method of repeated squaring

Canonical name	MethodOfRepeatedSquaring
Date of creation	2013-03-22 16:16:14
Last modified on	2013-03-22 16:16:14
Owner	Algeboy (12884)
Last modified by	Algeboy (12884)
Numerical id	9
Author	Algeboy (12884)
Entry type	Theorem
Classification	msc 20A05
Classification	msc 08A99
Classification	msc 20-00
Synonym	successive squaring

Theorem 1. *Given a semigroup S and an $n \in \mathbb{Z}^+$ then the function $f : S \to S$ defined by $f(a) = a^n$ has an SLP representations of computational length $O(\log_2 n)$. Inparticular, if we can compute a^n using only $O(\log_2 n)$ multiplications.*

Proof. Let $f_0 : S \to S$ be the map $f(x) = x$ and $f_1 : S \to S$ be defined by $f_1(x) = xx$. So far we recognize that $f_0(x)$ evaluates to $x^1 = x^{2^0}$ and $f_1(x)$ evaluates to $x^2 = x^{2^1}$. *But we caution that we do not define these functions with exponents because we want to demonstrate that from an algorithm to multiply we can create an efficient algorithm to exponentiate.*

For $1 < i \leq k$ define (recalling that f_{i+1} may use the output of any previous f_i evaluation)

$$f_{i+1}(x) := f_1(f_i(x)).$$

Evidently, $f_{i+1}(x)$ evaluates to $f_i(x)^2$. By induction, $f_i(x)$ evaluates to x^{2^i} and thus $f_{i+1}(x)$ evaluates to $(x^{2^i})^2 = x^{2^{i+1}}$. Therefore we establish that $f_j(x) = x^{2^j}$ for all nonnegative integers j. Furthermore, to evaluate $f_j(x)$ uses j multiplications.

Now write n in binary: $n = a_0 2^0 + a_1 2^1 + \cdots + a_k 2^k$ with $a_i = 0, 1$ for $0 \leq i < k \leq \log_2 n$. Let $0 \leq i_1 < i_2 < \cdots < i_s \leq k$ be the indices for which $a_{i_t} \neq 0$, $1 \leq t \leq s$. Then define

$$g_n(x) = f_{i_1}(x) f_{i_2}(x) \cdots f_{i_s}(x)$$

Thus, $g_n(x)$ evaluates to:

$$\begin{aligned}
(x^{2^{i_1}})(x^{2^{i_2}}) \cdots (x^{2^{i_s}}) &= (x^{2^0})^{a_0} (x^{2^1})^{a_1} \cdots (x^{2^k})^{a_k} \\
&= x^{a_0 2^0 + a_1 2^1 + \cdots + a_k 2^k} \\
&= x^n.
\end{aligned}$$

Because each $f_{i+1}(x)$ uses the output of $f_i(x)$, the cost of evaluating g_n is the cost of evaluating $f_{i_s}(x)$ plus the final cost of multiplying $f_{i_1}(x) \cdots f_{i_s}(x)$. Thus, evaluating g_n uses $i_s + (s-1)$ multiplications. As $s \leq \log_2 n$ it follows that evaluating g_n uses no more than $2 \log_2 n$ multiplications. $\square$