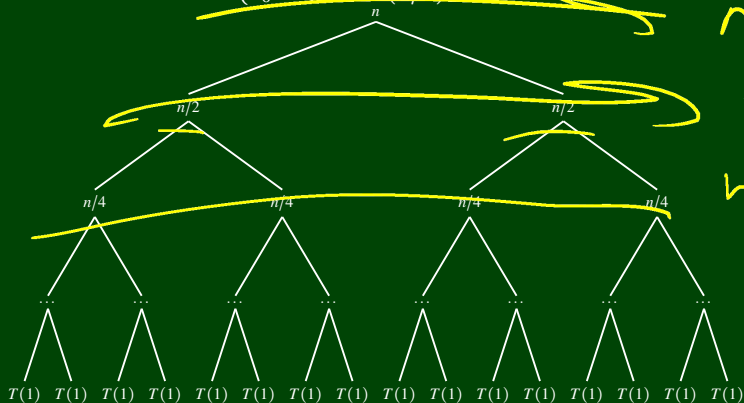
$$\sum_{i=0}^{\lg(n)-1} n^{1/2^i}$$

$$T(n) = \begin{cases} d_0 & \text{if } n=0 \\ d_1 & \text{if } n=1 \\ c_0 + c_1n + 2T(n/2) & \text{otherwise} \end{cases}$$

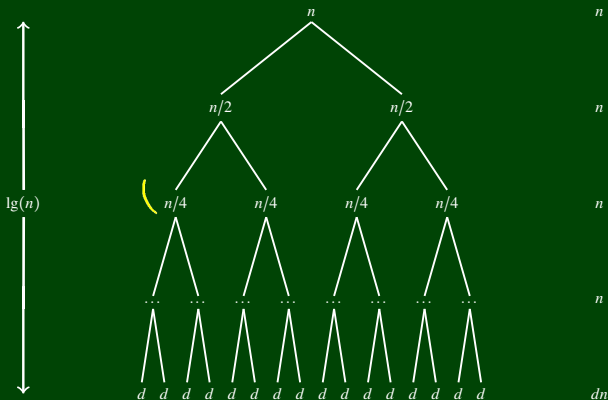
Am't of nr work in a single node on level i : $n^{1/2^i}$



$$T(n) = \begin{cases} d_0 & \text{if } n = 0 \\ d_1 & \text{if } n = 1 \\ c_0 + c_1n + 2T(n/2) & \text{otherwise} \end{cases}$$



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



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 }

```

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

$$T(n) = \underbrace{1 + 1 + \dots + 1}_{n/2}$$

Find A Big-Oh Bound For The Worst Case Runtime

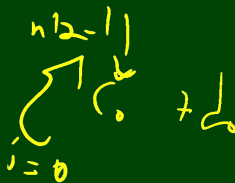
```

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

```

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

$$\begin{aligned}
T(n) &= c_0 + c_0 + \dots + c_0 + d_0 \\
&= c_0 \left(\frac{n}{2} \right) + d_0 \\
&= \mathcal{O}(n)
\end{aligned}$$



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 }

```

Handwritten annotations in yellow:

- A circle around line 9: $n/2$
- A circle around line 10: $n/2$
- A circle around line 11: $n/2$
- A circle around line 13: $n/2$

$$T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 1 + \max(T(n/2), T(n/2)) & \end{cases}$$