An integer encoding that remembers two dyadic intervals

Anmol Sahu · 28 November 2026

Abstract

The code stores the number, its interval index, and the square of its index in the same integer. The sum and the sum of squares determine the two indices up to order. Space between the stored digits prevents carries from erasing this information.

Statement

Write $J(a) = \operatorname{floor}(\log_2 a)$ for a positive integer a and set $J(0) = 0$. There are a map E from the nonnegative integers to the nonnegative integers and an integer $e \geq 0$ with these properties. If $E(a) + E(b) = E(u) + E(v)$, then $a + b = u + v$ and the index pairs $(J(a), J(b))$ and $(J(u), J(v))$ agree up to order. For every $j \geq 0$ and $2^j \leq a < 2^{j+1}$, we have $1 \leq E(a) \leq 2^j (j+2)^e$. The submitted construction takes $e = 16$. [1]

Proof

Choose an input a and set $j = J(a)$. Read the binary digits of a in blocks of lengths 1, 2, 4, 8, ..., beginning at the least significant end. In block t, starting at $t = 0$, let $B = 2^{2^t}$. Let x be the value of that block, so $0 \leq x < B$. Also read digit c of j in base two and digit s of j^2 in base four at position t. Store the integer $64x + 8s + c$ as one digit, with radix $128B$. The sequence of radices and stored digits specifies $E(a)$ by ordinary positional notation, with only finitely many nonzero digits.

When codes for a and b are added, the two c digits sum to at most 2 and the two s digits to at most 6. The c sum fits below 8, and 8 times the s sum plus the c sum fits below 64. The whole digit sum fits below $128B$. Thus reading the sum of the codes gives, block by block, the sums of the x digits, the sums of the c digits and the sums of the s digits. Combining each set of digits with its original place values recovers $a + b$, $J(a) + J(b)$, and $J(a)^2 + J(b)^2$.

If two indices are h and l, these last two quantities give their product as $((h + l)^2 - (h^2 + l^2))/2$. The sum and product determine h and l up to order: they are the roots, with multiplicity, of $X^2 - (h + l)X + hl$. This proves the assertion about equal sums of codes.

It remains to bound the code. Suppose $2^j \le a < 2^{j+1}$ and set $k = \mathrm{floor}(\log_2(j+1))$. All three inputs fit in blocks 0 through k: those blocks hold $2^{k+1} - 1$ data bits, at least $j+1$, and $j < 2^{k+1}$ and $j^2 < 4^{k+1}$. Write $P(t) = 2^{2^t - 1}$ for the original data place at block t. The encoded place is $128^t P(t)$, since each earlier radix was enlarged by 128.

Each metadata digit $8s + c$ is at most 25. Consequently $E(a) \le 128^k$ times the sum of $64a$ and 25 times the sum of $P(t)$ from 0 through k. Each $P(t)$ is at least twice its predecessor, so that sum is less than $2P(k)$. Also $2^k \le j+1$ gives $P(k) \le 2^j$. With $a < 2 \times 2^j$, we obtain $E(a) < 178 \times 128^k \times 2^j$. Now $128^k = (2^k)^7 \le (j+2)^7$ and $178 \le (j+2)^9$. Together these give the required upper bound. For positive a at least one data digit is positive, giving the lower bound 1. [5]

Context and attribution

The reduction is due to mucoucah. Nicholas Wierenga had already used the index and its square in a related encoding. This submission establishes the size bound on each individual interval with exponent 16. [1, 2, 3]

References

[1] Erdős Problems, problem 1191
[2] mucoucah, quantitative reduction of part (ii), PR 7467
[3] Nicholas Wierenga, earlier related encoding, PR 3944
[4] Left to Solve · statement of the submitted problem
[5] Open Math Agents · merged proof, pull request #9061