

>>> Assignment #6 for Simulation (CIS 4930) <<<

>>> SOLUTIONS <<<

This assignment covers material that may be helpful for the project.

Problem #1 (100 points)

Your task is to come-up with a scheme that predicts the next bit in the series of bits in `bits.txt` (download as <http://www.csee.usf.edu/~christen/class3/bits.zip>) with the lowest possible prediction error. A prediction error occurs when your program predicts a “1” and the next bit is a “0”, or visa versa. Starting with bit 0 you are to predict bit 1, then with bit 1 you are to predict bit 2, and so on for all the bits (1 millions of them) in `bits.txt`. The scaffold program `predictBit.c` (download as <http://www.csee.usf.edu/~christen/class3/predictBit.c>) should be your starting point.

You can imagine that `bits.txt` is the trace of some unknown process that generates bits. Your goal is to be able to predict the next bit that this process generates given only knowledge of previous bits generated.

What is the best possible prediction scheme that you can come-up with? That is, the scheme with the lowest possible prediction error. Can you argue why your scheme is best possible? You are to deliver:

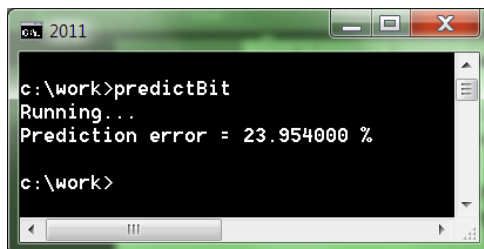
- a) A description (pseudocode or similar) of your prediction method.

25 points for a formal description of their method. This simple method is the best (I believe).

The best prediction appears to be `predictNextBit = currentBit`.

- b) A screenshot of the execution of your modified `predictBit.c` that implements your method.

Here is a screenshot of my execution:



25 points for the execution. The output should be as shown here.

- c) The source code of your modified `predictBit.c`

25 points for the code snippet.

The key change is to the `predict` function, which is:

```
//-----  
// - Function to predict next bit -  
//-----  
int predict(int currentBit)  
{  
    int predictNextBit;    // Predicted (next) bit  
  
    // The prediction method  
    predictNextBit = currentBit;  
  
    return(predictNextBit);  
}
```

A sample execution was also added to the program header

d) An argument or proof of why your method is the best possible for `bits.txt`.

Looking at the sequence of bits it is clear that there is autocorrelation (this from the long strings of consecutive 0's and 1's). Here is the autocorrelation from the first 10 lags (from `autoc.c`):

```
----- autoc.c -----  
Autocorrelation for lag 1 = 0.500951  
Autocorrelation for lag 2 = 0.250955  
Autocorrelation for lag 3 = 0.124560  
Autocorrelation for lag 4 = 0.061415  
Autocorrelation for lag 5 = 0.029525  
Autocorrelation for lag 6 = 0.014337  
Autocorrelation for lag 7 = 0.006370  
Autocorrelation for lag 8 = 0.002052  
Autocorrelation for lag 9 = 0.000502  
Autocorrelation for lag 10 = 0.000229  
-----
```

25 points for the "proof". The student must show/argue why their method is best. I believe that understanding autocorrelation is a key part of this argument. For full credit the student should be able to show why the prediction for their program is correct.

However, if we look at the length of the sequences of 0s and 1s, there appears to be no autocorrelation. Here is the autocorrelation on the length of 0 sequences:

```
----- autoc.c -----  
Autocorrelation for lag 1 = 0.000787  
Autocorrelation for lag 2 = 0.004868  
Autocorrelation for lag 3 = 0.004236  
Autocorrelation for lag 4 = -0.000650  
Autocorrelation for lag 5 = -0.001340  
Autocorrelation for lag 6 = -0.002726  
Autocorrelation for lag 7 = -0.004450  
Autocorrelation for lag 8 = -0.001228  
Autocorrelation for lag 9 = 0.003978  
Autocorrelation for lag 10 = -0.002181  
-----
```

And here for the length of 1 sequences:

```
----- autoc.c -----  
Autocorrelation for lag 1 = -0.003896  
Autocorrelation for lag 2 = -0.002572  
Autocorrelation for lag 3 = 0.003485  
Autocorrelation for lag 4 = -0.000939  
Autocorrelation for lag 5 = 0.000016  
Autocorrelation for lag 6 = 0.003122  
Autocorrelation for lag 7 = 0.000699  
Autocorrelation for lag 8 = 0.003653  
Autocorrelation for lag 9 = -0.000303  
Autocorrelation for lag 10 = 0.006619  
-----
```

This implies that the sequence lengths are independent (and thus no prediction can be made on sequence length). Thus, I believe that the best prediction possible is simply to recognize that the start of a sequence (of 1s or 0s) when it occurs and then predict that the next bit is part of this sequence.

In addition, we can look at the mean length of the zero sequence and one sequences, which are:

```
c:\work>summary1 < zero
----- summary1.c -----
Total of 119771 values
  Minimum = 1.000000 (position = 119770)
  Maximum = 47.000000 (position = 94438)
  Sum      = 600015.000000
  Mean     = 5.009685
  Variance = 19.982824
  Std Dev  = 4.470215
  CoV      = 0.892315
-----

c:\work>summary1 < one
----- summary1.c -----
Total of 119770 values
  Minimum = 1.000000 (position = 119768)
  Maximum = 40.000000 (position = 94560)
  Sum      = 399985.000000
  Mean     = 3.339609
  Variance = 7.732182
  Std Dev  = 2.780680
  CoV      = 0.832636
-----
```

Where the files zero and one contain the lengths of the zero and one sequences, respectively, in bits.txt. And note that we will be “wrong” in our guess twice ever 5.0 + 3.33 bits, which is $2.0 / 8.33 = 0.24$. Note that 0.24 is the error rate observed from running the prediction model.

Hint: Don’t forget that there may be useful tools here: <http://www.csee.usf.edu/~christen/tools/toolpage.html>.
