CS4102 - ALGORITHMS - SUMMER'21 — NATE BRUNELLE
DUE THURSDAY MAY 27, 2021 AT 11:59PM EASTERN

Little1c.Solution

PROBLEM 1 *Fast Exponentiation*

Given a pair of positive integers (a, n), devise a divide and conquer algorithm that computes a^n using only $O(\log n)$ calls to a multiplication routine. You need to show that the algorithm is correct (i.e. it always produces the right answer) and also that it only uses $O(\log n)$ multiplications.

Divide Step. For the input a, n we divide n by 2 to produce the one subproblem $(a, \lfloor \frac{n}{2} \rfloor)$. This has constant running time (with either 0 or 1 multiplication, depending on if you consider division to be a special case of multiplication, overall your asymptotic analysis will not be impacted by this decision).

Combine Step. Once we have the answer for $x = (a, \lfloor \frac{n}{2} \rfloor)$ then we return $x \cdot x$ when n is even, or $a \cdot x \cdot x$ When n is odd. This has constant running time (at most 2 multiplications).

Algorithm. For input (a, n), if $n = 1$ then return a. If $n \geq 1$ then recursively solve $(a, \lfloor \frac{n}{2} \rfloor)$ to get $x = a^{\lfloor \frac{n}{2} \rfloor}$. If n is even then return $x \cdot x$, otherwise if n is odd then return $a \cdot x \cdot x$.

Correctness. Take any input a. We use induction on the exponent n. The base case where $n = 1$ is immediate. We show that the algorithm is correct on input n (assuming it is correct for all exponents less than n). By the inductive hypothesis, the conquer step gives the value of $a^{\lfloor \frac{n}{2} \rfloor}$, then the combine step gives a^n since if n is even $\lfloor \frac{n}{2} \rfloor = \frac{n}{2}$, and so $a^{\lfloor \frac{n}{2} \rfloor} \cdot a^{\lfloor \frac{n}{2} \rfloor} = a$. If n is odd then $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{2} \rfloor = n - 1$ and so $a \cdot a^{\lfloor \frac{n}{2} \rfloor} \cdot a^{\lfloor \frac{n}{2} \rfloor} = a$

Running time. Let $T(n)$ be the number of multiplications done by this routine for the exponent n. This routine makes a recursive call on a new problem of at most half the size (i.e., the exponent will be at most $\frac{n}{2}$. The combine step requires at most 2 multiplications (when n is odd), therefore the total number of multiplications performed is given by the recurrence $T(n) = T(\frac{n}{2}) + 2$. To solve this recurrence we can use the Master Theorem with $a = 1$, $b = 2$, and $f(n) = 2$, thus $n^{\log_b a} = n^0 = 1$, so $f(n) = \Theta(n^{\log_b a})$, so Case 2 applies, giving the run time to be $\Theta(\log n)$