Experiment - Bad Genius

30ish pts

Problem Solving Technique: Familiar framing
If you can frame a weird scenario in terms of familiar (standard) objects, it'll be easier to make connections using your prior knowledge.

Instead of `T` or `F`, use `1` and `0`, representing the answer sheet as a bitstring.

Instead of "a 16-digit string which may have leading zeros", just say that your code is an integer from 0 to $10^{16} - 1$.

We *know* how to relate bitstrings with integers—using binary!

Because $2^{53} < 10^{16} \leq 2^{54}$, we frame the problem like this instead:

- Instead of sending a 16-digit code, you are sending a 53-bit code.

Then, getting 53/100 is pretty easy—just send over the first 53 answers directly, and guess `F` on items 54 and onwards. This gives 19.27 pts.

But why guess only `F`? If you guess randomly for items 54 and onwards, then by sheer random chance, you will get 63/100 with high probability, and re-submitting with different random seeds might let you get 64/100 or 65/100.

This gives 30ish pts.

Implementation
Remember to add padding leading zeros if the code is an integer with fewer than 16 digits!

57.598 pts

Seems a bit negligent to leave 47 items totally up to chance, don't you think?

It'd be nice to give *some* info about those items, even if it's not perfect.

Consider the extreme case: If we only allocate 1 bit to be responsible for all the remaining items, what could you do with it?

A lot, actually!

- Let the first 52 bits correspond to the answers of the first 52 items.
- Let the final bit be responsible for the final 48 items, giving you the value of **which of T or F appears more** among that 48.

This final bit guarantees that we know the corect answer to at least 24 out of those final 48 items (by answering all T, or all F, whichever appears more), even if we don't know which items those are!

Thus, we are guaranteed a 76/100, which gives 57.598 points.

You are expected to get 57.598 pts on this problem, *maybe 60.548* with some random tweaking, then move on with your life.

Road to 100 pts (87/100)

Getting even more points is now a *research* task.

> **Problem Solving Technique: Familiar framing**
> If you can frame a weird scenario in terms of familiar (standard) objects, it'll be easier to make connections using your prior knowledge.

The following framing might not initially be so "familiar" to you, but it is a helpful springboard for how to frame the problem in such a way that is amenable to "scouring the literature for useful ideas", i.e. research.

Suppose the test has 7 items only, and the code we can send over has 4 bits in it. Consider the following collection of 16 bitstrings of length 7:

- `0000000`
- `0101010`
- `1101001`
- `1000011`
- `1110000`
- `1011010`
- `0011001`
- `0110011`
- `1001100`
- `1100110`
- `0100101`
- `0001111`
- `0111100`
- `0010110`
- `1010101`
- `1111111`

It can be shown that given *any* bitstring of length 7, I can find a magic bitstring in this list that differs from it (this is called the Hamming distance) in at most 1 position only.

So, using only 4 bits, it is possible for me to get a 6/7 on the exam:

- We create a *codigo* that is just a list of all 16 of these bitstrings
- Alice finds which of these magic bitstrings is closest to the answer key, and encodes that bitstring's *index* in the *codigo* (0 to 15).
- A cheater has that *same codigo*, and answers according to the magic bitstring at the specified index (communicated through the code).

This is called the Hamming(7, 4) code, and is a classic result in error-correcting codes (and in our case, data compression).

But our exam has 100 items, so what can we do? Simple. Split the exam into **chunks of 7 bits at a time**, and also split the *code* into **chunks of 4 bits at a time**. For example...

- Split up items 1 through 91 into thirteen chunks of length 7.
- Split up the first 52 bits of the code into thirteen chunks of length 4.
- Each of these chunks in the code specifies which Hamming code you should use to answer its corresponding chunk of 7 questions.
- Finally, use the final bit of the code to encode which of T or F appears *more* among questions 92 through 100.

This guarantees a score of $13 \times 6 + 5$, for 83/100 (and 81.989 pts).

How might you discover this? With some experience, you might be able to formulate this *general* framing of the problem:

- Construct a magic set of 2^k bitstrings of length n with the property that: for any bitstring of length n, I can find a bitstring from the set such that the Hamming distance between it and my given bitstring is at most d.

(the Hamming codes are an example with $n = 7$ and $k = 4$) which has a rich amount of Google-able theory.

To get 100 pts in this problem, you need to be able to guarantee a score of 87/100 in the exam. If you want full points, I hope you have fun while exploring the fun world of error-correcting codes!