Flight to the Ford (communication)

BY LUKAS MICHEL (GERMANY)

Subtask 1. $N = 3$

By trial-and-error one can come up with a scheme that uses 3 bits. However, it is probably easier to find (and describe) a solution to this subtask that never looks at the return value of *send*. Put differently, we want to find three bitstrings (which after padding we can assume without loss of generality to be of the same length ℓ) such that for any bitstring B of length ℓ at most two of the fixed bitstrings could be possibly corrupted into B. You can find such bitstrings either by hand or by writing a separate program for that, which bruteforces all possible bitstrings of fixed small length; one possible choice for $\ell = 4$ is given by `0000`, `0110`, and `1111`. Alternatively, you can also construct three random bitstrings of medium to large length and hope to sneak past our grader.

Subtask 2. No further constraints

Partial solutions. The first thing we can try here is to iterate our solution from the previous subtask: in the beginning, the assistant only knows that the message X is one of $1, \dots, N$. If we partition the set $T_0 = \{1, \dots, N\}$ into three sets S_1, S_2, S_3 of roughly the same size and use the previous subtask to send the unique i with $X \in S_i$, then afterwards the assistant will know that $X \in S_i \cup S_j =: T_1$ for some j. Note that we will actually also know that value of j (and hence the set T_1) since *send* tells us which signals were actually sent. Thus, we can iterate the above process for the new set. As for $|T_n| > 2$ always $|T_{n+1}| < |T_n|$, this process will eventually terminate with only two possible messages left, and since in fact any step cuts down the size of T by a constant factor, we will only need $O(\log N)$ steps. Doing the exact maths or simply submitting and hoping for the best results in around 30 points (including the 15 points from the previous subtask) for the above scheme with 4 bits for $N = 3$ or even 50 points for the solution with bitstrings of length 3. Of course, we have to be slightly careful how we store the sets T as we have neither enough time nor space in our submarine to keep a set of up to 10^9 elements. However, we can arrange for the set T_n to be a union of at most linearly many disjoint intervals by simply going through the remaining possible messages in increasing order when doing the splitting. Storing the T's as such unions of disjoint intervals (and using this representation to compute the splittings for the next step) easily fits into the respective limits.

There are now several ways to improve this solution, leading to a wide range of partial scores. In particular, we can change the bitstrings we use to cut down the size of the search space. For example, we can observe that one can actually also solve the case $N = 4$ with the bitstrings `0000`, `0110`, `1001`, `1111` of length 4 (without looking at the return values), which gives around 75 points.

Another idea that works very well is to simply take *all* possible bitstrings of length ℓ. Let us think about how much this actually cuts down the search space: for a given bitstring B of length ℓ received by the assistant, without prior knowledge the possible original bitstrings are in 1-to-1-correspondence with the possible error patterns, i.e. the strings of length ℓ containing no consecutive ones (the correspondence is given by taking bitwise xor with B). We can compute the number F_ℓ of such strings recursively: if the last bit of our new string is `0`, then the remainder of the string is any valid error pattern; on the other hand, if the last bit is `1`, then the bit directly before that has to be `0` while the remaining part before that is again any valid error pattern. Thus, we get the relation $F_\ell = F_{\ell-1} + F_{\ell-2}$ for all $\ell \geq 2$ while $F_0 = F_1 = 1$, i.e. F_ℓ is the ℓ-th Fibonacci number.[*] For $\ell \gg 0$, $F_\ell \approx \Phi^{\ell+1}/\sqrt{5}$ with $\Phi = \frac{1}{2}(1+\sqrt{5})$,

[*] There is a major dispute between the two SC chairs whether one should define $F_0 = 0$ or $F_0 = 1$. I'm sorry if you are used

so that $|T_{n+1}| \approx (\sqrt{5}/\Phi) \cdot (\Phi/2)^{\ell} |T_n|$. Because of the constant factor $\sqrt{5}/\Phi > 1$, this does not work well for small ℓ while we may not make ℓ too large once the search space has been reduced sufficiently. A solution balancing these outperforms the solution based on four patterns of length 4 mentioned above slightly.

Full solution. (cf. J. Czyzowicz, K.B. Lakshmanan, A. Pelc. *Searching with local constraints on error patterns*. European J. Combin. **15** (1994), pp. 217–222) For the full solution we will refine the above ideas to keep two sets $T_n^{\text{corr}}, T_n^{\text{wrong}}$ in *decode*, denoting the sets of possible messages X if the last bit was correct or incorrect, respectively. We will assume for now (and the procedure below will guarantee) that these two sets are disjoint.

Given such a state, let $U \subset \{1, \dots, N\}$ such that we *send* the bit 1 precisely if $X \in U$. Then afterwards, *decode* will be in the state $(T_{n+1}^{\text{corr}}, T_{n+1}^{\text{wrong}}) = (U \cap (T^{\text{corr}} \cup T^{\text{wrong}}), T^{\text{corr}} \setminus U)$ if the signal received is 1 and $((T^{\text{corr}} \cup T^{\text{wrong}}) \setminus U, U \cap T^{\text{corr}})$ otherwise. In particular,

$$(|T_{n+1}^{\text{corr}}|, |T_{n+1}^{\text{wrong}}|) = \begin{cases} (|T_n^{\text{corr}} \cap U| + |T_n^{\text{wrong}} \cap U|, |T_n^{\text{corr}}| - |T_n^{\text{corr}} \cap U|) & \text{if the bit is 1} \\ (|T^{\text{corr}}| - |T_n^{\text{corr}} \cap U| + |T^{\text{wrong}}| - |T_n^{\text{wrong}} \cap U|, |T_n^{\text{corr}} \cap U|) & \text{otherwise.} \end{cases}$$

So how should we pick the intersections $T_n^{\text{corr}} \cap U$ and $T_n^{\text{wrong}} \cap U$? One idea one can come up with is that the sizes of the sets T_{n+1}^{corr} and T_{n+1}^{wrong} should actually be independent of the bit the other side receives (as we really only care about the sizes of the sets and not about their elements, this prevents our opponent from ever forcing us on the 'worse' path). Solving the corresponding system of linear equations then suggests that U should contain half of the elements of T_n^{corr} and T_n^{wrong} each. For the specific roundings

$$|T_n^{\text{corr}} \cap U| = \left\lceil \frac{|T_n^{\text{corr}}|}{2} \right\rceil \qquad |T_n^{\text{wrong}} \cap U| = \left\lfloor \frac{|T_n^{\text{wrong}}|}{2} \right\rfloor$$

one can (let a computer) calculate that this reduces the search space (for $N = 10^9$) after 96 bits to $|T_n^{\text{corr}}| + |T_n^{\text{wrong}}| \leq 3$. While this strategy isn't able to reduce the search space any further at this point, we can combine this with one of the solutions to the previous subtask to get a solution which uses 99 (or 100) bits. Again ensuring that T^{corr} and T^{wrong} are always given by a union of linearly many disjoint intervals, this can be implemented in linear space and time quadratic in the number of bits sent, which easily fits into the time and memory constraints and gets full score.

In fact, the above solution can be shown to be close to optimal. For this, let us consider any possible strategy to choose the sets U in each step and look at the binary decision tree with nodes the pairs $(T^{\text{corr}}, T^{\text{wrong}})$ and edges the bits received by the assistant. We can assume without loss of generality that all leaves are at the same height by simply sending `0` in *encode* once we have sent enough bits for the solution but not yet sent as much bits as on the worst possible path.

For every $k \geq 0$ we set $t^{\text{corr}}(k)$ to be the sum of the sizes of all the sets T^{corr} appearing at level k, and analogously we define $t^{\text{wrong}}(k)$; by convention, $t^{\text{corr}}(0) = N$ and $t^{\text{wrong}}(0) = 0$. Looking at the above formula, we see that independently of the choice of U we always have $t^{\text{wrong}}(k+1) = t^{\text{corr}}(k)$ and $t^{\text{corr}}(k+1) = t^{\text{corr}}(k) + t^{\text{wrong}}(k)$. Inductively we therefore see that $t^{\text{corr}}(k) + t^{\text{wrong}}(k) = (F_{k-1} + F_k) \cdot N = F_{k+1} \cdot N$. On the other hand, for any leaf $|T^{\text{corr}}| + |T^{\text{wrong}}| \leq 2$, and since there are precisely 2^Q nodes at level Q, we see that if the above tree has height Q, then $2 \cdot 2^Q \geq F_{Q+1} \cdot N$, i.e. $2^{Q+1}/F_{Q+1} \geq N$. Using the approximation $F_k \leq \Phi^{k+1}/\sqrt{5}$ again, we therefore see that $Q \geq \log_{2/\Phi} \frac{N\Phi^2}{2\sqrt{5}}$. Plugging in $N = 10^9$ shows that $Q \geq 96$, i.e. we need at least 96 bits.

to the other convention—you are obviously wrong though

Final remarks

1. The above argument can be adapted to give a lower bound of 96 on the number of queries in *any* solution, by setting T^{corr} and T^{wrong} to be the sets of all messages that *decode* could possibly return for any possible future communication.
2. It is actually not hard to prove that it is impossible to unambiguously communicate under the given circumstances already for $N = 2$: considering two runs of *encode*, we can always arrange that for any i the i-th call to *send* in the first run returns the same as the i-th call in the second run by simply alternatingly returning what the first or the second run wanted to send. Thus, doing this for two distinct messages X and Y, we can guarantee that one of *encode*(N, X) and *encode*(N, Y) is a prefix of the other, making it impossible for *decode* to decide which message *encode* wanted to communicate (as it has to stop reading after having received the prefix).
3. Writing a grader for this problem was a lot of fun.[†] Of all the strategies which bits to flip we tried, two ideas proved to be particularly successful: the first one tries to force the output into repetitions of a fixed pattern, 'learning' from previous iterations of your program which pattern to use; in the second family of strategies, we actually run multiple instances of your program in parallel for different messages, using the above trick to enforce that the signals sent in one of the runs is a prefix of at least one other run's signals, combined with several different heuristics on what to do in cases where we have a choice which bits to flip.

[†] Well, at least most of the time...