Math 343, lecture 14

① Minimal change orderings

So far for combinatorial generation we've discussed

-
-
-
-

Now lets spend a few day focusing on latter successor functions.

Successor functions are particularly useful for

What we need is

minimal change ordering

(2) Gray codes

The most famous minimal change orderings are the ones for the class of binary strings

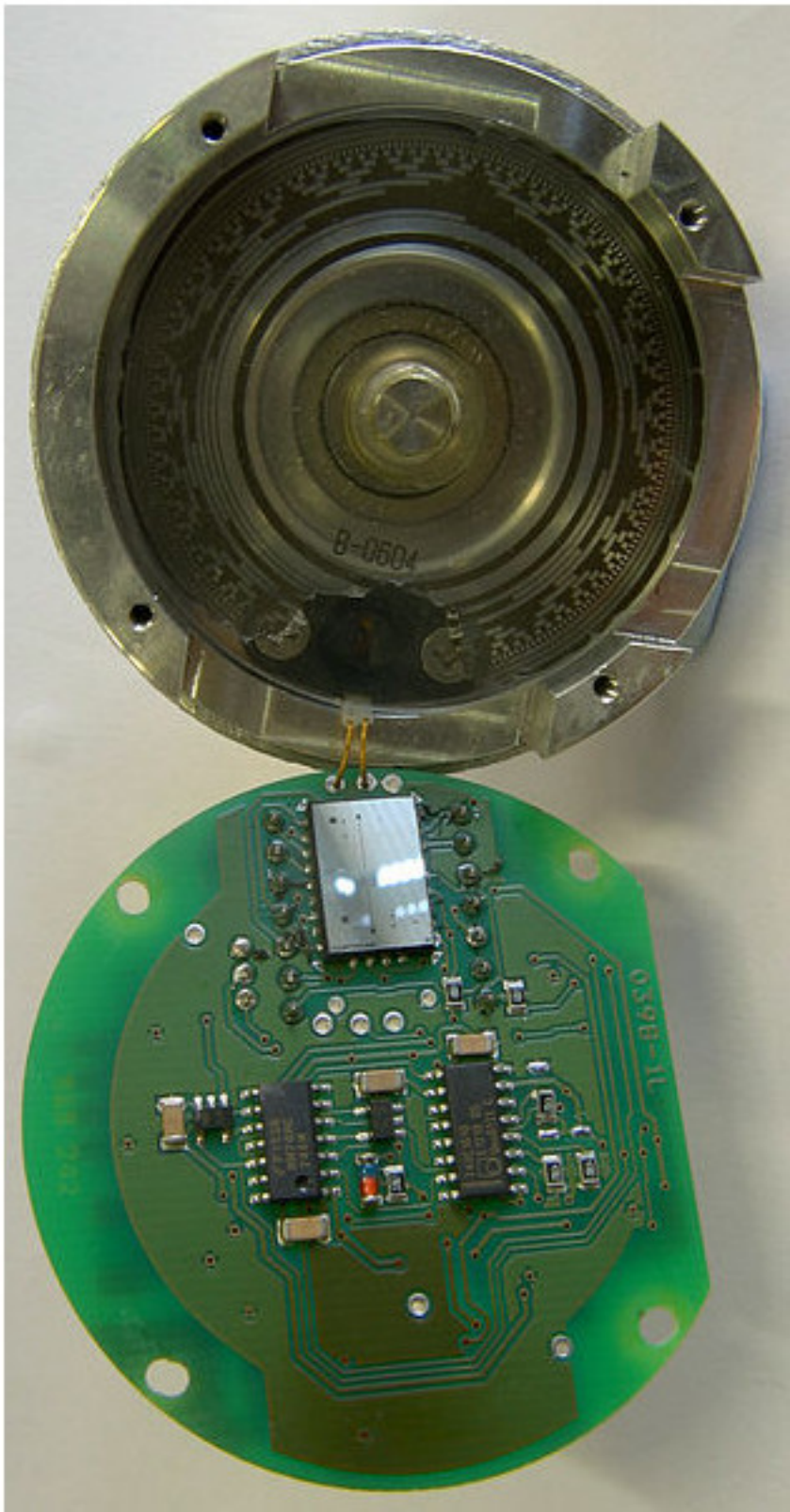
Def Let $w = (a_1 a_2 \dots a_n)$ $w' = (b_1 \dots b_n)$ be two binary strings of length n then the (Hamming) distance $d(w, w')$ between w and w' is

Note $d(w, w')$ is

What we would like is

Gray codes.

Gray codes are very useful in practice.



(Image from Wikipedia "Rotary Encoder")

There is more than one possible Gray code for each n

To see this

eg $n=3$

In general

Any Hamiltonian path of this graph will give a Gray code

An important Gray code is reflected binary Gray code which is defined as follows.

Def Let $R(1) = 0, 1$

Write $R(j)_i$ to denote the ith element in the order $R(j)$. Then define recursively

$R(n) = \{$

note:

eg $R(2) =$

$R(3) =$

Note

cyclic

Prop $R(n)$ is a cyclic Gray code

pf There are two things we need to show. First that every binary word appears and second that the distances are always 1 between successive elements and between the first and last.

We will prove this by induction on n.

For $n = 1$

Take $n \geq 2$, assume the result holds for $n-1$ and consider $R(n)$.

Def Given a binary string w let $d(w)$ be the number of 1s in w. This is called the Hamming weight of w

This result gives the following algorithm

Algorithm GraySuccessor

input n, w w a binary word of length n

result = w

eg $n=4$ $w=1110$

eg $n=4$ $w=1010$

And for comparison the full order for $n=4$ is

```
input n, w
result = w
if d(w) is even
    flip last bit of result
else
    j = n
    while result(j) = 0 and j > 0
        j = j - 1
    if j = 1
        return no successor
    flip (j-1)th bit of result
return result
```

We can prove by induction that this algorithm works.

First observe it works for $n=1$

Now assume it works for some $k \geq 1$ and consider $n=k+1$

Let w be a binary string of length n

Let $w = \varepsilon w'$ where $\varepsilon = 0$ or 1 and w' is a binary string of length $n-1$.

If $\varepsilon = 0$

If $\varepsilon = 1$

If $d(\omega)$ is even

If $d(\omega)$ is odd

The algorithm makes it clear that not every bit is changed the same number of times — the last bit is changed $\frac{2^n}{2} = 2^{n-1}$ times but the first bit is only changed once, twice if we consider it cyclically.

We could ask for a cyclic Gray code with balanced transition counts, that is every bit is changed (cyclically) exactly $\frac{2^n}{n}$ times

This could make a good project for someone. There is an 11 page paper from 1996 which gives the construction.

③ Rank and unrank for reflected binary Gray code.

Consider the following table

rank	rank in binary	element of R
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

take a row

eg

then observe

Lemma Let $n \geq 1$, $0 \leq r \leq 2^n - 1$.

Let $a_{n-1} a_{n-2} \ldots a_0$ be the r^{th} binary string in Gray code order

Let $b_{n-1} b_{n-2} \ldots b_0$ be r as a binary string

Then

$$a_j = b_j + b_{j+1} \mod 2 \qquad \text{(by convention } b_n = 0\text{)}$$

$$b_j = \sum_{i=j}^{n-1} a_i \mod 2$$

Proof First prove the first equation by induction on n

For $n=1$ just check

Suppose the result holds for $k \geq 1$ and consider $n = k+1$

If $r \leq 2^{(k+1)-1} - 1$

If $r \geq 2^{i-1}$

Now for the second equation we can use the first and some telescoping

This gives the following algorithms

Algorithm GrayRank

input n, w w binary string of length n

$r = 0$

$b = 0$

eg GrayRank (3, 101)

Algorithm Gray Unrank

input n, r

$w = \underbrace{0 \ldots 0}_{n}$

$b' = 0$

eg GrayUnrank(3, 6)

④ Next time

Minimal change orderings for k subsets