1 Blind Counting II

See “Balancing cyclic R-ary Gray codes” by Fahive and Bose(2007).

1.1

What does it mean formally to read an n-bit number?

As we did last time, we build a tree to create an encoding, with the number of the read bit assigned to a node.

We will define the worst reading length as the maximum depth of the tree representing the encoding.

The problem is that it is memory-wise expensive to draw out the entire tree. What procedure may allow us to compress and simplify the process?

Imagine that we have two leaves with the same decision. Why do we need the tree structure at all? We can construct a directed acyclic graph! For example, these two leaves can be glued into one, since they represent the same outcome. Moreover, we can get rid of the *long paths* by shortening the sequence leading to the final state. In general, we try to deduplicate our tree.

Another method is to build our graph in such a way that positions are read in the order of their significance.

It is worthwhile to note that in this newly constructed graphs two edges leading to the same leaf by construction.

How effective can our compression method be in the worst case?

Not that effective. However, if our condition is a bit better, the efficiency of compression does improve our situation.

One of the methods allowing us to traverse the decision tree is always going to the left until we stumble upon the false decision, in which case we change the direction and go to the right.

It turns out that ordered and compressed decision tree is very easy to combine.

Suppose we have two diagrams. What can we do with them, if we want to calculate some arbitrary boolean function? One of the ways is to compute the function on the respecitve leaves from each tree, and then combine the result.

Note that we have two dimensions of interest to us – the total size and depth of a tree.

Suppose we have two vectors, each of n bits, and we take a dot product modulo 2. How can we represent it in a tree?

Suppose that we are given $n+\log n$ bits, and we assign a bit to the position corresponding to the number already read from the left. In this case, we cannot compress the tree starting from the leaves due to the fact that we do not know anything until we read the entire number.

Let’s count the fraction of functions $\mathbb{B}^{l^2} \to \mathbb{B}^l$ such that, after reading l positions, we can surely identify if some decision is impossible. The upper bound can be computed to be $l^{2l}2^l2^l(1-\frac{1}{2^l})2^{l^2-l}$.