

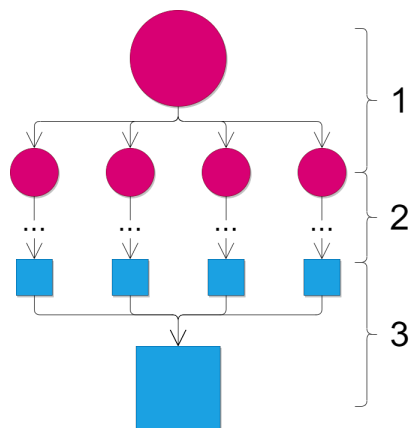
# 50001 - Algorithm Analysis and Design - Lecture 6

Oliver Killane

12/11/21

## Divide & Conquer

1. Divide a problem into subproblems
2. Divide and conquer subproblems into subsolutions
3. Conquer subsolutions into a solution



## Merge Sort

```

1 msort :: Ord a => [a] -> [a]
2 msort [] = []
3 msort [x] = [x]
4 msort xs = merge (msort us) (msort vs)
5   where (us, vs) = splitAt (length xs `div` 2) xs
6
7 merge :: Ord a => [a] -> [a] -> [a]
8 merge [] ys = ys
9 merge xs [] = xs
10 merge xss@(x:xs) yss@(y:ys)
11   | x <= y = x : merge xs yss
12   | otherwise = y : merge xss ys

```

**SplitAt** divides, and **merge** Conquers. We can calculate the time complexity for the recurrence relations below (based on recursive structure of **msort**):

$$T_{msort}(0) = 1$$

$$T_{msort}(1) = 1$$

$$T_{msort}(n) = T_{length}(n) + T_{splitAt}\left(\frac{n}{2}\right) + T_{merge}\left(\frac{n}{2}\right) + 2 \times T_{msort}\left(\frac{n}{2}\right)$$

We can simplify the complexity of **msort**

$$\begin{aligned}
 T_{msort}(n) &= T_{length}(n) + T_{splitAt}\left(\frac{n}{2}\right) + T_{merge}\left(\frac{n}{2}\right) + 2 \times T_{msort}\left(\frac{n}{2}\right) \\
 &= n + \frac{n}{2} + \frac{n}{2} + \frac{n}{2} + 2 \times T_{msort}\left(\frac{n}{2}\right) \\
 &= \frac{5}{2} \times n + 2 \times T_{msort}\left(\frac{n}{2}\right)
 \end{aligned}$$

#### Master Theorem

For an algorithm *algo* such that:

$$T_{algo}(n) = a \times T_{algo}\left(\frac{n}{b}\right) + f(n) + \text{base cases}$$

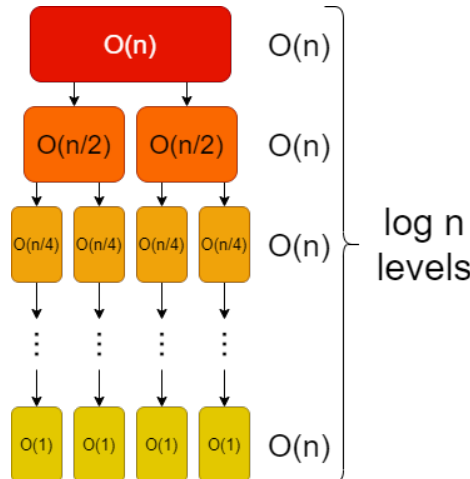
The work at recursion level  $\log_b n$  is  $\Theta(a^{\log_b n})$  To calculate the order of the time complexity:

1. Get the recurrence relation in the form above.
2. Get the critical exponent  $E$  by formula:  $E = \log_b a = \frac{\log a}{\log b}$ .
3. Given  $f(n) = n^c$  we can express the work as a geometric sum  $\sum_{i=0}^{\log_b n} ar^i$  where  $r = \frac{a}{b^c}$ .

$$r > 1 \Leftrightarrow a > b^c \Leftrightarrow \log_b a > c \Leftrightarrow E > c$$

$$\begin{aligned}
 E < c & \quad T_{algo}(n) \in \Theta(f(n)) \\
 E = c & \quad T_{algo}(n) \in \Theta(f(n) \log_b n) = \Theta(f(n) \log n) \\
 E > c & \quad T_{algo}(n) \in \Theta(a^{\log_b n}) = \Theta(n^{\log_b a}) = \Theta(n^E)
 \end{aligned}$$

By master theorem we can easily see  $T_{msort}(n) \in \Theta(n \log n)$ . We can also calculate it using a graph:



## Quicksort

```

1  qsort :: Ord a => [a] -> [a]
2  qsort [] = []
3  qsort [x] = [x]
4  qsort (x:xs) = qsort us ++ x:qsort vs
5      where (us, vs) = partition (<x) xs
6
7  partition :: (a -> Bool) -> [a] -> ([a],[a])
8  partition p xs = (filter p xs, filter (not . p) xs)
9
10 dxgfdg

```

Note for simplicity, we assume the lists are split into equal parts.

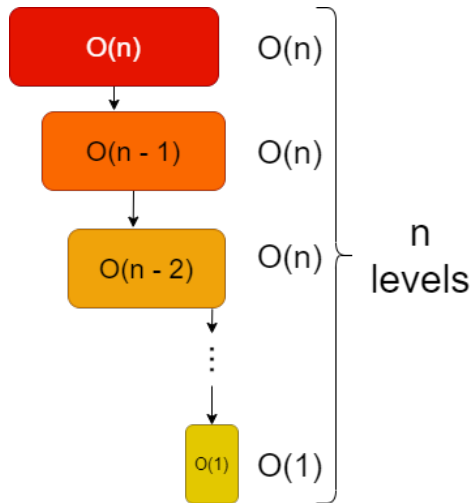
$$T_{qsort}(0) = 1$$

$$T_{qsort}(1) = 1$$

$$T_{qsort}(n) = T_{partition}(n-1) + T_{++}\left(\frac{n}{2}\right) + 2 \times T_{qsort}\left(\frac{n}{2}\right)$$

In the worst case, the partition splits  $xs$  into  $(xs, [])$ , we have complexity:

$$\begin{aligned}
 T_{qsort}(n) &= T_{partition}(n-1) + T_{++}(n-1) + T_{qsort}(0) + T_{qsort}(n-1) \\
 &= 2(n-1) + n + 1 + T_{qsort}(n-1)
 \end{aligned}$$



We can once again use master theorem, or a diagram such as below to see the complexity: Hence in the worst case  $T_{qsort}(n) \in O(n^2)$  (same as insertion sort).