Bitwisdom problem from TopCoder

Neeraj Kashyap

September 3, 2015

1 Problem (taken from TopCoder website)

In this problem we are dealing with a string of N bits, numbered from 0 to $N-1$.

Julia likes the string that consists of N zeros.

If you give her any N-bit string, she will convert it into a string of N zeros using the smallest possible number of actions.

An action consists of selecting an integer k ($1 \leq k \leq N$) and flipping either the first k bits or the last k bits of the string.

Formally, Julia can flip either all bits with number i such that $i < k$ or all bits with number i such that $i >= N - k$.

You are going to generate a random string of N bits.

You are given a $int[]p$ with N elements.

For each i, bit number i has a $p[i]$ percent chance of being a 1 and a $(100 - p[i])$ percent chance of being a 0.

The values of the bits are chosen independently from each other.
After you generate the string, you are going to give it to Julia.

Please find and return the expected number of actions Julia will take.

2 Getting to a solution

I found the statement of this problem a bit difficult to understand, especially because the poser states that we are going to be generating a string according to the given probabilities but then later asks us for the expected number of actions, which would suggest that they want us to calculate the expectation of the number of actions that Julia takes over all possible strings that we could

generate. The second problem sounded a lot more interesting, so that is the one I solved. Anyway, both problems involve the same basic reasoning. The second one involves just a little bit more work.

Here are a few examples I worked out where I performed the task of flipping strings to all-zeros using the actions available to Julia:

- For the string 01101, here is a possible sequence of moves:

  $$01101 \to 10010 \to 00010 \to 00001 \to 00000.$$

  This sequence is inferior to the following:

  $$01101 \to 01100 \to 10000 \to 00000.$$

  Another sequence which achieves the result in 3 moves:

  $$01101 \to 00010 \to 00001 \to 00000.$$

  This example teaches us that we may as well assume that every flip we perform takes place at a bit with value 1 because we will never want to flip a 0 to a 1 if we can help it, which would just create more work for us to do.

- For the string 10101, we can do:

  $$10101 \to 00101 \to 00010 \to 00001 \to 00000,$$

  or

  $$10101 \to 10100 \to 01000 \to 10000 \to 00000.$$

- For the string 00110011, we can do:

  $$00110011 \to 00001100 \to 00000011 \to 00000000,$$

  or:

  $$00110011 \to 00110000 \to 11000000 \to 00000000.$$

  The other principle that becomes apparent here is that all flips will be made at "outside" 1's.

Putting together the observations from these examples, we can deduce that runs of 0's or 1's do not affect the number of moves that Julia will have to make. Thus, we need only consider the number of moves we'd have to make on strings with alternating 0's and 1's of various lengths. Let us call such strings *admissible*.

Note that there will always be a minimal reduction of an admissible string to all 0's which does not begin with flipping every bit of the string from one end to the other. The reason for this is that:

1. We would not begin with a flip of an outside 0.

2. If the string had a 1 on one end, flipping from that end to the other would flip the adjoining 0 to a 1, which would take an additional action to get make a 0 again. Therefore, getting both bits to be 0 in this manner would take a total of two actions at minimum, whereas it would take only one action to simply make the tailing 1 a 0.

We can deduce from this that constructing an admissible string from an admissible string by tacking on an extra bit results in an increase in the number of steps Julia will require to reduce it. Therefore, the number of steps required for reduction of an admissible string is the number of changes in parity we see going from one end to the other. This doesn't cover the all-1 string, which requires one move in order to reduce per Julia's standards

Therefore, we can calculate the expectation as the sum

$$\prod_{i=0}^{n-1} p[i] + \sum_{i=p}^{n-2} \left(p[i](1-p[i+1]) + (1-p[i])p[i+1]\right).$$