Addition in $\log_2 n + O(1)$ Steps on Average: A Simple Analysis

Richard Beigel* Yale
Bill Gasarch† Maryland
Ming Li‡ City Univ. Hong Kong
Louxin Zhang§ Waterloo

Abstract

We demonstrate the use of Kolmogorov complexity in average case analysis of algorithms through a classical example: adding two n-bit numbers in $\lceil \log_2 n \rceil + 2$ steps on average. We simplify the analysis of Burks, Goldstine, and von Neumann in 1946 [2] and (in more complete forms) Briley[1] and Schay[6].

1 Introduction

Fifty years ago, Burks, Goldstine, and von Neumann [2] obtained an upper bound of $\log_2 n$ on the expected carry sequence length (that is, the longest sequence of consecutive nonzero carries when adding two n-bit binary numbers by the trivial ripple-carry algorithm). They did not propose an algorithm for addition based on it. In computer architecture design, efficient design of adders directly affects the length of the CPU clock cycle. The following algorithm (and its analysis using [2]) for adding two n-bit binary numbers x and y is known to computer designers and can be found in standard computer arithmetic design books such as [4] (in the algorithm, LSH stands for left-shift.):[1]

```
begin NO-CARRY ADDER
  $(\text{sum}, \text{carry}) := (x, y)$;
    while $\text{carry} \neq \vec{0}$ do
        $(\text{sum}, \text{carry}) := (\text{sum} \oplus \text{carry}, (\text{sum} \wedge \text{carry})$ LSH $1)$
end NO-CARRY ADDER
```

The algorithm terminates when there is no carry, hence the name. In our analysis, we assume that loop overhead and the time for an assignment statement or comparison to $\vec{0}$ are negligible, and

*Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Currently on sabbatical at the University of Maryland, College Park, MD 20742, USA. Email: beigel@cs.umd.edu. Supported in part by NSF grants CCR-8952528, CCR-9415410, and CCR-9522084 and by NASA (NAG52895).

†Department of Computer Science, University of Maryland, College Park, MD 20742, USA. Email: gasarch@cs.umd.edu

‡On sabbatical leave from: Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. Email: mli@cs.cityu.edu.hk. Supported in part by the NSERC Operating Grant OGP0046506, ITRC, and a CGAT grant.

§Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. Email: lzhang@daisy.uwaterloo.ca. Supported in part by CGAT.

[1]Schay [6] credits the connection of this algorithm and [2] to himself; however, this connection is alluded to in [7, Page 49] and clearly stated in [4]. It is curious that the architect of modern computers, von Neumann, did not invent this algorithm himself.

that $(\text{sum} \oplus \text{carry}, (\text{sum} \wedge \text{carry}) \text{ LSH } 1)$ can be calculated in a single time unit. Thus the running time of this algorithm is the number of executions of the body of the while loop. It is well known (and not hard to see) that this number is one greater than the length of the carry sequence for the pair of inputs.

Thus the expected $\log_2 n$ carry sequence length upper bound of [2, 1] implies that the no-carry adder runs in expected time at most $\log_2 n + 1$.[2] Schay [6] proves matching upper and lower bounds on the carry sequence length of approximately $\log_2 n - 0.65$, which implies that the no-carry adder runs in expected time approximately $\log_2 n + 0.35$. But the proofs in [2, 1, 6] involve fairly complicated probabilistic analysis. In this note, we will give a very simple proof of a $\log_2 n + O(1)$ upper bound using Kolmogorov complexity. (Kolmogorov complexity has already proved quite useful in the average-case analysis of algorithms, e.g., in Ian Munro's calculation of the expected number of comparisons made by Heapsort [5, pp. 334–338].)

We briefly review the definition of Kolmogorov complexity and some of its basic properties. For a more complete treatment of this subject, see [5]. Fix a universal Turing machine U with binary input alphabet. The machine U takes two inputs p and y. U interprets p as a program and simulates p on input y. The Kolmogorov complexity of a binary string x, given y, is defined as

$$C(x|y) = \min\{|p| : U(p, y) = x\}.$$

Thus $C(x|y)$ is the minimum number of bits from which x can be effectively reconstructed given y. Let $C(x) = C(x|\lambda)$, where λ denotes the null string.

Clearly $C(x|n, z) \geq 0$. For any universal program U, there is a constant δ (essentially the overhead for a PRINT statement), such that $C(x|n, z) \leq n + \delta$. Because there are at most 2^{n-i} programs of length $n - i$, there are at most 2^{n-i} strings satisfying $C(x|n, z) = n - i$, so for random x of length n the probability that $C(x|n, z) = n - i$ is at most 2^{-i}. The key to using Kolmogorov theory for average case analysis is to fix some parameter z, analyze the algorithm for inputs x satisfying $C(x|n, z) = n - i$, and then take the weighted average over all i between $-\delta$ and n.

2 The Average Case Analysis

Theorem 1 *For sufficiently large n, the no-carry adder runs in time at most $\lceil \log_2 n \rceil + 2$ on average.*

Proof: Let $n \geq 2^{\delta+1}$, so we have $-\delta \geq 1 - \lceil \log_2 n \rceil$. Let $\overline{s}$ denote the bitwise complement of a string s. For any x and y such that $|x| = |y| = n$, if the algorithm runs in exactly $t + 1$ steps on input x, y, where $t \geq 1$, then x and y can be written either as

- $(x, y) = (x'bu1x'', y'b\overline{u}1y'')$
- or as $(x, y) = (bux'', \overline{b}\overline{u}y'')$

where $|x'| = |y'| \geq 1$, $b \in \{0, 1\}$, $|u| = t - 1$, and $|x''| = |y''|$. Then x can be described using y, n,

[2]Although it runs in n steps in the worst case, the no-carry adder is the most efficient addition algorithm currently known for the average case. On average, the no-carry adder is exponentially faster than the ripple-carry adder and two times faster than the well-known carry-lookahead adder, which uses divide and conquer to add two n-bit numbers in $2\log_2 n + 1$ steps. The carry-lookahead adder is used in nearly all modern computers. Both of those algorithms can be found in almost any standard textbook, such as [4, 3].

and a fixed program q that reconstructs x from the concatenation of the following binary strings:

- the position of u in y, encoded in binary with *exactly* $\lceil \log_2 n \rceil$ bits (padding with 0s if necessary)
- $x'x''$

Since the concatenation of the two strings above has length $n-(t+1)+\lceil \log_2 n \rceil$, it together with n determines t. Thus $C(x|n,y,q) \leq n-t-1+\lceil \log_2 n \rceil$. Therefore, for any string x of length n with $C(x|n,y,q)=n-i$ where $i \geq 1-\lceil \log_2 n \rceil$, the algorithm must finish in at most $\lceil \log_2 n \rceil + i$ steps on input x, y.

Let p_i denote the probability that $C(x|n,y,q)=n-i$ where x is a uniform random string of length n. Then $p_i \leq 2^{-i}$ and $\sum p_i = 1$. Summing over all possible values of i, we find that the average running time for each y is bounded above by

$$\begin{aligned}\sum_{i=-\delta}^{n} p_i(\lceil \log_2 n \rceil + i) &= \sum_{i=-\delta}^{n} p_i \lceil \log_2 n \rceil + \sum_{i=-\delta}^{n} p_i i \\ &\leq \lceil \log_2 n \rceil + \sum_{i=0}^{\infty} 2^{-i} i \\ &= \lceil \log_2 n \rceil + 2.\end{aligned}$$

Since this holds for every y, $\lceil \log_2 n \rceil + 2$ is an upper bound on the average-case running time of the algorithm. ■

3 Acknowledgments

We would like to thank Behrooz Parhami for providing information on adder design, Ricard Gavaldá for pointing out some errors in an earlier draft, Richard Chang for proofreading, and Paul Vitanyi for helpful discussions. We would also like to thank Geza Schay, whose article [6] pointed us towards this problem.

References

[1] B.E. Briley, Some new results on average worst case carry. *IEEE Trans. Computers*, C-22:5(1973).

[2] A.W.Burks, H.H. Goldstine, J. von Neumann, Preliminary discussion of the logical design of an electronic computing instrument. Institute for Advanced Studies, Report (1946). Reprinted in *John von Neumann Collected Works*, vol 5 (1961).

[3] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms.* MIT Press, 1990.

[4] K. Hwang, *Computer arithmetic: principles, architecture, and design.* Wiley, New York, 1979.

[5] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications.* Springer, New York, 1993.

[6] G. Schay, How to add fast–on average. *American Mathematical Monthly*, 102:8 (1995), 725-730.

[7] N.R. Scott, *Computer number systems & arithmetic.* Prentice-Hall, 1985.