Division[1]

1 Division

DIVISION
Input: n-bit number x, m-bit number y expressed as bit-arrays $x[0:n-1], y[0:m-1]$.
Output: The quotient-remainder pair (q,r) such that $x = qy + r$ where $r < y$.
Size: The number of bits $n+m$.

Our final course of the day is integer division. We want to take input two numbers x, y, and return the quotient and remainder obtained when x is divided by y. That is, we want to find non-negative integers (q,r) such that $x = qy + r$ and $r < y$.

Once again, we define a recursive algorithm to do the same. First we identify the base cases. If $x < y$, then we know that the quotient is 0 and remainder is x. If $x = y$, then the quotient is 1 and remainder is 0. Now suppose $x > y$.

Case 1. $x = 2k$ is even. Then if (q', r') is what we obtain *recursively* when we divide the smaller number k by y, that is, $k = q'y + r'$, then $x = 2q' \cdot y + 2r'$. Therefore, we should return $(2q', 2r')$, except $2r'$ may be bigger than y. In which case, we should return $(2q'+1, 2r'-y)$. This suffices since $r' < y$ and so $2r' < 2y$ and so $2r' - y < y$. That is, $2r' - y$ is the correct remainder.

Case 2. $x = 2k+1$ is odd. Again, suppose (q', r') is obtained *recursively* when we divide k by y. Then we get $x = 2k + 1 = 2q'y + 2r' + 1$. Now, since $r' < y$, that is, $r' \leq y - 1$, we get that $2r' \leq 2(y-1) = 2y' - 2$. Therefore, $2r' + 1 \leq 2y - 1 < 2y$. Which, in turn, implies $2r' - y < y$. That is, $2r' - y$ is the correct remainder.

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 19th Mar, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

1: **procedure** DIVIDE(x, y):
2: ▷ *The two numbers are input as bit-arrays; x has n bits, y has m bits. $n \geq m$.*
3: ▷ *Returns (q, r) where $x = qy + r$ and $0 \leq r < y$.*
4: **if** $x < y$ **then**:
5: **return** $(0, x)$
6: **if** $x = y$ **then**:
7: **return** $(1, 0)$
8: $x' \leftarrow \lfloor x/2 \rfloor$ ▷ *Obtained by right shifts*
9: $(q', r') \leftarrow$DIVIDE(x', y)
10: $q \leftarrow 2q'$; $r \leftarrow 2r'$ ▷ *Obtained by left shifts*
11: **if** x is odd **then**:
12: $r \leftarrow r + 1$ ▷ *Obtained by* ADD$(r, 1)$.
13: **if** $r \geq y$ **then**:
14: $q \leftarrow q + 1$ ▷ *Obtained by* ADD$(q, 1)$.
15: $r \leftarrow r - y$ ▷ *Subtraction is just addition with the "complement"*
16: **return** (q, r).

Once again, let us work to figure out the recurrence inequality for the running time. Let $T(n, m)$ be the time taken to divide an n-bit number by an m-bit number.

- Let's first understand the base cases. Line 5 and Line 7 take $O(1)$ time. Thus, we get $T(n, m) = O(1)$ if $n < m$. Note, we cannot say $n = m$ for x and y can both be m bits big and yet $x > y$.
- Line 8 and Line 10, as in the case of MULT, take $O(1)$ time as well.
- The recursive call in Line 9 takes at most $T(n-1, m)$ time. This is because x' has $n-1$ bits, and the definition of worst-case runtime.
- Now consider Line 12 and Line 14. Note that in both cases we are adding 1 to an even number (see Line 10), that is, a number whose last bit is 0. Thus, one needs only one BIT-ADD to increment an even number by one. Thus, these steps cost $O(1)$ time.
- Line 15 is the "time-taking" step of *subtraction*. How does one subtract? As you can see in the supplement, subtraction is simply an addition[2] with "complementing", the time (or number of BIT-ADDS) is the same as to add. And thus, since both y and r have $\leq (m+1)$ bits (note $r \leq 2y$), this step takes $O(m)$ time.

Therefore, we get the following recurrence for DIVIDE.

$$\begin{aligned} T(n, m) &= O(1) \quad \text{if } n < m \\ T(n, m) &\leq T(n-1, m) + O(m), \quad \text{if } n \geq m \end{aligned} \tag{1}$$

[2] If you have never seen this before, then I recommend going and reading this in the supplement.

Theorem 1. DIVIDE takes $T(n, m) = O(m \cdot (n - m + 1))$ time (i.e. BIT-ADDs/elementary operations) to divide an n-bit number by an m-bit number where $n \geq m$.

Proof. Once again, this can be solved by the kitty method as follows. Again, we may assume there is a large enough constant C such that $T(n, m) \leq T(n-1, m) + Cm$ if $n \geq m$.

$$\begin{aligned} T(n, m) &\leq T(n-1, m) + Cm \\ &\leq T(n-2, m) + Cm + Cm \\ &\vdots \\ &\leq T(m-1, m) + Cm \cdot (n - m + 1) \end{aligned}$$

The proof completes by noting $T(m-1, m) = O(1)$. $\square$

Corollary 1. If $n = m + c$, that is, x has only c more bits than y and c is some fixed constant (like 25), then DIVIDE takes $O(n)$ time.