

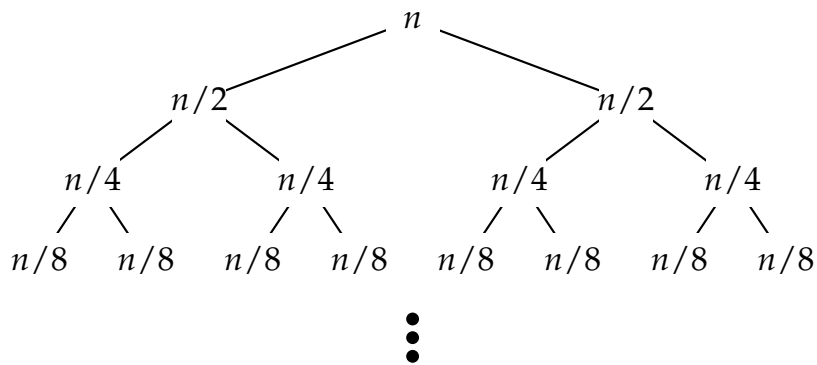
## Algorithms: Introduction to Divide & Conquer

### Model 1: Merge sort

```
mergesort(xs) =  
  if  $\text{len}(xs) \leq 1$  then return  $xs$   
  split  $xs$  into halves  $(xs_1, xs_2)$   
   $xs'_1 \leftarrow \text{mergesort}(xs_1)$   
   $xs'_2 \leftarrow \text{mergesort}(xs_2)$   
   $xs' \leftarrow \text{merge}(xs'_1, xs'_2)$   
  return  $xs'$ 
```

$$T(1) = \Theta(1)$$

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



Recall the *merge sort* algorithm, which works by splitting the input list into halves, recursively sorting the two halves, and then merging the two sorted halves back together.

- 1 How long does mergesort take on a list of length 1?
- 2 Just by looking at the code, how many recursive calls does merge-sort make at each step?
- 3 If  $xs$  has size  $n$ , what is the size of the inputs to the recursive calls to mergesort?

**Learning objective:** Students will use recurrence relations and recursion trees to describe and analyze divide and conquer algorithms.

*Hint:* don't overthink this one; yes, it's really that easy.

- 4 (Review) How long does it take (in big- $\Theta$  terms) to merge  $xs_1$  and  $xs_2$  after they are sorted?
  
- 5 Let  $T(n)$  denote the total amount of time taken by mergesort on an input list of length  $n$ . Use your answers to the previous questions to explain the equations for  $T(n)$  given in the model. This is called a *recurrence relation* because it defines  $T(n)$  via recursion.
  
- 6 Suppose algorithm  $X$  takes an input of size  $n$ , splits it into three equal-sized pieces, and makes a recursive call on each piece. Deciding how to split up the input into pieces takes  $\Theta(n^2)$  time; combining the results of the recursive calls takes additional  $\Theta(n)$  time. In the base case, algorithm  $X$  takes constant time on an input of size 1. Write a recurrence relation  $X(n)$  describing the time taken by algorithm  $X$ , similar to the one given in the model.
  
- 7 Now suppose algorithm  $X$  makes only two recursive calls instead of three, but each recursive call is still on an input one-third the size of the original input. How does your recurrence relation for  $X$  change?
  
- 8 Write a recurrence relation for binary search.

Now let's return to considering merge sort. The tree shown in the model represents the call tree of merge sort on an input of size  $n$ , that is, each node in the tree represents one recursive call to merge sort. The expression at each node shows how much work happens at that node (from merging).

- 9 Notice that the entire tree is not shown; the dots indicate that the tree continues further with the same pattern. What is the depth (number of levels) of the tree, in terms of  $n$ ?



- 10 How much total work happens on each individual level of the tree?
- 11 How much total work happens in the entire tree?
- 12 Draw a similar tree for the second version of algorithm X. Be careful to distinguish between the *size of the input* and the *amount of work* done at each node.

