which means for $f \simeq 0.01$ that we need an extra 7 bits above $\log_2 S$.

The important point to note is the scaling of T with S in the two cases (12.7, 12.8). If we want the hash function to be collision-free, then we must have T greater than $\sim S^2$. If we are happy to have a small frequency of collisions, then T needs to be of order S only.

Solution to exercise 12.5 (p.198). The posterior probability ratio for the two hypotheses, $\mathcal{H}_+$ = 'calculation correct' and $\mathcal{H}_-$ = 'calculation incorrect' is the product of the prior probability ratio $P(\mathcal{H}_+)/P(\mathcal{H}_-)$ and the likelihood ratio, $P(\text{match}\,|\,\mathcal{H}_+)/P(\text{match}\,|\,\mathcal{H}_-)$. This second factor is the answer to the question. The numerator $P(\text{match}\,|\,\mathcal{H}_+)$ is equal to 1. The denominator's value depends on our model of errors. If we know that the human calculator is prone to errors involving multiplication of the answer by 10, or to transposition of adjacent digits, neither of which affects the hash value, then $P(\text{match}\,|\,\mathcal{H}_-)$ could be equal to 1 also, so that the correct match gives no evidence in favour of $\mathcal{H}_+$. But if we assume that errors are 'random from the point of view of the hash function' then the probability of a false positive is $P(\text{match}\,|\,\mathcal{H}_-) = 1/9$, and the correct match gives evidence 9:1 in favour of $\mathcal{H}_+$.

Solution to exercise 12.7 (p.199). If you add a tiny $M = 32$ extra bits of hash to a huge N-bit file you get pretty good error detection – the probability that an error is undetected is 2^{-M}, less than one in a billion. To do error *correction* requires far more check bits, the number depending on the expected types of corruption, and on the file size. For example, if just eight random bits in a megabyte file are corrupted, it would take about $\log_2 \binom{2^{23}}{8} \simeq 23 \times 8 \simeq 180$ bits to specify which are the corrupted bits, and the number of parity-check bits used by a successful error-correcting code would have to be at least this number, by the counting argument of exercise 1.10 (solution, p.20).

Solution to exercise 12.10 (p.201). We want to know the length L of a string such that it is very improbable that that string matches any part of the entire writings of humanity. Let's estimate that these writings total about one book for each person living, and that each book contains two million characters (200 pages with 10 000 characters per page) – that's 10^{16} characters, drawn from an alphabet of, say, 37 characters.

The probability that a randomly chosen string of length L matches at one point in the collected works of humanity is $1/37^L$. So the expected number of matches is $10^{16}/37^L$, which is vanishingly small if $L \geq 16/\log_{10} 37 \simeq 10$. Because of the redundancy and repetition of humanity's writings, it is possible that $L \simeq 10$ is an overestimate.

So, if you want to write something unique, sit down and compose a string of ten characters. But don't write `gidnebinzz`, because I already thought of that string.

As for a new melody, if we focus on the sequence of notes, ignoring duration and stress, and allow leaps of up to an octave at each note, then the number of choices per note is 23. The pitch of the first note is arbitrary. The number of melodies of length r notes in this rather ugly ensemble of Schönbergian tunes is 23^{r-1}; for example, there are 250 000 of length $r = 5$. Restricting the permitted intervals will reduce this figure; including duration and stress will increase it again. [If we restrict the permitted intervals to repetitions and tones or semitones, the reduction is particularly severe; is this why the melody of 'Ode to Joy' sounds so boring?] The number of recorded compositions is probably less than a million. If you learn 100 new melodies per week for every week of your life then you will have learned 250 000 melodies at age 50. Based

on empirical experience of playing the game '`guess that tune`', it seems to me that whereas many four-note sequences are shared in common between melodies, the number of collisions between five-note sequences is rather smaller – most famous five-note sequences are unique.

In `guess that tune`, one player chooses a melody, and sings a gradually-increasing number of its notes, while the other participants try to guess the whole melody.

The *Parsons code* is a related hash function for melodies: each pair of consecutive notes is coded as `U` ('up') if the second note is higher than the first, `R` ('repeat') if the pitches are equal, and `D` ('down') otherwise. You can find out how well this hash function works at `http://musipedia.org/`.

Solution to exercise 12.11 (p.201). (a) Let the DNA-binding protein recognize a sequence of length L nucleotides. That is, it binds preferentially to that DNA sequence, and not to any other pieces of DNA in the whole genome. (In reality, the recognized sequence may contain some wildcard characters, e.g., the `*` in `TATAA*A`, which denotes 'any of `A`, `C`, `G` and `T`'; so, to be precise, we are assuming that the recognized sequence contains L non-wildcard characters.)

Assuming the rest of the genome is 'random', i.e., that the sequence consists of random nucleotides `A`, `C`, `G` and `T` with equal probability – which is obviously untrue, but it shouldn't make too much difference to our calculation – the chance that there is no other occurrence of the target sequence in the whole genome, of length N nucleotides, is roughly

$$(1-(1/4)^L)^N \simeq \exp(-N(1/4)^L), \tag{12.9}$$

which is close to one only if

$$N4^{-L} \ll 1, \tag{12.10}$$

that is,

$$L > \log N / \log 4. \tag{12.11}$$

Using $N = 3 \times 10^9$, we require the recognized sequence to be longer than $L_{\min} = 16$ nucleotides.

What size of protein does this imply?

- A weak lower bound can be obtained by assuming that the information content of the protein sequence itself is greater than the information content of the nucleotide sequence the protein prefers to bind to (which we have argued above must be at least 32 bits). This gives a minimum protein length of $32/\log_2(20) \simeq 7$ amino acids.
- Thinking realistically, the recognition of the DNA sequence by the protein presumably involves the protein coming into contact with all sixteen nucleotides in the target sequence. If the protein is a monomer, it must be big enough that it can simultaneously make contact with sixteen nucleotides of DNA. One helical turn of DNA containing ten nucleotides has a length of 3.4 nm, so a contiguous sequence of sixteen nucleotides has a length of 5.4 nm. The diameter of the protein must therefore be about 5.4 nm or greater. Egg-white lysozyme is a small globular protein with a length of 129 amino acids and a diameter of about 4 nm. Assuming that volume is proportional to sequence length and that volume scales as the cube of the diameter, a protein of diameter 5.4 nm must have a sequence of length $2.5 \times 129 \simeq 324$ amino acids.

(b) If, however, a target sequence consists of a twice-repeated sub-sequence, we can get by with a much smaller protein that recognizes only the sub-sequence, and that binds to the DNA strongly only if it can form a *dimer*, both halves of which are bound to the recognized sequence. Halving the diameter of the protein, we now only need a protein whose length is greater than $324/8 = 40$ amino acids. A protein of length smaller than this cannot by itself serve as a regulatory protein specific to one gene, because it's simply too small to be able to make a sufficiently specific match – its available surface does not have enough information content.