# A Cultural Gap Revisited

*This column is devoted to mathematics for fun. What better purpose is there for mathematics? To appear here, a theorem or problem or remark does not need to be profound (but it is allowed to be); it may not be directed only at specialists; it must attract and fascinate.*

*We welcome, encourage, and frequently publish contributions from readers—either new notes, or replies to past columns.*

Please send all submissions to the Mathematical Entertainments Editor, **Alexander Shen,** Institute for Problems of Information Transmission, Ermolovoi 19, K-51 Moscow GSP-4, 101447 Russia; e-mail:shen@landau.ac.ru

Fourteen years ago *The Mathematical Intelligencer* published an article by E.W. Dijkstra ("On a Cultural Gap," vol. 8, no. 1, 48–52) that discussed the roots of a cultural gap between the typical computer scientist and the typical mathematician. According to Dijkstra, this gap is a significant obstacle both for mathematicians and programmers and should disappear in the future when "programs will display all the beauties of a crisp argument."

Looking at today's software (commercial and even free), we have to admit that this future hasn't come yet. But Dijkstra's arguments are still convincing, and such a future indeed looks possible (though it may never come due to commercial reasons).

Nevertheless, there may be a cultural gap between mathematics and computer science (or programming) at a more subtle level. To explain it, let us consider the following puzzle.

*There are N objects that seem to be identical, but in fact belong to several different types. One of the types forms a majority (more than N/2 objects belong to that type). Our task is to point out one of the objects from this majority. The only tool we have is a detector that cannot tell the type of an object but that when applied to any two objects will say whether those two objects are of the same type or not.*

Of course, one can apply the detector to all pairs of objects and get a complete classification, but the number of measurements will be about $N^2/2$. How many measurements do we really need? It turns out that a more efficient approach is possible, where the number of measurements is proportional to $N$, not $N^2$.

I am now going to present two proofs of this claim, a mathematician's proof and a programmer's proof. See if you agree with me about the difference in viewpoint.

Both proofs start with the following simple observation: *if two objects have different types, both of them can be discarded without changing the majority type.* Indeed, in discarding two objects of different types, we discard at most one "good" object and at least one "bad," so the majority remains a majority. For the same reason, two friends who are going to vote for different candidates may agree to ignore elections if both are sure that there is some candidate who has more than 50% support.

The mathematician's proof continues: Let us assume for the moment that $N$ is even. Then we can group our $N$ objects into $N/2$ pairs and apply the detector to each pair, making $N/2$ measurements. Pairs that contain different objects are discarded. After that, we are left with a number of pairs, each consisting of objects of the same type; from each pair we retain only one object. Then we have the same problem of finding the majority representative, but with (at most) $N/2$ objects. Therefore we get the recurrence

$$T(N) \leq N/2 + T(N/2),$$

where $T(n)$ is the minimal number of measurements required to solve the problem for at most $n$ objects. Taking into account that $T(2) = 0$, by induction we see that $T(N) < N$.

It remains to explain why odd values of $N$ do not spoil this nice picture. In the odd case, after grouping elements into pairs and discarding pairs with different elements, we have some pairs with equal elements and one unmatched element. For example, we may have pairs $(a,a)$, $(b,b)$, $(c,c)$ and one unmatched element $d$. In this example 7 elements remain. We know that "winning" elements form a majority among them, so there must be at least 4 winning elements. But then at least two winning pairs exist, otherwise there would be at most 3 winning elements. Therefore winning elements form a majority among $a,b,c$, and we can drop $d$ completely.

The situation is different if after discarding equal pairs we have four pairs $(a,a)$, $(b,b)$, $(c,c)$, $(d,d)$ and one unmatched element $e$. Here we need 5 elements to form a majority, and it may be achieved using two pairs and $e$. So winning elements do not need to form a majority among $a,b,c,d$. But they do need to form a majority among $a,b,c,d,e$; for otherwise there would be 5 losing elements. So in this case we should retain $e$ in the sample.

It is easy to see that one of these two arguments is applicable; we need only to keep the sample's size odd. (End of mathematician's proof.)

The programmer replies: Imagine that you are locked in a magic room with the detector and $N$ objects. There are three big boxes in the room. The boxes are labeled: UNTESTED, IDENTICAL, and DISCARDED. Initially all the objects are in the box labeled UNTESTED, and it is guaranteed that most of them have the same type ("winning type").

The magic room has two laws, called also "invariant relations." (If you violate one of them, you are executed immediately.) Here they are:

> All objects in the IDENTICAL box must be identical (if the box is not empty).
>
> Objects of the winning type (which is determined in advance but unknown) must form a majority among non-discarded objects (i.e., among objects that are in either the UNTESTED or the IDENTICAL box).

Evidently, these conditions are satisfied in the initial state.

When the UNTESTED box (**U** for short) becomes empty, the door is unlocked and you are free. (You deserve it, for at that point all non-discarded objects are identical, so you have found at least one object of the winning type.)

What will you do after the rules are explained? Some observations are almost evident.

First, if the IDENTICAL box (**I** for short) is empty (while **U** is not), one can safely move one object from **U** to **I**. (The set of non-discarded objects remains the same, so the laws are not violated.)

Second, if both **I** and **U** boxes are not empty, one can take one object from each and compare them, using the detector. If they have the same type, it is safe to put both objects into **I**; if they have different types, it is safe to discard both (as we have seen earlier).

It remains to point out that (1) in any situation one of these observations can be applied (unless **U** is empty); (2) the number of untested objects decreases at each step; (3) the detector is used at most once at each step. Therefore the **U** box becomes empty after at most $N$ operations (in fact fewer, because the first operation does not involve the detector). (End of programmer's proof.)

It is easy to transform the programmer's story into a short program (the curious reader may find it on pp. 71–72 of my book *Algorithms and Programming: Problems and Solutions*, Birkhäuser, 1997). The mathematician's argument, of course, also can be transformed into a program, but this program is much more complicated.

Looking at this example, one may try to interpret the difference between mathematician's and programmer's viewpoints. Here is one possible explanation. If some problem $P$ is decomposed into many similar subproblems $P_1, \ldots, P_n$ all of which are trivial, the mathematician is in the habit of considering $P$ as trivial and may write something like, "$P_i$ having been proved, we may leave all other $P_i$ as exercises for the pedantic reader." The programmer, on the other hand, knows very well that it is her/his task to write programs for all $P_i$, so even if all these programs are short, for a big $n$ (s)he has a lot of work, and it would be better to find another solution for $P$ that does not involve many subproblems.

**Pentangram: Correction**

We received the following letter in response to the column about pentangrams:

> I would like to point out that there is an error in the article about pentangrams in the spring 1999 issue of *The Mathematical Intelligencer*. In the 3rd paragraph of the left-hand column of page 16, the author states that for regular polygons with 7, 9, 11, 13 or 19 sides, the ratio of the chords and sides is transcendental. This is plainly wrong. The ratios are algebraic.
>
> Gabor Megyesi
>
> e-mail: gmegyesi@am.ma.umist.ac.uk

The same error was also pointed out by Prof. John Sharp (Watford, England). I apologize for not finding this error earlier. Clearly, for any $n$ the $n$th roots of unity are algebraic and all ratios formed from them are algebraic too.