CISC320 Algorithms, Spring 2010
Modexp() and Addition Chains

function modexp(a, e, N)
Input: n-bit positive integers a, e, N.
Output: a^e mod N.
Basic idea: repeated squaring
if $e = 1$, return a
$b = \text{modexp}(a, \lfloor e/2 \rfloor, n)$
$b = b * b \bmod N$
if e is odd,
    $b = b * a \bmod N$
return b

By definition, an n-bit number, e, has binary expansion $e = \sum_{i=0}^{n-1} e_i 2^i$, where each e_i is a bit with value 0 or 1. Let us say that the length of a number e is the minimum n such that e is a n-bit number. In other words, if k is the largest index such that the e_k bit in the binary expansion of e is a 1, then the length of e is $k + 1$. Put yet another way, the length of e, len(e), is the least exponent m such that $e \leq 2^m$.

We have observed that if len(e) is m, then the number of squarings mod N is $m - 1$ and the number of multiplications $b * a$ in the else clause is $k - 1$, where k is the number of 1-bits in the binary expansion of e. Thus we know that the number of multiplications modulo N done in `modexp()` is O(n) and more precisely, for len(e) $= m$, is at most $2m - 2$ and at least $m - 1$. Question: Is `modexp` optimal? That is, might there be an algorithm using fewer multiplications for some exponent e?

Consider the sequence of powers of a computed as successive values of b in the algorithm[1]. For instance, when $b = 11 = 1011_2$, the sequence of values of b is $(a^1, a^2, a^4, a^5, a^{10}, a^{11})$. For short, let's list just the sequence of exponents of a, $(1, 2, 4, 5, 10, 11)$. Note that each entry in th sequence is either double a previous entry or one more than a previous value. More generally, an *addition chain for* e is a sequence of integers such that the first is 1, and each succeeding entry is either the sum of two previous or double a previous one, and the last entry is e. So our optimality question can be converted to this: "is there an exponent e which has a shorter addition chain for it than the one generated by `modexp`?"

The answer is yes, and one example is $e = 31$. Our `modexp` builds the length 9 addition chain (1, 2, 3, 6, 7, 14, 15, 30, 31). But e also has the chain (1, 2, 3, 6, 12, 24, 30, 31) of length 8.

Addition chains have been studied at considerable length, but no systematic patterns have emerged of general use for algorithm design yielding anything that is a real improvement over `modexp`. And of course, `modexp` is optimal up to big-O. This is basically because no addition chain can have an i-th entry larger than 2^i.

Postscript: There is no shorter addition chain for $e = 11$ than the one produced by `modexp`.

[1]Strictly speaking, b is a local variable in each each recursive call to modexp, but since each local b is first set as the result of a call and is the returned value of that call, there is no ambiguity in considering all of the local b's as being one variable.