The number of function calls in the fast exponentiation algorithm

Consider the fast exponentiation algorithm

function \( \text{Exp}(a, n) \)
    if \( n == 0 \) then
        return 1
    end if
    if \( n \) is divisible by 2 then
        return \( (\text{Exp}(a, n/2))^2 \)
    end if
    return \( a(\text{Exp}(a, (n - 1)/2))^2 \)
end function

**Proposition.** The number of calls of \( \text{Exp} \) function during the computation of \( \text{Exp}(a, n) \) is equal to \( \lceil \log_2(n + 1) \rceil + 1 \) for every integer \( n \geq 0 \).

Before we dive into the formal proof, let us discuss informally (this can be turned into a rigorous argument as well as was discussed in the lecture) how one could come up with such a formula. We start with an experiment and compute the number of calls for relatively small \( n \)'s.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

We make an observation that we have one 2, two 3’s, four 4’s, eight 5’s and so on. This can be explained by noticing that each number \( k \) in the second row corresponding to number \( n \) in the first row gives rise to two \( k + 1 \)'s corresponding to \( 2n \) and \( 2n + 1 \). Then the last appearance of \( k \) in the second row will occur after all the appearances of the numbers less than or equal to \( k \), so it will correspond to

\[
    n = 1 + 2 + 4 + \ldots + 2^{k-2} = 2^{k-1} - 1.
\]

Thus, \( \text{Exp}(a, n) \) invokes exactly \( k \) calls of \( \text{Exp} \) if and only if

\[
    2^{k-2} - 1 < n \leq 2^{k-1} - 1.
\]

Adding one to both sides and taking log, we obtain

\[
    k - 2 < \log_2(n + 1) \leq k - 1.
\]

Manipulating with the cailing function, one can now show that

\[
    k = 1 + \lceil \log_2(n + 1) \rceil.
\]

The same formula can be proved formally by induction as follows.

**Proof of the proposition.** We will prove the statement by induction on \( n \).

**Base** The base case is \( n = 0 \), there will be only one call. Since \( \log_2 1 + 1 = 1 \), the base case is proved.

**Hypothesis** Assume that for nonnegative positive integer \( k < n \) the statement is proved.
**Step**  We will prove the statement for \( n \). Consider two cases:

1. \( n \) is odd. In this case the next call will be \( \text{Exp}(a, (n-1)/2) \). Due to the induction hypothesis, it will yield \( \lceil \log_2((n-1)/2 + 1) \rceil + 1 \) calls, so the total number of calls will be

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

2. \( n \) is even. Then inside \( \text{Exp}(a, n) \) the next call will be \( \text{Exp}(a, n/2) \). Due to the induction hypothesis, there will be \( \lceil \log_2(n/2 + 1) \rceil + 1 \) calls for \( \text{Exp}(a, n/2) \), so in total there will be

\[
1 + \lceil \log_2(n/2 + 1) \rceil + 1 = \lceil 1 + \log_2(n/2 + 1) \rceil + 1 = \lceil \log_2(n+2) \rceil + 1
\]
calls.

We will prove that \( \lceil \log_2(n+2) \rceil = \lceil \log_2(n+1) \rceil \), and this will finish the inductive step. Let \( k = \lceil \log_2(n+1) \rceil \), then \( 2^{k-1} < n+1 \leq 2^k \). Since \( n+1 \) and \( 2^k \) are of different parity, difference between them is at least 1, the latter inequality is actually strict. Thus, \( 2^{k-1} < n+1 < 2^k \), so we have \( 2^{k-1} < n+2 \leq 2^k \). This implies that \( k = \lceil \log_2(n+2) \rceil \). The proof is finished.

\[\square\]