

Problem 1: Building A Finite Field (30 pts.)**Background**

As we have seen in class, there is a unique finite field of size p^k for any prime p and $k \geq 1$. Needless to say, the case $p = 2$ it is particularly interesting for actual implementations: the prime field can naturally be represented by bits and the arithmetic operations are given by *xor* (addition) and *and* (multiplication).

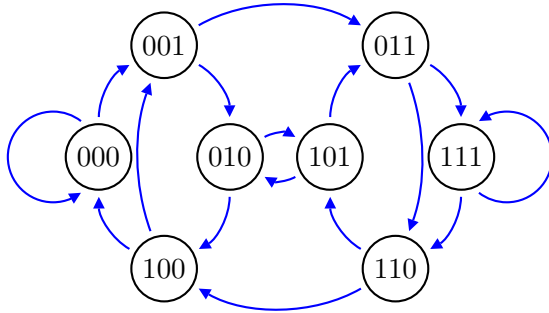
Building a finite field \mathbb{F}_{2^k} requires a little more work.

Task

- A. Show how to construct the finite field \mathbb{F} of size 256. What data structures would you use, how would you implement arithmetic in this field?
- B. How many primitive polynomials and primitive elements are there in this field?
- C. What are all the subfields of \mathbb{F} ? Why?
- D. If we had constructed a field of size 32, what would the subfields be?
- E. What is the main difficulty in constructing the field of size 2^{1024} ?

Problem 2: Hamiltonian Sequences (30 pts.)**Background**

A de Bruijn sequence of order k is a bit-sequence H of length $2^k + k - 1$ that contains every bit-sequence of length k as a factor. E.g., $H = 0001110100$ is a de Bruijn sequence of order 3. One way of generating such sequences is to construct a de Bruijn graph B_k of order k , and then to trace a Hamiltonian cycle in B_k (and flattening out the node sequence to a bit-sequence). Recall that a Hamiltonian cycle is a cycle that uses every node exactly once. Here is a picture of B_3 .



A useful idea in this context is that of an Eulerian cycle: a cycle that uses every edge exactly once (edge, not vertex!). It is easy to see that a directed graph is Eulerian if and only if it is connected and every node has the same in-degree and out-degree.

Task

1. What is the relationship between Eulerian and Hamiltonian cycles for the family of de Bruijn graphs?
2. Explain why these graphs are Hamiltonian.
3. Find a linear time algorithm to construct an Eulerian cycle.
4. Now explain how to construct a de Bruijn sequence of order k , for any k .
5. What is the running time of your algorithm as a function of k ? What is the memory requirement? You have to account for the cost of constructing the graph, as well as the cost of finding the cycle.

Comment

Testing whether a general directed graph is Hamiltonian is very difficult, no polynomial time algorithm is known (nor is one likely to exist). But Eulerianess (Eulericity?, Eulerhood?) is very easy to check.

For the Eulerian cycle algorithm you may assume that the graph is given in adjacency list form. The running time of your Eulerian cycle algorithm must be $O(n + e)$, the size of the adjacency list, where n is the number of nodes of the graph, and e the number of edges. The standard graph-theory argument that one can simply remove a cycle, assume inductively that one already has an Eulerian cycle for the remainder of the graph, and then glue the two cycles together is not good enough algorithmically: you have to explain exactly how to do this in linear time.

Problem 3: Analyzing LFSRs (40 pts.)**Background**

Consider LFSRs with characteristic 2 and span k (as always, assume that there is a tap at the last register). Then the global map is injective and the diagram of the global map is just a decomposition of $\mathbf{2}^k$ into disjoint cycles. For large span, experimental verification of the properties of LFSRs is computationally infeasible, but for small k one can easily compute these cycles. Hence we can determine, for each given LFSR, the *average cycle length* and the *maximum cycle length*.

Task

- A. Find all LFSRs of span $k = 10$ that maximize the maximum cycle length.
- B. Find all LFSRs of span $k = 10$ that maximize the average cycle length.
- C. Any observations regarding items (A) and (B)? What is going on?
- D. Find all LFSRs of span $k = 10$ that minimize the average cycle length. Hint: there is exactly one.
- E. Explain how this LFSR from part (D) works and how one can compute the average cycle for this type of LFSR directly for all k , without simulation.

Comment

Unless you are Gauss or von Neumann you will probably want to write a program that computes the cycle decomposition for the LFSRs in question; doing this by reasoning alone is hard. For part (E) you do not have to prove that the LFSRs there are in fact always the ones that minimize the average cycle length, just figure out how to compute the average cycle length for the special LFSR discovered in part (D).