Flash memory

Spoiler

What is maximum average number of messages that can be written for $B = 2$, $M = 1$?

10 points. Copy variable to memory. If the memory is already nonzero, return that you can't write anymore.

20 points. Writing: Append 1 to the value. If there are at least $M + 1$ bits 0 at the end of memory, write the amended value after the last 1 in memory. Reading: Read the M bits of memory before the last 1. This solution can write $\lfloor \frac{B}{M+1} \rfloor$ values.

40 points. Split memory into M blocks of length $\frac{B}{M}$. The i-th bit of memory is encoded as the XOR of the bits in the i-th block of memory. While writing change one bit in the i-th block of memory from 0 to 1 if this block would be decoded wrong otherwise. The probability of a write increasing the number of 1-s in a block of memory is $\frac{1}{2}$. Because of this, you can write $2\frac{B}{M}$ times in a single block on average. As you can no longer write when one of the blocks is full, you can write sligtly less than $2\frac{B}{M}$ values on average.

Solving $M = 8$ near optimally. For each bit of memory, pick a random binary vector of length M (0 and 1 equiprobable). It is important to have a fixed seed so that the vectors created by encoder and decoder are the same. Values (which are also binary vectors of length M) are encoded as the XOR of the vectors for which the respective bit in memory is set to 1. This can also be thought of as matrix multiplication.

When writing a value, you want to change as few bits as possible from 0 to 1. To accomplish this we can do bit-mask DP with states DP$[B][2^M]$. DP$[i][j]$ is the minimum number of bits that need to be changed from 0 to 1 among the first i bits to have the XOR of the vectors corresponding to 1-s in the first i positions of memory be equal to j. The time complexitiy of an single encode is $\mathcal{O}(B2^M)$. The average number of values written is $\mathcal{O}(\frac{B}{M}\log(\frac{B}{M}))$.

75 points. Combine the ideas from the last two sections. Split variable into k blocks of length rougly 11. Split memory into k blocks with roughly equal lengths. Apply the scheme from the previous section for each pair of blocks. In C++ it runs fast enough with 11, smaller values get slightly fewer points. The time complexitiy of a single encode is $\mathcal{O}(B2^M)$. If B is at least a few times bigger than M, then the average number of values written is $\mathcal{O}(\frac{B}{M}\log(\frac{B}{M}))$.

Theoretical bounds. It can be proven with dynamic programing that the average number of writes is no greater than the P given in the task statement for any program. It can be proven that $P = \mathcal{O}(\frac{B}{M}\log(\frac{B}{M}))$. For $B = 16$, $M = 8$ this can be achieved. The jury has a solution that gets 80 points and has ideas how to improve it to 85+ points.

Credits

- Task: Heno Ivanov, Martin Širokov (Estonia)
- Solutions and tests: Oliver Nisumaa, Martin Širokov, Andres Unt (Estonia)