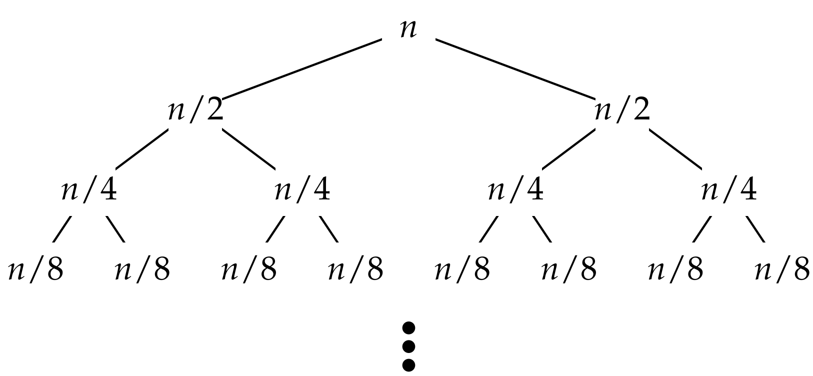
Algorithms Activity 8: Intro to Divide & Conquer Algorithms

Model 1: Merge sort

mergesort(xs) =
  if $len(xs) = 1$ **then return** xs
  split xs into halves (xs_1, xs_2)
  $xs_1 \leftarrow$ mergesort(xs_1)
  $xs_2 \leftarrow$ mergesort(xs_2)
  $xs' \leftarrow$ 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.

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

1 How many recursive calls does mergesort make? (*Hint*: don't overthink this one; yes, it's really that easy.)

2 If xs has size n, what is the size of the inputs to the recursive calls to mergesort?

3 How long does it take to merge xs_1 and xs_2 after they are sorted?

4 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.

5 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.

6 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?

7 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).

8 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?

9 How much total work happens on each individual level of the

tree?

10 How much total work happens in the entire tree?

11 Draw a similar tree for the second version of algorithm X.

Model 2: Arithmetic

$$
\begin{array}{rcccccccc}
  &   & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
+ &   & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
  & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1
\end{array}
$$

$$
\begin{array}{ccccccccccccccc}
 & & & & & & & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
 & & & & & & \times & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
 & & & & & & & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
 & & & & & & 1 & 0 & 1 & 1 & 0 & 1 & 0 & \\
 & & & 1 & 0 & 1 & 1 & 0 & 1 & 0 & & & & \\
 & & 1 & 0 & 1 & 1 & 0 & 1 & 0 & & & & & \\
+ & 1 & 0 & 1 & 1 & 0 & 1 & 0 & & & & & & \\
\hline
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0
\end{array}
$$

$$X = 01101001_2 \qquad X_1 = 0110_2 \qquad X_2 = 1001_2$$
$$Y = 11100100_2 \qquad Y_1 = 1110_2 \qquad Y_2 = 0100_2$$

In many situations, when we use arithmetic operations on integers such as addition or multiplication, we simply assume they take constant time. This is appropriate when the integers are bounded by some fixed size, *e.g.* if all the integers are 64-bit integers. However, if we want to be able to deal with integers of arbitrary size, this is no longer appropriate; we must take the size of the integers into account

Learning objective: Students will analyze arithmetic operations on n-bit integers.

12 Approximately how many single bit operations (*e.g.* adding or comparing two bits) are needed to compute the addition $1011010_2 + 1110011_2$ shown in the model?

13 In general, suppose we want to add two integers of n bits each. In terms of Θ, how many bit operations are needed?

14 Explain why it is not possible to do any better than this.

15 If we have a list of $\Theta(n)n$ integers, each with $\Theta(n)$ bits, how long will it take (in terms of single bit operations) to add all of them?

Now consider the multiplication shown in the model.

16 Why are there five rows in between the two horizontal lines?

17 How many operations are needed to produce each such row? (*Hint*: you may assume that multiplying by a power of two takes constant time.)

18 If we are multiplying two integers with n bits each, how many intermediate rows could there be in the worst case? In the average case?

19 How long will it take to add them all?

Let's now consider whether it is possible to multiply two n-bit integers any faster.

20 What is the relationship between X, X_1 and X_2 in the model? What about Y, Y_1, and Y_2?

21 Suppose $Z = 1011100101_2$. What are Z_1 and Z_2?

22 What is $2^4 \cdot X_1$ in binary?

23 Write an equation expressing X in terms of X_1 and X_2, and another expressing Y in terms of Y_1 and Y_2.

24 In general, suppose A is an n-bit integer, and we split it into A_1 and A_2. Generalize your previous answer to express A in terms of A_1 and A_2.

25 Suppose A and B are n-bit integers, and consider the product AB. Expand both A and B using your previous answer, and distribute the resulting product. You should end up with an expression involving only A_1, A_2, B_1, and B_2.

26 How many multiplications are required to compute the expression from Question 25? (Remember that multiplying by a power of two takes constant time and does not need to be counted.)

27 How big (how many bits) are the inputs to each multiplication in Question 25?

28 Explain how we can use the equation from Question 25 as a recursive algorithm to compute AB.

29 Let $M(n)$ denote the time taken to multiply two n-bit integers, and write a recurrence relation for $M(n)$ corresponding to this recursive algorithm.