CS 70 Discrete Mathematics and Probability Theory
Summer 2011 Kamil PA 2

Due Monday, July 18, 11:59pm

You must complete this programming assignment on your own. Remember that you have three slip days that you can allocate between the three programming assignments. Note that there is a short grace period (on the order of minutes, not hours) after the above deadline to account for clock discrepancies, last minute submission hiccups, etc.

This assignment will likely take you longer to complete than PA1, so we recommend that you get started early.

Overview

Recall the error correction scheme from class: given a message consisting of a series of packed $m_1, m_2, \dots, m_n$, the sender constructs the following polynomial:

$$P(x) \equiv m_n x^{n-1} + m_{n-1} x^{n-2} + \cdots + m_2 x + m_1 \pmod{q}$$

for some prime $q > m_i$. The sender then transmits the packets $P(1), P(2), \dots, P(n+k)$, where k is the maximum number of packets lost. The receiver can then use polynomial interpolation to recover $P(x)$, as long as at least n packets arrive, and thereby read off the coeffecients of $P(x)$ to recover the original message.

In this programming assignment, you will implement the decoding scheme. Given a prime modulus q, a value k for the maximum number of errors, and $n+k$ encoded message packets, some of which may be erased, it is your job to recover the polynomial $P(x)$ and the original message. Your program must work for any values of q, k, and n, though you may assume that they are all less than 2^{16} so that you can perform your operations using 32-bit integers. You may use any of the following programming languages: Scheme, Java, Python, C, or C++. (Note that though you may use any of these languages, we can only provide you skeletons in a subset of the languages. It may be beneficial to look over a skeleton for another language if one is not available in the language of your choice.) You have access to `stk`, `javac` and `java`, `python`, `gcc`, and `g++` from your class accounts. Skeletons for Scheme, Java, and Python with some useful auxiliary functions are provided at `http://www.cs.berkeley.edu/~kamil/cs70/pa/pa2/`. We strongly recommed that you use one of the skeletons for this assignment. The skeletons already implement polynomial multiplication, which may be tedious to replicate (and not the point of the assignment!).

For the languages other than Scheme, input will be passed in on the command line, with q as the first argument, k as the second argument, and the remaining arguments representing the encoded message packets $M_1, \dots, M_{n+k}$. A value of -1 for a particular message packet signifies that that packet was erased. The skeletons for Java and Python process these arguments for you.

As output, you should convert the coefficients of the reconstructed polynomial $P(x)$ into a string corresponding to their ASCII values. Thus, a coefficient with a value of 65 corresponds to the capital letter 'A'. (See `http://www.asciitable.com/` for a table of ASCII values.) In most languages you should be able to do the conversion by casting a coeffcient to the `char` type. Thus, the polynomial $P(x) = 33x^2 + 105x + 72$ corresponds to the message "Hi!" You may find functions in the Scheme, Java, and Python skeletons to help you make these conversions. Once you've recovered the original string, you should print it to standard output.

Your program need not perform any error checking (e.g. that each message packet is in the range $[0, q-1]$).

Special Notes for Scheme

If you use Scheme, you should instead define a function called `decode` that takes in three arguments: q, k, and a list of encoded message packets. You should display the recovered string as output. You can find pre-defined functions in the skeleton to process input as well as manipulate polynomials. You may also find the built-in Scheme functions `quotient` and `modulo` to be useful. (Note that while some of the skeleton functions may use features of Scheme that you are unfamiliar with, such as looping, you should be able to complete your solution using just recursion. You are welcome to use more advanced features if you like, however.)

Testing Your Code

We have made a message encoder written in Java available at `http://www.cs.berkeley.edu/~kamil/cs70/pa/pa2/pa2gen.java` to help you test your code. Once you copy it to your class account, you can compile it as follows:

```
javac pa2gen.java
```

You can then run it as follows:

```
java pa2gen q k <message>
```

The `<message>` argument is the plain text message you wish to encode. If it contains spaces, make sure to enclose it with quotes so that it is passed to `pa2gen` as a single argument.

There is also a Python version of the message encoder at `http://www.cs.berkeley.edu/~kamil/cs70/pa/pa2/pa2gen.py`. The Scheme skeleton contains an `encode` function that similarly generates an encoded message.

Submission Instructions

You must submit your assignment electronically with the following command from your class account:

```
submit pa2
```

Two files are required in the directory from which you run the above command. A `README` file described below and a ZIP archive called `pa2.zip`. This archive should include all files necessary to compile and run your program. You can construct this archive using the following command:

```
zip pa2.zip <file1> ...  <fileN>
```

If you are adding directories, you will need to pass the `-r` flag to `zip`. As a sanity check, run

```
unzip -l pa2.zip
```

afterwards to make sure that all intended files are included in the archive. **We will not be able to grade incomplete submissions, so make sure to double check that all files are included.**

The README File

In your `README` file, include explicit instructions on how to build and run your code. In addition, answer the following questions:

(a) Did you prefer writing a program to perform error correction or carrying it out on paper?

(b) How long did this assignment take you to complete?