On Asymptotic Estimates for Arithmetic Cost Functions

Carlos Gustavo T. de A. Moreira

I M P A

Estrada Dona Castorina, 110

22460-320, Rio de Janeiro-RJ

Abstract: Let $\tau(n)$ (the cost of n) be the minimum number of arithmetic operations needed to obtain n starting from 1. We prove that $\tau(n) \geq \frac{\log n}{\log \log n}$ for almost all $n \in \mathbf{N}$, and, given $\varepsilon > 0$, $\tau(n) \leq \frac{(1+\varepsilon)\log n}{\log \log n}$ for all n sufficiently large. We prove analogous results for costs of polynomials with integer coefficients.

Introduction:

The aim of this work is the study of arithmetic cost functions. The most basic case is the cost of an integer, which is the minimum number of arithmetic operations needed to obtain the integer starting from 1. That is, given $k \in \mathcal{Z}$, $\tau(k)$ is defined as the minimum $m \in \mathbf{N}$ such that there exists a sequence $(s_0, s_1, \ldots, s_m)$ where $s_0 = 1$, $s_m = k$ and for each $\ell \geq 1$ there are i, j, $0 \leq i$, $j < \ell$ with $s_\ell = Op(s_i, s_j)$, $Op \in \{+, -, \cdot\}$. This function plays an important role in the paper by Shub and Smale ([SS]), in which some questions are proposed about the behavior of the function τ (which we shall discuss later) that have consequences in the study of an algebraic version of the famous and fundamental problem in computer science known as "$NP \neq P$" (see [SS]). In [SS] and in [MS] the following universal bounds for τ are presented:

$$\log_2 \log_2 m + 1 \leq \tau(m) \leq 2 \log_2 m, \quad \forall m \in \mathbf{N}.$$

The first inequality cannot be improved, since $\tau(2^{2^k}) = k + 1$. These inequalities are proved in the paper by de Melo and Svaiter ([MS]), where it is also proved that for any fixed $\varepsilon > 0$, we have $\tau(n) \geq \frac{\log n}{(\log \log n)^{1+\varepsilon}}$ for almost all $n \in \mathbf{N}$.

Here we improve the above results, by proving the following:

For almost all $n \in \mathbf{N}$ we have $\tau(n) \geq \frac{\log n}{\log \log n}$ and, for any given $\varepsilon > 0$, we have $\tau(n) \leq \frac{(1+\varepsilon) \log n}{\log \log n}$, for every $n \in \mathbf{N}$ sufficiently large. More precisely, given $\varepsilon > 0$,

$$\begin{cases} \tau(n) \geq \frac{\log n}{\log \log n} + (1 - \varepsilon) \frac{\log n \log \log \log n}{(\log \log n)^2}, & \text{for almost all } n \in \mathbf{N} \\ \tau(n) \leq \frac{\log n}{\log \log n} + (3 + \varepsilon) \frac{\log n \log \log \log n}{(\log \log n)^2}, & \text{for } n \text{ large enough.} \end{cases}$$

Moreover, the first inequality does not depend on the number of binary operations that we can use (provided that this number is finite).

Subsequently, we extend our results for more general cost functions, such as cost functions of polynomials, and discuss some open problems.

Section 1: The lower estimates.

Theorem 1. *Suppose that we have O binary or unary operations in the definition of τ. Then $N(k) = \#\{n \in \mathbf{N} \mid \tau(n) \leq k\}$ satisfies $N(k) \leq A^k \cdot k^k$ for some constant $A > 0$.*

Proof. Let $\Omega = \{Op_1, Op_2, \ldots, Op_O\}$ the set of operations. Then $\tau(n) = k \Rightarrow \exists (s_0, s_1, \ldots, s_k)$; $s_0 = 1$, and for each $\ell \geq 1$ $\exists i_\ell, j_\ell$, $0 \leq i,\ j < \ell$ such that $s_\ell = Op(s_{i_\ell}, s_{j_\ell})$, $Op \in \Omega$, with

$s_k = n$. Let $(r_1, r_2, \ldots, r_{2k}) := (i_1, j_1, i_2, j_2, \ldots, i_k, j_k)$. We must have $\{i_1, j_1, \ldots, i_k, j_k\} = \{0, 1, 2, \ldots, k-1\}$, since otherwise we would create an unnecessary s_i, and so $\tau(n) < k$. Moreover, we may assume that there exists a sequence $1 \le \ell_1 < \ell_2 < \cdots < \ell_k \le 2k$ such that $r_{\ell_i} = i - 1$, for $i = 1, 2, \ldots, k$. Indeed, we may assume $i_\ell \le j_\ell$, for $\ell = 1, 2, \ldots, k$ (perhaps by creating the new operations $Op_j^*(x, y) = Op_j(y, x)$), and, if $P(i) := \min\{j \in \{1, 2, \ldots, 2k\} \mid r_j = i\}$, for $i = 0, 1, \ldots, k-1$, we shall have, perhaps rearranging the order of the operations, that $P(0) < P(1) < \cdots < P(k-1)$, because if $P(i) > P(i+1)$, then we do not use s_i to create s_{i+1}, and can therefore create s_{i+1} before s_i. Now we can take $\ell_i := P(i)$.

Let $N'(k) = \#\{n \in \mathbf{N} \mid \tau(n) = k\}$. By the above arguments, $N'(k) \le O^k \cdot N''(k)$. where $N''(k) = \#\{(r_1, r_2, \ldots, r_{2k}) \mid r_i \in \{0, 1, \ldots, k-1\}$. There are $1 \le \ell_1, < \ell_2 < \cdots < \ell_k \le 2k$ with $r_{\ell_i} = i - 1$, for $i = 1, 2, \ldots, k\}$ but then $N''(k) \le C_{2k}^k$. $k^k \le 4^k \cdot k^k$ thus $N'(k) \le (4O)^k \cdot k^k$, and so $N(k) \le \displaystyle\sum_{r=0}^{k} N'(r) \le A^k \cdot k^k$, for $A = 4O + 1$. $\square$

Corollary. *Given $\varepsilon > 0$, we have, for almost all $n \in \mathbf{N}$, that*

$$\tau(n) \ge \frac{\log n}{\log\log n} + (1-\varepsilon)\frac{\log n \log\log\log n}{(\log\log n)^2} =: f(n)$$

Proof. Let us compute $B(n) = \#\{k \le n \mid \tau(k) \le f(k)\}$. If $k \in B(n)$ then $\tau(k) \le f(k) \le f(n)$, and we have at most $N(f(n))$ numbers with this property, that is, at most $A^{f(n)} \cdot f(n)^{f(n)}$ numbers with this property, by the theorem, but now we have, for n large enough, $A^{f(n)} f(n)^{f(n)} = \exp(f(n)\log(Af(n))) \le \exp\left(f(n)\log\left(\frac{\log n}{(\log\log n)^{1-\frac{\varepsilon}{2}}}\right)\right) = \exp(\log n\left(1 + (1-\varepsilon)\frac{\log\log\log n}{\log\log n}\right)\left(1 - (1-\frac{\varepsilon}{2})\frac{\log\log\log n}{\log\log n}\right) \le n\exp\left(-\frac{\varepsilon}{2}\frac{\log n \log\log\log n}{\log\log n}\right)$, therefore $\lim_{n\to\infty} \frac{B(n)}{n} = 0$. $\square$

Remark. If we have p-ary operations instead of binary operations ($p \geq 2$), we have an analogous result by changing $N(k) \leq A^k \cdot k^k$ into $N(k) \leq A^k k^{(p-1)k}$, in the theorem and $f(n)$ into $\frac{f(n)}{p-1}$ in the Corollary.

Section 2: The upper estimates

Theorem 2. *Given $\varepsilon > 0$, we have, for n large enough*

$$\tau(n) \leq \frac{\log n}{\log\log n} + (3+\varepsilon)\frac{\log n \log\log\log n}{(\log\log n)^2} =: g(n)$$

Proof. Let $B = [\frac{\log n}{(\log\log n)^3}]$, $C = B^k$, where $k = [\log\log n]$ (in the calculations we shall omit the integer parts). Take

$s_0 = 1, s_1 = 2, s_2 = 3, \ldots, s_{B-2} = B-1,$

$s_{B-1} = B, s_B = 2B, s_{B+1} = 3B, \ldots, s_{2B-3} = (B-1)B, \ldots,$

$s_{(k-1)(B-1)}B^{k-1}, s_{(k-1)(B-1)+1} = 2B^{k-1}, \ldots, s_{k(B-1)-1} = (B-1)B^{k-1}, s_{k(B-1)} = B^k = C.$

Now take the representation of n in basis C

$$n = a_0 + a_1 C + \cdots + a_r C^r, 0 \leq a_i < C, r = [\frac{\log n}{\log C}] \sim \frac{\log n}{(\log\log n)^2},$$

and the representations of the a_i in basis B:

$$a_i = b_{i1} + b_{i2}B + \cdots + b_{ik} \cdot B^{k-1}, 0 \leq b_{ij} < B.$$

Observe now that we have constructed the $b_{ij} \cdot B^{j-1}$, so we can construct an a_i doing $k-1$ sums. Since we have $r+1$ coefficients a_i we have a total time $(k-1)(r+1)$ to generate all the a_i's. But $(k-1)(r+1) \sim (\frac{\log n}{\log C})(\frac{\log C}{\log B}) + k - r \sim \frac{\log n}{\log B} + k - r$. Having generated the a_j, we do the following, in $2r$ steps:

$$a_r \to a_r \cdot C \to a_r \cdot C + a_{r-1} \to (a_r \cdot C + a_{r-1}) \cdot C \to \cdots \to a_r C^r + a_{r-1} C^{r-1} + \cdots + a_1 C + a_0 = n.$$

The total time is less than $k \cdot B + \frac{\log n}{\log B} + r \sim \frac{\log n}{(\log\log n)^3} \cdot \log\log n + \frac{\log n}{\log\log n - 3\log\log\log n} + \frac{\log n}{(\log\log n)^2} = \frac{\log n}{\log\log n} + \frac{3\log n \log\log\log n}{(\log\log n)^2} + O(\frac{\log n}{(\log\log n)^2}) \le g(n)$. $\square$

Remark. Using essentially the same proof we can replace $g(n)$ by $\frac{g(n)}{p-1}$ if we have the binary product and the p-ary sum $s(x_1, x_2, \ldots, x_p) = x_1 + x_2 + \cdots + x_p$.

Section 3: The cost of Polynomials

We shall extend the previous results to polynomials in several variables whose coefficients are integers or natural numbers.

Let $P(X_1, X_2, \ldots, X_k)$ be a polynomial. We define $\tau(P)$ as the minimum $m \in \mathbf{N}$ such that there exists a sequence $(s_{-k}, s_{1-k}, \ldots, s_{-1}, s_0, s_1, \ldots, s_m)$, where $s_{-i} = X_i$, $i = 1, 2, \ldots, k$; $s_0 = 1$ and $s_m = P$, such that for each $\ell \ge 1$ there are $i, j < \ell$ with $s_\ell = Op(s_i, s_j)$, $Op \in \{+, -, 0\}$.

Theorem 3. *Let P be a polynomial whose coefficients are natural numbers.*

Let d_i be the degree of P in the variable X_i, and let

$$h(P) = (1+d_1)(1+d_2)\ldots(1+d_k)\max_{(I)} \log(1 + a_{(I)}),$$

where $a_{(I)}$ *are the coefficients of* P. *Then, we have the following results:*

i) $\tau(P) \geq \frac{h(P)}{\log h(P)}$ *for almost all* P *(in the sense that the ratio of the number of polynomials that satisfy this property to the number of polynomials with* $h(P) \leq M$ *tends to* 1 *as* M *tends to infinity);*

ii) For any given $\varepsilon > 0$, $\tau(P) \leq \frac{(1+\varepsilon)h(P)}{\log h(P)}$, *provided* $h(P)$ *is large enough.*

Remark 1. If we consider polynomials with integer coefficients we still have the above, results provided we define

$$h(P) = (1+d_1)(1+d_2)\ldots(1+d_k)\max_{(I)}\log(1+2\left|a_{(I)}\right|).$$

Remark 2. In the proof below we work with $\log = \log_2$.

Proof. i) Given $d_1, d_2, \ldots, d_k$ and $M \in \mathbf{N}$, there are $N = (1+M)^{(1+d_1)(1+d_2)(1+d_k)}$ polynomials with degrees bounded by the d_i's and with $\max_{(I)} a_{(I)} \leq M$ (the $a_{(I)}$'s are natural numbers). Since $\frac{\log N}{\log\log N} = \frac{h}{\log h}$, where $h = (1+d_1)\ldots(1+d_k)\log(M+1)$, the result will follow exactly as in Theorem 1, Corollary.

ii) Let $B = \left[\frac{h(P)}{(\log h(P))^3}\right]$, $C = B^{[\log h(P)]}$. At least one of the two following cases must occur:

ii.1) $\max_{(I)} a_{(I)} \geq C^{[\log h(P)]}$. In this case we proceed as in the Theorem 2. We construct $1, 2, \ldots, B-1, B, 2B, \ldots, (B-1)B, \ldots, (B-1)B^{[\log h(p)]}, C$ and then construct all the $a_{(I)}$'s using basis C, writing the obtained coefficients in basis B. After this, we obtain the polynomial term by term. The total cost of this is bounded by $B\frac{\log C}{\log B} + \left(1 + \frac{\log M}{\log B} + \frac{\log M}{\log C} + \frac{\log C}{\log B}\right) D + 2D$, where $M = \max_{(I)}(1 + a_{(I)})$ and

$D = (1 + d_1)(1 + d_2) \ldots (1 + d_k)$, and so is bounded by $(1 + \frac{\varepsilon}{2}) \frac{D \log M}{\log B} < (1 + \varepsilon) \frac{h(P)}{\log h(P)}$, for $h(P)$ sufficiently large.

ii.2) $2^{(1+d_i)} \geq C^{[\log h(P)]}$ for some i, say for $i = 1$. Let $M = \max_{(I)}(1 + a_{(I)})$. We have two subcases:

ii.2.1) $M \geq 2^{\sqrt{\log h(P)}}$: In this case we construct all the polynomials $\sum_{j=0}^{r-1} \delta_j X_1^j$, with $\delta_j \in \{0, 1\}$, $\forall j$ and $r = [\log B] + 1$, forming the set β and then we construct all the polynomials in the set $\Gamma = \{X_1^{jr} \cdot P \mid P \in \beta, 0 \leq j \leq [\log h(P)]\}$. Let $t = X_1^{r[\log h(P)]}$ and $A = \{ \sum_{j=0}^{r[\log h(p)]} \delta_j X_1^j, \delta_j \in \{0, 1\}\}$. Then, we can write

$$(*) \quad P(x) = \sum \tilde{a}_{(I)} \cdot 2^{i_0} t^{i_1} X_2^{i_2} X_3^{i_3} \ldots X_k^{i_k}, \text{ where } i_0 < \log(M + 1), i_1 \leq \frac{d_1}{r[\log h(P)]},$$

$i_r \leq d_r$ for $r \geq 2$ and $\tilde{a}_{(I)} \in A$. An element of A can be obtained as the sum of $[\log h(P)]$ elements of Γ, and, since the total number of coefficients $\tilde{a}_{(I)}$ is bounded by $(1 + \log(M + 1)) \left(1 + \frac{d_1}{r[\log h(P)]}\right) (1 + d_2) \ldots (1 + d_k)$, we can obtain all the $\tilde{a}_{(I)}$'s and then obtain P using (*), and we can estimate the total cost as before, obtaining $\tau(P) \leq \frac{(1+\varepsilon)h(P)}{\log h(P)}$, for $h(P)$ sufficiently large.

ii.2.2) $M < 2^{\sqrt{\log h(P)}}$: In this case we construct all the polynomials $\sum_{j=0}^{r-1} \delta_j X_1^j$, with $\delta_j \in \{0, 1, 2, \ldots, M - 1\}$, $\forall j$ and $r = [\frac{\log B}{\log M}]$, forming the set β and then we construct all the polynomials in the set $\Gamma = \{X_1^{jr} \cdot P \mid P \in \beta, 0 \leq j \leq [\log h(P)]\}$. Let $t = X_1^{r[\log h(P)]}$

and $A = \{ \sum_{j=0}^{r[\log h(p)]} \delta_j X_1^j, \delta_j \in \{0, 1, 2, \ldots, M-1\}\}$. Then we can write

$$(*) \qquad P(x) = \sum \tilde{a}_{(I)} \cdot t^{i_1} X_2^{i_2} X_3^{i_3} \ldots X_k^{i_k}, \text{ where } i_1 \leq \frac{d_1}{r[\log h(P)]},$$

$i_r \leq d_r$ for $r \geq 2$ and $\tilde{a}_{(I)} \in A$. An element of A can be obtained as the sum of $[\log h(P)]$ elements of Γ, and, since the total number of coefficients $\tilde{a}_{(I)}$ is bounded by $\left(1 + \frac{d_1}{r[\log h(P)]}\right)(1+d_2)\ldots(1+d_k)$, we can obtain all the $\tilde{a}_{(I)}$'s and then obtain P using (*), and we can estimate the total cost as before, obtaining $\tau(P) \leq \frac{(1+\varepsilon)h(P)}{\log h(P)}$, for $h(P)$ sufficiently large.

Remark. The case of integer coefficients with $\max_{(I)} \left|a_{(I)}\right| = M$ is essentially equivalent to the case of natural coefficients with $\max_{(I)} a_{(I)} = 2M$. We can easily see this by adding or subtracting the polynomial $M(1 + X_1 + \cdots + X_1^{d_1}) \ldots (1 + X_k + \cdots + X_k^{d_k})$, which has very small cost.

Section 4: Remarks and open problems.

The most important open question related to $\tau(n)$ is to decide whether there are natural numbers $m_k \geq 1$ and a $C > 0$ such that $\tau(m_k \cdot k!) \leq (\log k)^C$, $\forall k \in \mathbf{N}$. If such a sequence does not exist then the algebraic version of $NP \neq P$ is true, as is proved in [SS]. About this question we have the following remarks:

- If there is a $C > 0$ such that $\tau(C_{2^n}^{2^{n-1}}) \leq n^C$ then there exists such a sequence m_k, namely $m_k = (2^{[\log_2 k]+1})!/k!$, because $(2^n)! = C_{2^n}^{2^{n-1}} (C_{2^{n-1}}^{2^{n-2}})^2 (C_{2^{n-2}}^{2^{n-3}})^4 \ldots (C_2^1)^{2^{n-1}}$

therefore $\tau((2^n)!) \leq n^{C'}$, for some constant C'. ($C_{2^n}^{2^{n-1}}$ is the central coefficient of $(1+X)^{2^n}$ and, given X, we can compute $(1+X)^{2^n}$ using $n+1$ operations.)

• If we have the operations $+, \cdot, -$ and 2^n then there exists such a sequence m_k. Indeed, we can construct a sequence M_k with $\tau(M_k) \leq 4k$ satisfying $n \leq 2^k$ hence $n|M_k$. It is enough to take $M_0 = 1$, $M_{k+1} = 2^{M_k}(2^{2M_k} - 1)$, because $n \leq 2^{k+1}, nodd \Rightarrow \frac{\varphi(n)}{2} \leq 2^k \Rightarrow \frac{\varphi(n)}{2} |M_k \Rightarrow n|\, M_{k+1}$ (M_{k+1} is divisible by sufficiently large powers of 2). Now is easy to check that $(2^n)!|M_0 \cdot M_1^2 \cdot M_2^4 \ldots M_n^{2^n} =: P_n$, which satisfies $\tau(P_n) \leq \frac{5n(n+1)}{2}$.

There is another open question concerning costs with bounded memory. Let $g \geq 1$ and define $\tau_g(n)$ as $\tau(n)$ with the restriction that for each $\ell \geq 1$, $s_\ell = Op(s_i, s_j)$ with $s_i, s_j \in \{1, s_{\ell-g}, s_{\ell-g+1}, \ldots, s_{\ell-1}\}$. The question is to determine if there are $g \geq 1$ and $C > 0$ such that $\tau_g(n) \leq \tau(n)^C$, for every $n \in \mathbf{N}$. About this we have the following remarks:

• If $g = 5$ and we have the operations $+, -, \cdot, /$ and $[\quad]$ (the integer part function) then $\tau_g(n) \leq \tau(n)^C$ for some C. Indeed, if $(s_0, s_1, \ldots, s_k)$, $s_k = n$ is a computing sequence for n we may keep $2s_0 + 4s_1 + 16s_2 + \cdots + 2^{2^r}s_r$, in the position $P1$ of the memory and use the position $P2$ to generate 2^{2^j}. We obtain s_j doing $\left[\left(\frac{X_j}{(2^{2^j})^2} - \left[\frac{X_j}{(2^{2^j})^2}\right]\right) \cdot 2^{2^j}\right] = s_j$, (recall that $s_j \leq 2^{2^{j-1}}$) where $X_j = 2s_0 + 4s_1 + \cdots + 2^{2^j}s_j$. We put s_i and s_j on the positions $P3$ and $P4$ and do $Op(s_i, s_j) = s_{r+1}$. Then we put s_{r+1} in the position $P3$ and $X_{r+1} = X_2 + 2^{2^{r+1}}s_{r+1}$ in the position $P1$. We proceed so until we obtain s_k. It is easy to see that the total time is polynomially related to $\tau(n) = k$.

• If $g = 1$ and the operations are $+, -, \cdot$ and $/$ then τ_g is not polynomially related to τ. Indeed, we can only transform an s_i into $s_{i+1} = s_i \pm 1$, $s_{i+1} = 2s_i$ or $s_{i+1} = s_i^2$.

If $(s_0, s_1, \ldots, s_k)$ is a computing sequence for $n = s_k$, and r is the maximum index such that $s_r = s_{r-1}^2$, we will have $s_k = 2^m s_{r-1}^2 + p$, with $m \geq 0$, $|p| \leq 2^{k-r} \leq 2^k$ hence there is $\ell \in \mathbf{N}$ such that $\left|n - \ell^2\right| \leq 2^k$ or $\left|n - 2\ell^2\right| \leq 2^k$. If $X_m = \frac{1}{2}((3 + 2\sqrt{2})^{2^m} + (3 - 2\sqrt{2})^{2^m})$, $Y_m = \frac{1}{2\sqrt{2}}((3 + 2\sqrt{2})^{2^m} - (3 - 2\sqrt{2})^{2^m})$, $n = X_m^2 + 2^{2^m}$, then $X_m^2 - 2Y_m^2 = 1$, $X_{m+1} = 2X_m^2 - 1$ so $\tau(X_m) \leq 3m$ and $\tau(X_m^2 + 2^{2^m}) \leq 4m + 1$, but if $a \in \mathbf{N}$, $\left|a^2 - (X_m^2 + 2^{2^m})\right| \geq \min\{2^{2^m}, \left|(X_m + 1)^2 - (X_m^2 + 2^{2^m})\right|\} = 2^{2^m}$, and $\left|2a^2 - (X_m^2 + 2^{2^m})\right| = \left|2a^2 - 2Y_m^2 - 1 - 2^{2^m}\right| \geq \min\{2^{2^m}+1, \left|2((Y_m + 1)^2 - Y_m^2) - 1 - 2^{2^m}\right|\} = 2^{2^m} + 1 \Rightarrow 2^k \geq 2^{2^m}$ thus $k \geq 2^m$, so $\tau_1(X_m^2 + 2^{2^m}) \geq 2^m >> \tau(X_m^2 + 2^{2^m})^C$, $\forall C > 0$ fixed, if m is large enough.

Notice, however, that $\tau_1(2^{2^k}) = k + 1$, and $\tau_1(n) \leq 2 \log_2 n$, for every $n \in \mathbf{N}$.

Acknowledgements. I would like to thank Steve Smale, Mike Shub, Welington de Melo and Benar Svaiter for many helpful conversations and sugestions of questions which are discussed in this article.

References:

[SS] Shub, Michael and Smale, Steve. On the Intractability of Hilbert's Nullstellensatz and an algebraic version of "$NP \neq P$?", to appear.

[MS] de Melo, W. and Svaiter, B.F.. The cost of computing integers, to appear.

[H] Heintz, Joos. "On the Computational Complexity of Polynomials and Bilinear Mappings. A Survey" in Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (Edited by G. Goos and J. Hartmanis), Lecture Notes in Computer Science 356, Springer Verlag.