CSC373— Algorithm Design, Analysis, and Complexity

Divide and Conquer, Worked Example: Mod of Powers

Mod of Powers. For integers y and b, with $b > 0$, we define the operation $y \mod b$ as: $z = y \mod b$ if and only if $z = y - bj$ where j is the maximum integer such that $bj \le y$. So, in particular, $0 \le y \mod b < b$.

Suppose we have a somewhat limited built-in mod function that runs in constant time, but it does not apply to really large input integers. In particular, assume that $x \mod b$ can only be computed with the built-in function for integers x with $|x| \le Y$ for some positive constant Y. We assume y and b^2 are less than Y.

(a) We wish to compute $a = y^n \mod b$ for $n = 2^k$ where $k \ge 0$ an integer. You can assume both y and b are positive integers. Use a divide and conquer approach to compute a. Your algorithm must run in $\Theta(\log(n))$ time. (Take careful note of the restriction on the built-in mod function described above.) Clearly explain your algorithm and show how you derived its runtime estimate.

(b) Modify your algorithm in part (a) to work for any integer $n > 0$, not just powers of two. The runtime should still be $\Theta(\log(n))$. Clearly explain your algorithm and show how you derived its runtime estimate. (The limitation on the built-in mod function still applies.)

Sample Solution: The key property of the mod operator that we need to use is as follows:

Property 1. For any two integers $u, v > 0$, we have $(uv \bmod b) = ([(u \bmod b)(v \bmod b)] \bmod b)$.

The proof of this is not required, although we sketch it below.

To prove Property 1, let j be the max integer such that $j \le u/b$, and k be the max integer such that $k \le v/b$. Then, by the definition of mod, $u = jb + (u \bmod b)$ and similarly $v = kb + (v \bmod b)$. Therefore

$$\begin{aligned}(uv \mod b) &= ((jb + (u \mod b))(kb + (v \mod b))) \mod b \\ &= ([mb + (u \mod b)(v \mod b)] \mod b), \quad \text{for some integer } m, \\ &= [(u \mod b)(v \mod b)] \mod b\end{aligned}$$

Here, on the right hand side of the first equation above, three of the terms in the product are integer multiples of b. These terms are collected into the term mb in the second line. The last equation above uses the fact that, for any integer k, $((x + kb) \bmod b) = (x \bmod b)$.

(a) See part b below. The same algorithm and runtime analysis applies.

(b) We define a function powerMod(y, n, d) to return the appropriate value. The key idea is to recursively call the function powerMod(y, m, d) **only once** per recursive step. We can do this by writing $y^n = y^m y^m y^\delta$, where $m = \text{floor}(n/2)$ and $\delta = 0$ or 1. We can then apply Property 1, making sure to do only one recursive call for (y^m mod b). See the algorithm below.

```
r = powerMod( y, n, b)
    % Precondition: y, n, b postive integers, with y and b^2 ≤ Y,
    % where Y is the max allowable argument of Y mod b.

    if n == 1
        return (y mod b)

    m = floor(n/2)                      % So n − 1 ≤ m + m ≤ n
    r = powerMod(y, m, b)
    r = ((r * r) mod b)                 % Now r = ((y^{2m}) mod b) by Property 1.
    if 2 * m < n                        % In this case y^n = y^{2m} y.
        r = (r * (y mod b)) mod b       % So r = (y^{2m})(y) mod b by Property 1.

    return r
```

The recurrence relation for the runtime of this algorithm is $T(n) = T(n/2) + \Theta(1)$. Therefore, the Master Theorem applies with $a = 1$, $b = 2$ and $d = 0$. In this case we have $a = b^d = 1$, so the theorem ensures $T(n) = \Theta(\log n)$.