Algorithms in the Real World (15-853), Fall 09
Assignment #3 Due: 15 October 09

You are not permitted to look at solutions of previous year assignments. You can work together in groups, but all solutions have to be written up individually.

Problem 1: Suffix Trees (10pt)

Consider a suffix tree build over a text string S of length n. Describe how to augment Suffix trees to support each of the following types of queries.

1. FindBefore(P,e) : find a string P in S starting at any position before e, if one exists. This should run in $O(|P|)$ time and the space for the augmented data should be $O(n)$. I briefly described this in class.

2. FindClosest(P,s,e) : find the string P in S that starts before s. If P appears multiple times before s, it must return the closest one to s. This should run in $O(|P| \log n)$ time and the space for the augmented data should be at most $O(n^2)$. If you can reduce the time or space, you can get extra credit (if described concisely and clearly). Note this query can be useful for Lempell-Ziv compression since the closer a match is to the "prompt" the smaller the offset, and hence the better the compression.

3. FindBetween(P,s,e) : find a string P in S that starts between the locations s and e. The bounds should be the same as the previous problem.

Problem 2: Binary versus q-ary Codes (10pt)

In class we showed a lower bound on the number of code bits required to fix s bit-errors for a binary linear code with with k message bits. In particular by arguing that every codeword requires a ball of radius $(d-1)/2$ around it we derived the formula

$$n \geq k + \log(1 + \sum_{i=1}^{s} \binom{n}{i}) \ .$$

Generalize this to q-ary codes.

Problem 3: Reed Solomon (10pt)

Lets say we are using the "systematic" version of Reed-Solomon codes described on page 5 and we are implementing a $(5, 3, 3)_8$ Reed-Solomon codes (i.e. $s = 1$), using the Galois Field GF(2^3) from page 9. We therefore need to evaluate a polynomial at $\alpha^7(1)$ and $\alpha(2)$.

1. Describe a simple way to test if there are one or two errors (don't just repeat what slide 5 says).

2. Given an input string "010 110 000" where the first element (3 digits) represents the highest order coefficient what is the codeword. Please show your work.

3. The Fast Fourier Transform only works well on powers of 2—actually on any size that has only small factors since each factor f has to be done using the f^2 method. Why does this cause a problem for efficiently evaluating polynomials over GF(2^3)? What about GF(2^8) and GF(65537)?

Problem 4: A variant on LDPC codes (15pt)

Consider the following variant on LDPC codes. Like LDPC codes the code is given by a bipartite graph, but now assume that the neighbors for each node on the right must form a proper Hamming code. To be concrete lets assume each vertex on the right has degree 15 and the bits on the neighbors must form a $(15, 11, 3)$ Hamming code. We will assume each vertex on the left has degree $d = 3$ so the number of nodes on the right is $n/5$.

1. What is the rate of this code (i.e. k/n)?

2. Assuming the bipartite graph has expansion (α, β) with $\beta = d/2 = 1.5$ prove that the code has distance at least αn. This is a similar argument to the one given in class for LDPC codes, but for LDPC codes we required that $\beta > d/2$.