MIT/LCS/TM-131

TIME, SPACE AND RANDOMNESS

Leonard M. Adelman

March 1979

MIT/LCS/TM-131

TIME, SPACE AND RANDOMNESS

LEONARD M. ADLEMAN
DEPARTMENT OF MATHEMATICS
AND
LABORATORY FOR COMPUTER SCIENCE

MARCH 1979

This report was prepared with the support of the National Science Foundation Grant No. MCS-78-04343.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

Abstract

Space and time are the fundamental parameters of complexity theory. The thesis of this paper is that randomness is of equal importance. We introduce a notion of randomness (based on Kolomogorov-Chaitin-Randomness), which we suspect will contribute to the understanding of some of the central problems in complexity theory. The purpose of this paper is primarily conceptual, though several easy theorems are given which clarify the relationship of this notion of randomness to the NP = P question, the complexity of integer factoring, and the sets computable in random polynomial time. Finally, using factoring as an example, we raise the possibility of performing experiments on functions of unknown complexity to indicate the extent of their tractability.

KEY WORDS: Computational complexity
Information theory
NP = P
Factoring

I. Motivation — An analogy with chemistry

At a philosophical level there is a similarity between chemistry and number theory. Both fall into the (rather large) class of disciplines where the main topic of concern (matter, numbers) can be understood in terms of simple combinations of special elements (atoms, primes). For the purpose of motivation we will pursue this similarity further.

The following is a familiar equation in chemistry:

$$2H_2 + O_2 \rightleftharpoons 2H_2O$$

It indicates a reversible reaction; oxidation to produce water and reduction to produce elemental hydrogen and oxygen. The analogous equation in number theory might be:

$$p \times q \quad = \quad pq$$

where multiplication results in a composite number and factoring produces primes. As anyone familiar with chemistry knows, the oxidation process in the first equation is quite rapid; however, the reverse reaction, carried out by the method of electrolysis, is quite slow. Of course, here too the analogy holds since multiplication is fast and factoring by known methods is slow. It is reasonable to wonder if the chemists have an $NP = P$-type question of their own: Are there faster methods for reduction than electrolysis, or are there reasons in principle why all methods must be slow? Fortunately for the chemist there are reasons in principle. They concern the idea of potential energy. Elemental hydrogen and oxygen have relatively large amounts of potential energy compared with water. While little energy (of activation) is needed to convert the elements to water, any process which does the reverse must pump large amounts of energy back. This manifests itself as fast and slow reactions.

It is natural to wonder if the analogy still holds. Is there a notion of storing potential in numbers with the result that high potential primes sometimes have relatively low potential products which are hard to factor by any means because all methods must take the time to pump the potential back? In the next section we will define such a notion of potential and prove that if factoring is not in P then this is the reason why.

II. Potent Numbers

Kolomogorov [7] and Chaitin [5] have established an excellent theory of the information content of strings. Intuitively, the amount of information in a string is the size of the smallest program which, starting on a blank input tape, outputs the string and halts. Thus 1^n contains little information because a program of size about $|n|$ will output it.[†] On the other hand, so-called Chaitin-random strings of length n require programs

[†] For our theory the choice of programming systems is not critical; most natural programming systems will suffice. In this abstract we will not bother specifying the particular one we have chosen.

of size about n (i.e. large enough to have the string stored in a table). Our notion of potential is related to concepts studied by Kolomogorov and Levin [8]; and Meyer and McCreight [10] and involves a refinement of the ideas of Kolomogorov and Chaitin to take into account the amount of time required to generate the string in question. The idea can be illustrated with 3 examples.

α_1 = 100000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
00000000000000001

α_2 = 100110110000100111111100010111100001001000100000100010111010
0000001

α_3 = 101000011101100110011010011111100100011110111001010010110110
001100001

α_1 is the binary representation of $2^{136}+1$. There is a small program (of size about $||\alpha_1||$) which generates it, and runs in about $|\alpha_1|$ steps. Figure 1 (on the next page) is a graph of the time-space tradeoff associated with α_1, where the value at i is the number of steps required by the fastest program of size i which generates α_1 (undefined if no such program exists).

α_2 was generated by flipping coins and is therefore Chaitin-random with high probability. Its time-space graph should look like the one shown in Figure 2. (Notice for any string of size n there is always a program of size about n which outputs the string in about n steps.)

α_3 looks as random as α_2; however, it is very likely that in the Kolomogorov-Chaitin measure it is more closely related to α_1. It is the largest prime factor of $2^{136}+1$ and so has a small description. However,

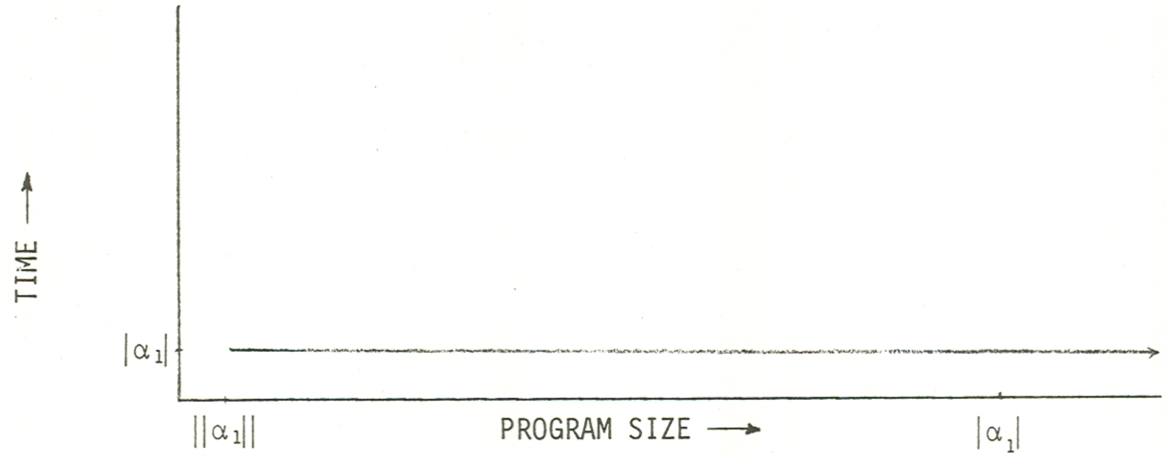

FIG. 1

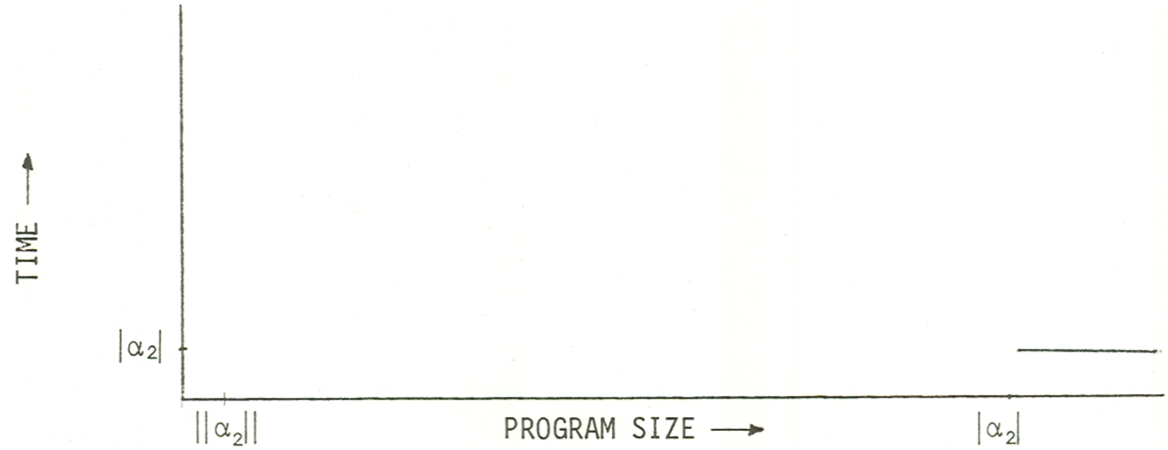

FIG. 2

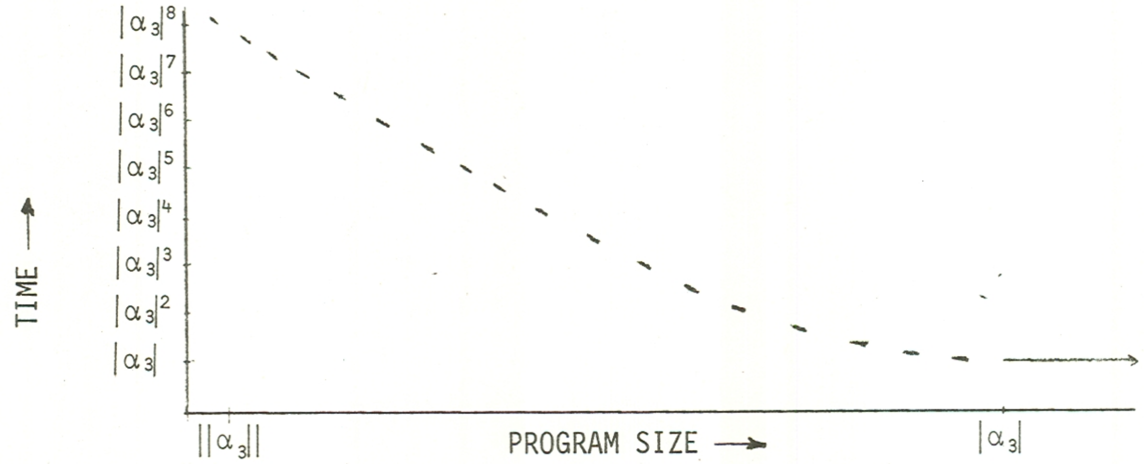

FIG. 3

in another sense it is likely to be much different from α_1, since it took approximately 2^{42} steps! on a special purpose device (the Lehmer Sieve [3] to find this factor. So the time-space graph for α_3 may be as shown in Figure 3.

Thus it appears that we may distinguish α_1 and α_3 by the amount of time small programs use to create them. We think of strings as storing this time. This motivates the following definition:

Definition For all $K \varepsilon N$, For all $\sigma \varepsilon \{0,1\}^*$ (For all $\tau \varepsilon \{0,1\}^*$), σ is K-potent (with respect to τ) iff there is a program p of size less than or equal to $K\|\sigma\|$, which with blanks (τ) as input halts with output σ in less than or equal to $|\sigma|^K$ steps.

Example For almost all n, 1^n is 2-potent.

Example For all K, for almost all Chaitin-random σ, σ is not K-potent.

Example Let $\overline{\alpha}_2 = \alpha_2 \oplus 1^{66}$ ($\oplus$ is exclusive or) then $\overline{\alpha}_2$ is Chaitin-random (assuming α_2 was) and 1-potent with respect to α_2.

The following is a useful fact about potent numbers.

Thrm. I For all K, the function

$$f_K(\sigma,1^n) = \{\alpha \,\Big|\, |\alpha| \le n \;\&\; \alpha \text{ is K-potent wrt } \sigma\}$$

is computable in polynomial time.

pf There are at most $2^{K|n|} \cong n^K$ programs of length $K|n|$. By simulating them on input σ for n^K steps the result is obtained.

The Kolomogorov Chaitin theory says that no computable function can map non-Chaitin-random strings into Chaitin-random strings except finitely often. However, computable functions which infinitely often produce outputs of higher potential than the inputs do exist (a straightforward diagonal argument suffices). We call such functions "inflating" and give a precise definition below.

Definition For all functions $f: \{0,1\}^* \to \{0,1\}^*$, f is an inflating function iff for all $K \in N$ there are infinitely many $\sigma \in \{0,1\}^*$ such that $f(\sigma)$ is not K-potent with respect to σ.

Example $$f(\phi) = \left\{\begin{array}{l} 1 \text{ if } \phi \text{ is a true sentence in th } (<\omega,+>) \\ 0 \text{ otherwise} \end{array}\right\}$$

is not inflating since $f(\sigma)$ is never very potent. If Range (f) is finite then f is not inflating.

Example $f(\sigma) = \sigma^R$ (σ reversed) is not inflating (though Range $(f) = N$ since $f(\sigma)$ is always of low potency with respect to σ.

Intuitively, algorithms computing inflating functions must infinitely often "scramble" inputs in order to get outputs — a syntactical process which could be expected to take time independent of the underlying semantics of f (i.e. whether f is number theoretic, graph theoretic, etc.).

Thrm II Let $\mathbb{F}$ denote the integer factoring function. Then

$$\mathbb{F} \notin P \Longleftrightarrow \mathbb{F} \text{ is inflating} .$$

pf (i) $\mathbb{F} \in P \Longrightarrow \mathbb{F}$ is not inflating

Assume there is an algorithm of size S which factors in time n^K for some K. Let α be any input of size $2^{S/K}$ or greater, then $\mathbb{F}(\alpha)$ is produced by a program of size $S \leq K||\mathbb{F}(\alpha)||$ (since $|\mathbb{F}(\alpha)| \geq |\alpha|$), starting on input α, and halting within $|\mathbb{F}(\alpha)|^K$ steps. Thus $\mathbb{F}(\alpha)$ is K-potent with respect to α and $\mathbb{F}$ is not inflating.

(ii) $\mathbb{F}$ not inflating $\Longrightarrow \mathbb{F} \varepsilon P$.

Assume $\mathbb{F}$ is not inflating, then there is a K such that for almost all α, $\mathbb{F}(\alpha)$ is K-potent with respect to α. But $|\mathbb{F}(\alpha)| \leq 2|\alpha|$. By Theorem I, $f_K(\alpha, 1^{2 \cdot |\alpha|})$ is computable in polynomial time. Since all prime factors of α are in $f_K(\alpha, 1^{2|\alpha|})$ we can in polynomial time arive at $\mathbb{F}(\alpha)$ by taking GCD's.

Thus factoring is difficult iff multiplication has infinitely often taken highly potent numbers and produced relatively low potency products, which are hard to factor because the potential must be pumped back. In fact searching for factors of a number by enumerating candidates in order of potency makes sense. Many other search procedures, for example exhaustive search, waste time on inputs like 1^n (so-called Mersenne numbers) by mostly checking Chaitin-random strings which could not possibly be factors (similar reasoning holds for searches on combinatorial problems). In fact search by increasing potency gives a polynomial time factoring algorithm if any polynomial time algorithm exists.

An important consequence of Thrm II is that it shows that the difficulty of factoring is not a global phenomenon but rather is a local one which we might hope to see in particular inputs and outputs. In [1] it was shown that the analogous statement for deciding sets in R is false. In section V we will pursue this further.

III. Potential Numbers and the NP = P Question

What makes logical theories like Presburger Arithmetic [6] and WSIS [9] intractible? One answer is their "expressive power". The theories have very short formulas (say of length n) which encode very long (of length at least 2^n) computational ID's. Unfortunately, it seems unlikely that such long sequences of ID's could be encoded in propositional formulas. Cook [4] has shown how to encode ID's of length n in propositional formulas of about length n, but it does not appear possible that ID's of length 2^n could be so encoded. Accordingly, it seems unlikely that "expressive power" as used above is the reason SAT $\notin$ P. Below we will show that SAT $\notin$ P iff there are formulas which are satisfiable but only by truth assignments with high relative potential. That is the "expressive power" of propositional formulas is not in their ability to encode very long sequences, but rather to encode very potent ones.

Thrm III $\text{SAT} \notin P \Longleftrightarrow (\forall K)(\overset{\infty}{\exists}\phi \in \text{SAT})\Big[(\forall T)[T \text{ is a truth assignment satisfying } \phi \Longrightarrow T \text{ is not K-potent with respect to } \phi]\Big]$.

pf

(1) SAT ε P

$$\Rightarrow f_{SAT}(\phi) = \begin{cases} 0 \text{ if } \phi \notin \text{SAT} \\ \text{the least T st T satisfies } \phi \text{ if } \phi \in \text{SAT} \end{cases}$$

is in p. (by binary searching truth assignments)

$\Rightarrow f_{SAT}$ is not inflating

$\Rightarrow (\exists K)\Big[(\overset{\infty}{\forall}\phi \in \text{SAT})(\exists T)\ [T \text{ is a truth assignment satisfying } \phi \text{ and } T \text{ is K-potent with respect to } \phi)\Big]$.

(2) $(\exists K)\left[(\overset{\infty}{\forall}\phi \in \mathrm{SAT})(\exists T)\right.$ [T is a truth assignment satisfying ϕ and T is K-potent with respect to ϕ]$\left.\right]$

$\Longrightarrow$ SAT ε P since using the technique described in the proof of Thrm II we could enumerate all of the T which are K-potent in ϕ and confirm if one satisfies ϕ in polynomial time.

IV. Potent Numbers and Random Polynomial Time

Intuitively we have been thinking of numbers as storing potential time. Below we give an illustration of how some highly potent numbers store time which can actually be "tapped" to speed up computation. In [1] we showed how for any set $A \in R$ (random polynomial time) it was possible to find tables which could be used to construct small fast programs (or small circuits) for deciding long initial segments of A. Let T_n^A denote the table constructed according to [1][†] which suffices for inputs of size n or less. Notice that the rules given in [1] show how to construct T_n^A from input n. It follows that the T_n^A are not in general Chaitin-random, but rather have very small descriptions; however, the next theorem shows that they have high potential:

Definition For all $S \leq \{0,1\}^*$, S is inflating iff $(\forall K)(\overset{\infty}{\exists}\sigma \in S)$ [σ is not K-potent].

Thrm IV Let $A \in R - P$ then

$$\{T_n^A \mid n \in N\} \text{ is inflating}$$

[†] with the additional rule that at stage i, w_i is the least witness with the desired property.

<u>pf</u> If $(\exists K)(\overset{\infty}{\forall} n)[T_n^A$ is K-potent]

then A may be decided in polynomial time by, on input σ generating all K-potent numbers of appropriate length, and outputing "yes" if one "witnesses" σ and "no" otherwise.

So for any K there are many T_n^A of potential greater than K and these T_n^A store time in a very tangible way. Without such a T_n^A the computation of a finite function (the characteristic function of A for inputs of size n or less) is slow (for small programs). With it, the same function can be computed quickly (in the chemistry analogy such a string would be a catalyst). Notice that this is precisely the property which a decryption key in an effective cryptographic system is supposed to have. Notice also that by being highly potent such a key is difficult to discover by any general means. We will not pursue the cryptographic aspects of these notions any further in this abstract, except to point out that in a discipline like cryptography which is shifting from an information theoretic base to a complexity theoretic one, a notion like potency is a possible bridge.

We now consider a refinement of the notion of an inflating set and relate it to the question of almost everywhere hard (nonpolynomial) sets in NP.

<u>Definition</u> For all $S \subseteq \{0,1\}^*$, <u>S is almost everywhere inflating</u> iff $(\forall K)(\overset{\infty}{\forall}\sigma \in S)[\sigma$ is not K-potent]

How hard is it to decide almost everywhere inflating sets? Intuitively it seems to demand non-polynomial time. However, the existence of almost everywhere hard to compute sets in NP would imply that this intuition is wrong.

Thrm V If there is an almost everywhere hard to compute set A in NP then there is an almost everywhere inflating set in NP with an infinite almost everywhere inflating subset in P.

pf Let $A \in NP$ then there is a polynomial time predicate Q and a constant c such that

$$x \in A \Longleftrightarrow (\exists y)[|y| \leq |x|^c \ \& \ Q(x,y)]$$

wlog we may add a slight refinement

$$x \in A \Longleftrightarrow (\exists y)[|x| \leq |y| \leq |x|^c \ \& \ Q(x,y)]$$

then let $W_A = \{y|(\exists x)[|x| \leq |y| \leq |x|^c \ \& \ Q(x,y)]\}$

W_A is the set of witnesses for A.

claim $W_A \in NP$.

By the definition of W_A this follows.

claim W_A is almost everywhere inflating.

If not then for some K there are infinitely many $y \in W_A$ which are K-potent. Such y (of appropriate length) could be generated in polynomial time on all inputs and infinitely often witness elements of A, implying A is not almost everywhere hard.

claim $\exists$ infinite set $V \subseteq W_A$ which is almost everywhere inflating and which is in P.

Since A is almost everywhere hard it follows that there are infinitely many n for which $1^n \varepsilon A$.

let $V = \{y|(\exists n)[n \leq |y| \leq n^c \& Q(1^n,y)]$

then $V \subseteq W_A$, V is clearly decidable in polynomial time, and V is almost everywhere inflating (since W_A was).

It is particularly hard as a result of the above theorem to imagine a set in R which is almost everywhere hard since in this case V is a huge set [i.e. $\limsup_{n\to\infty} \frac{V_n}{2^n} > 1/2$ where V_n is the number of elements of length n in V]. In the next section we will be concerned with the amount of computation required to distinguish highly potent numbers from low potency ones.

V. An Experiment

Below the reader will find 15 strings which he is asked to spend a few minutes examining in order to list them in order of randomness. The less enghusiastic reader is asked to pick the six strings which appear least random. An explanation follows the strings.

```
E01:=100011001110110,110101110100000,0000101000000001110000101101,100000110
     010100111110111101,0100011011110000010110100100,011000101100110

E02:=01,0001011001110,00001001110111001000,00111011111110001111110001,11011
     0010101100101110101011 10,000000000100010110001001011 01

E03:=11,1011,11111011,11111100011,1101101100000111010110011011,100101011001
     1011000111111010110000111110101100010110010001110100010010011

E04:=11,1010101010101010101010101010101010101010101010101010101010101010101
     01010101010101010101010101010101010101010101010101010101011

E05:=11,11,11,11,11,11,11,11,100111001,100111001,100111001,100111001,100111
     001,100111001,100111001,100111001,111010111,111010111,111010111,111010
     111,111010111,111010111,111010111,111010111

E06:=11,1011,1101,1011011100011010101,1010101100011101101,10110000100111011
     11,1101011010000000111,111000000101101011,11110111001000001101,1111111
     111111111111

E07:=11,1111111111111111111111111111111111111111111111111111111111111111111
     1111111111111111111111111111111111111111111111111111111111111111

E08:=11,10000011001,10010000110010001101001,1001001101001001001100011001010
     0110110010100011101001110011101001001001101011110010001001111101011

E09:=11,101011,101010101010101011,1000110101011,101011001000011010011010000
     01100011000001011111100101001111100010110010001100001111101111101000 1001

E10:=100000001,1011101101000100001,1000100111011000101000000110010110010000
     01,10100001110110011001101001111100100011101110010100101101100011000
     01

E11:=1,111,11110,11101111,01010001110,100000001100110001,110000000010001011,
     1001001101100001001111111,0001011110000100100010000,0100010111010000000
     01001100

E12:=11,10000011,10001001,10001111,10010001,10011101,10100111,10101011,1011
     1001,10111111,11000001,11001011,11010011,11010101,11100101,11101111,11
     110001,11110111,11111101

E13:=11,11111,100001000010000100001,1000000000000000000000001000000000000000
     00000000000100000000000000000000000100000000000000000000000001

E14:=1001,100100,110101,000001100110111,00001110101001110011101,00010111000
     0110101000000,10001000010111011010100111011,1011011100111101100010011 1
     110

E15:=10101,1110011,1111001,0011011,101111000,1110101011101,0001110010110111
     ,0011111011100011001110111100,0100011010101111001010001001 11
```

In section II it was shown that the integer factoring function $\mathbb{F}$ is in P iff it is inflating, and it was mentioned that this showed that the difficulty of $\mathbb{F}$ (if in fact it is difficult) is not a global property but rather can be found in particular elements $<\alpha,\mathbb{F}(\alpha)>$ of the graph of $\mathbb{F}$. This is true of any function G for which it can be shown that $G \varepsilon P \Leftrightarrow$ G inflating. In the case that such a $G \varepsilon P$ (especially if G is computable in n^2 or n^3 time) then for "simple" (i.e. low potency) α, $G(\alpha)$ must also be simple. In the case that $G \notin P$ then for simple α, $G(\alpha)$ might be "intricate" (i.e. highly potent). Thus, given a function G of unknown complexity, it might be possible to compute $G(\alpha)$ for a few simple α by brute force, and then try to detect whether or not $G \varepsilon P$ by testing the simplicity of $G(\alpha)$. The 15 strings given above are part of an informal experiment to test the feasibility of such an approach. The following table contains some of the results.

	Source	Plemp	Normalized	Times
E07	$G(x^{131}+1)$	4	.09	less than 2 min
E05	$G(x^{136}+1)$	12	.23	less than 2 min
E13	$G(x^{125}+1)$	11	.26	less than 2 min
E06	$G(x^{133}+1)$	28	.60	less than 2 min
E12	$G(x^{127}+1)$	43	.88	less than 2 min
E04	$\mathbb{F}(2^{127}+1)$	6	.14	less than 2 min
E08	$\mathbb{F}(2^{131}+1)$	35	.81	unknown
E09	$\mathbb{F}(2^{133}+1)$	37	.84	unknown
E10	$\mathbb{F}(2^{136}+1)$	40	.90	approx. 156,000 min
E03	$\mathbb{F}(2^{125}+1)$	39	.92	unknown
E02	R	37	.88	--
E01	R	41	.96	--
E15	R	43	1.02	--
E14	R	46	1.03	--
E11	R	49	1.07	--

plemp In an attempt to get a subjective idea of the randomness of the strings, a modification of an algorithm due to A. Lempel was used. This algorithm runs in linear time and is sensitive primarily to repetition and palidromes. Low plemp values indicate little randomness. We have no particular reason to believe this algorithm is good in any rigorous sense at detecting non-randomness of numbers of this length.

Normalized Since the amount of information in a string is dependent on its length, the plemp values were normalized to reflect this. Roughly, the normalized value is (plemp value)/(expected plemp value for a random string of the same length).

Times Where available, the actual amount of CPU time used in computing the strings is given for comparison with the randomness of the string (see the comment on E12 below). There is the suggestion here that the amount of time taken to produce E10 was related to its intrinsic potency rather than the particular program used to produce it.

G G is the function which maps polynomials into their factorizations over GF(2). It is easy to show that $G \notin P \Longleftrightarrow G$ is inflating. Since all polynomials over GF(2) have 0,1 coefficients, a polynomial like $x^{136}+1$ is encoded as $1{*}0^{135}{*}1$ and its factorization (E05) is presented using the same encoding. G has an interesting history. Kronecker gave an exponential time algorithm in the 19th century. Until 1968 no better algorithm had emerged (even for polynomials of the form x^n+1). In 1968, Berlekamp [2] gave an algorithm which computes G quickly (there is a subtle point here, since Berlekamp's algorithm runs in random polynomial time not polynomial time — we will ignore this). That a fast algorithm existed was not widely expected, and even recent journal publications [11] suggest G's use in cryptographic schemes, mistakenly thinking it is intractible. In our experiment we chose 5 simple inputs ($1*0^n*1$ for $n = 125,127,131,133,136$) and computed G on them (we did not use Kronecker's algorithm). All 5 outputs were clearly non-random. The fact that E12 has a normalized value of .88 shows the inadequacy of the plemp algorithm for this task (though in general it worked quite well). E12 is non-random since 1) the commas are evenly spaced, 2) it is a form of palidrome; if for $2 \le i \le 10$ we add the i^{th} group of 0's and 1's to the $(21 - i)^{th}$ group we always get 2^8. This behavior is consistent with a polynomial time non-inflating function. If Berlekamp's algorithm had not been discovered, would such evidence have been useful in directing research toward faster algorithms? If the integer factoring function $\mathbb{F}$ showed the same behavior would this be grounds to suspect it was tractable? Below we see the results of exactly the same test for $\mathbb{F}$.

$\underline{\mathbb{F}}$ E04, E08, E09, E10, E03 are the prime factorizations of $2^n + 1$ (i.e., $1*0^n*1$) for $n = 125,127,131,133,136$. We are indebted to J. Brillhart, D. H. Lehmer, and J. L. Selfridge who produced these factorizations in [3]. There is no known polynomial time algorithm for factoring in general, or for numbers of the form $2^n + 1$. With the exception of E04 all of these strings appear to be highly random[†]. Since in the Kolomogorov-Chaitin sense they all have small descriptions this apparent randomness is a reflection of potency. It is interesting that E04 was produced in less than 2 minutes since the factorization of $2^{127}+1$ turned out to be 3·prime. This pattern of results is much different than that for G, and is consistent with (and gives evidence that?) $\mathbb{F}$ is inflating and therefore not in P.

R Strings E02, E01, E15, E14 and E13 were obtained by flipping coins and are therefore likely to be Chaitin-random. They were correctly given high plemp values.

The experiment above was informal, but the results are suggestive. Statistical analysis of these results, new tests, and a rigorous investigation of the underlying principles would be helpful. Better algorithms than plemp can easily be found (e.g. checking strings against a master list of actual 3-potent numbers). If further investigations show the feasibility of such experiments, then they could prove to be a powerful tool in directing research.

VI. Conclusion

We think that perceiving time and space as attributes of strings as well as of functions will be useful in complexity theory and cryptography. This paper was inteneded to indicate the breadth of possibilities associated

[†] This is a subjective opinion and the author would appreciate the opinions of readers.

with this concept.

Acknowledgement

The author wishes to thank Ron Fagin, Abraham Lempel, Effrem Lipkin, Gary Miller, Vaughn Pratt, Ron Rivest, Adi Shamir, and Rich Zippel for many contributions both conceptual and technical.

1. Adleman, L., "Two Theorems on Random Polynomial Time", Proceedings 19th Annual Conference on Foundations of Computer Science (1978).

2. Berlekamp, E., "Factoring Polynomials Over Large Finite Fields", Mathematics of Computation 24 (1970), 713-735.

3. Brillhart, J., Lehmer, D., Selfridge, J., "New Primality Criteria & Factorizations of $2^m \pm 1$", Mathematics of Computation 29 (1975), 620-647.

4. Cook, S., "The Complexity of Theorem Proving Procedures", Conf. Records 3rd Annual ACM Symposium on Theory of Computing (1971).

5. Chaitin, G., "On the Length of Programs for Computing Finite Binary Sequences", JACM (13), 1966, 547-569.

6. Fischer, M., Rabin, M., "Super Exponential Complexity of Pressburger Arithmetic", Complexity of Computations, Edited by R. Karp, SIAM-AMS Proc., Vol. 7, Amer. Math. Soc. 1974, 27-41.

7. Kolomogorov, A., "Three Approaches to the Quantitative Definition of Information", PROB. INFO. TRANSMISSION, Vol. 1, No. 1, Jan. '65, 1-7.

8. Levin, A., private communication.

9. Meyer, A., "Weak Monadic Second Order Theory of Successor Is Not Elementary-Recursive", Bost. Univ. Logic Colloq. Proc. 1975.

10. Meyer, A., McCreight, E., "Computationally Complex and Pseudo-Random 0-1 Valued Functions"

11. JACM, exact reference unavailable.