The number of function calls in the fast exponentiation algorithm

Consider the fast exponentiation algorithm

```plaintext
function Power(b, n)
    if n == 0 then
        return 1
    end if
    if n is divisible by 2 then
        return (Power(b, n/2))^2
    end if
    return b(Power(b, (n - 1)/2))^2
end function
```

**Proposition.** The number of calls of Power function during the computation of Power(b, n) is equal to \([\log_2 n] + 2\) for every positive integer \(n\).

**Proof.** We will prove the statement by induction on \(n\). Base case is \(n = 1\), and direct computation shows that there will be exactly two calls. Since \(\log_2 1 + 2 = 2\), the base case is proved.

Hypothesis: assume that for every positive integer \(k < n\) the statement is proved.

Step: we will prove the statement for \(n\). Consider two cases:

1. \(n\) is even. Then inside Power(b, n) the next call will be Power(b, n/2). Due to the induction hypothesis, there will be \([\log_2 (n/2)] + 2\) calls for Power(b, n/2), so in total there will be

   \[1 + [\log_2 (n/2)] + 2 = [1 + \log_2 (n/2)] + 2 = [\log_2 n] + 2\]

   calls.

2. \(n\) is odd. In this case the next call will be Power(b, (n - 1)/2). Due to the induction hypothesis, it will yield \([\log_2 ((n - 1)/2)] + 2\), so the total number of calls will be

   \[1 + [\log_2 ((n - 1)/2)] + 2 = [1 + \log_2 ((n - 1)/2)] + 2 = [\log_2 (n - 1)] + 2.\]

We will prove that \([\log_2 (n - 1)] = [\log_2 n]\), and this will finish the inductive step. Let \(k = [\log_2 (n - 1)]\), then \(2^k \leq n - 1 < 2^{k+1}\). Since \(n - 1\) and \(2^{k+1}\) are both even, difference between them is at least 2, so \(2^k < n < 2^{k+1}\). But the latter inequality means that \(k = [\log_2 n]\). This finishes the proof.

\(\square\)