1 Divide and Conquer

The idea behind divide and conquer is as follows:

1. Break problems into similar pieces.
2. Solve pirces recursively.
3. Recombine the pieces to get an answer.

1.1 Integer Multiplication

Suppose we're given two n-bit numbers and are asked to find their product.

1.1.1 Naive Algorithm

The naive algorithm is to do multiplication like we would from elementary school. This runs in $\mathcal{O}(n^2)$ time because we need to write down $\mathcal{O}(n^2)$ bits of numbers to add (addition is done in linear time and is omitted).

1.1.2 Improving the Algorithm: Two-Digit Multiplication

If we multiplied $ab \times cd$ digit-wise, we would get:

$$ab \times cd = (ac)(bc + ad)(bd)$$

This requires 4 one-digit multiplications and one addition. The **trick** is to compute ac, bd, $(a+b)(c+d)$. We note that:

$$bc + ad = (a+b)(c+d) - ac - bd$$

This requires *3* one-digit multiplications and 4 addition/subtractions.

1.2 Generalization

We will often get runtime recurrences with divide and conquer looking something like:

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

Here, the second line is saying a subproblems of size $\frac{n}{b}$.

1.2.1 Tracking Recursive Calls

We have:

- 1 recurisve calls of size n
- a recursive calls of size $n/b + O(1)$
- a^2 recursive calls of size $n/b^2 + O(1)$
- ...
- a^k recursive calls of size $n/b^k + O(1)$

So, the total runtime is:

$$\begin{aligned} \text{Total Runtime} &= \sum_{k=0}^{\log_b(n)} a^k O\left(\left(\frac{n}{b^k}\right)^d\right) \\ &= O(n^d) \sum_{k=0}^{\log)b(n)} \left(\frac{a}{b^d}\right)^k \end{aligned}$$

There are several cases to consider.

1. $a > b^d$: The runtime would be $O(n^{\log)b(a)})$.
2. $a < b^d$: The runtime is $O(n^d)$.
3. $a = b^d$: The runtime is $O(n^d \log(n))$.

1.2.2 Master Theorem

Theorem 1.1: Master Theorem

Let $T(n)$ be given by the recurrence:

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

Then we have:

$$T(n) = \begin{cases} O(n^{\log_b(a)}) & a > b^d \\ O(n^d \log(n)) & a = b^d \\ O(n^d) & a < b^d \end{cases}$$