

Assignment 1 – alternative binary representations

due in class on Mon 10 Feb (40 points)

This assignment is an activity for groups of three. We'll work on it in class on Wed 29 Jan, and then your group must submit **one** response before the deadline.

Introduction

We have learned about using binary (base two) as a positional numbering system, where the values of the columns from right to left are powers of two: 1, 2, 4, 8, 16, and so on. In some kinds of systems, however, it can be useful to invent alternative representations of numbers as bits. This assignment explores two such alternatives.

The odometer effect

One feature of positional numbering systems is what I like to call the *odometer effect*. This is what happens when a mechanical mileage counter on a car reads something like 9999 and then “flips over” to 10000. All the dials have to turn at the same time.



Figure 1: A mechanical car odometer [\[mulears on Flickr\]](#)

The odometer effect happens much more often in binary. Incrementing any odd number ends up flipping more than one bit at a time. Here are some examples:

```
increment 3 to 4: 011 to 100 (3 bits change)
increment 5 to 6: 101 to 110 (2 bits change)
increment 7 to 8: 0111 to 1000 (4 bits change)
```

In some electro-mechanical systems, flipping so many bits on every increment can be awkward – perhaps it uses a lot of power, or it produces excessive vibration. Imagine counting through all 32 configurations of a sequence of 5 light switches. These transitions would seriously slow you down!



Figure 2: An array of switches [[randomcuriosity on Flickr](#)]

The punch card

Another type of system where an alternative binary representation can help is one with **punched cards**. Paper with holes punched (or not) in particular locations can easily represent a binary code. These devices have a long history, from the Jacquard loom (in 1800) to automatic pianos and census cards (1900) to mainframe computers (1950s).

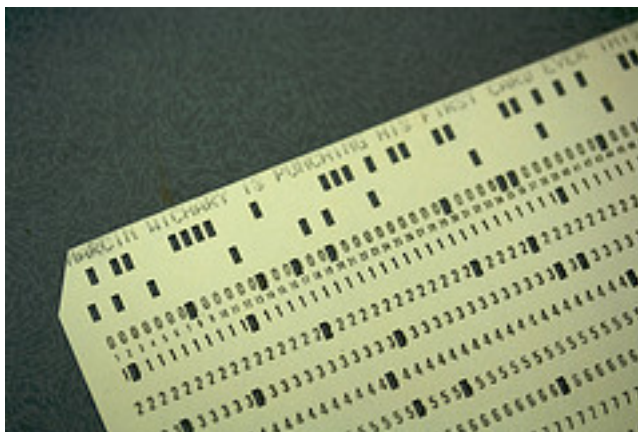


Figure 3: A punch card with a text message [[Marcin Wichary on Flickr](#)]

If you punch too many holes in a card or paper tape, then it can lose its structural integrity and might rip or fall apart in some machine that processes it. So a card that contains the binary number 1111111 is not a great idea.

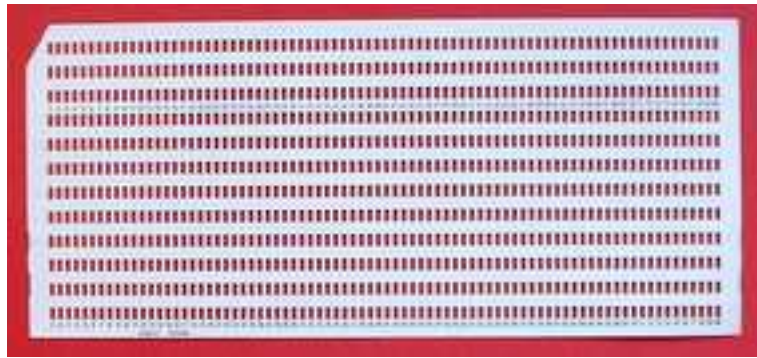


Figure 4: A ‘lace’ card, sometimes created as a practical joke to jam a card reader [\[Wikimedia Commons\]](#)

For this reason, binary encodings developed for punch cards limited the number of ‘on’ bits (ones) per column. In the cards shown here, each column represents one character. The cards have twelve rows that are punchable – the top ones usually called X and Y, and then those below numbered 0–9.

What to do

0. Write the names of everyone in your group here.

1. Develop a binary representation of the numbers 0–3 (two bits each) that **avoids** the odometer effect. In each transition, **exactly one bit** should change. Arrange the numbers in a circle, so that the transition from 3 back to 0 also changes just one bit. Use the red-white counter chips and the attached worksheets to help you visualize the representation.

0 =	__ __	1 =	__ __
2 =	__ __	3 =	__ __

2. Expand your representation to the range 0–7 using three bits each. Again, in each transition (including 7 back to 0), exactly one bit should change.

0 =	__ __ __	1 =	__ __ __
2 =	__ __ __	3 =	__ __ __
4 =	__ __ __	5 =	__ __ __
6 =	__ __ __	7 =	__ __ __

3. Once more, expand your representation to the range 0–15 using four bits each. This time, try to make sure the representation follows some pattern that can be derived from smaller ranges.

0 =	__ __ __ __	1 =	__ __ __ __	2 =	__ __ __ __	3 =	__ __ __ __
4 =	__ __ __ __	5 =	__ __ __ __	6 =	__ __ __ __	7 =	__ __ __ __
8 =	__ __ __ __	9 =	__ __ __ __	10 =	__ __ __ __	11 =	__ __ __ __
12 =	__ __ __ __	13 =	__ __ __ __	14 =	__ __ __ __	15 =	__ __ __ __

4. Describe a **general pattern** that one can use to create a representation by using one from a smaller range. For example, how would you make a 7-bit representation using the 6-bit one?

5. Now we switch to punch card encodings. Using 4 bits, **how many** distinct representations are there such that **at most two** of the four bits are ones. For example, 1000 and 0011 are a valid representations, but 1011 is not.

6. Using 5 bits, **how many** distinct representations are there such that **at most two** of the five bits are ones?

7. Try to come up with a **general pattern** or formula that determines how many distinct **N-bit** representations there are with at most two ones. Apply your pattern to determine these values:

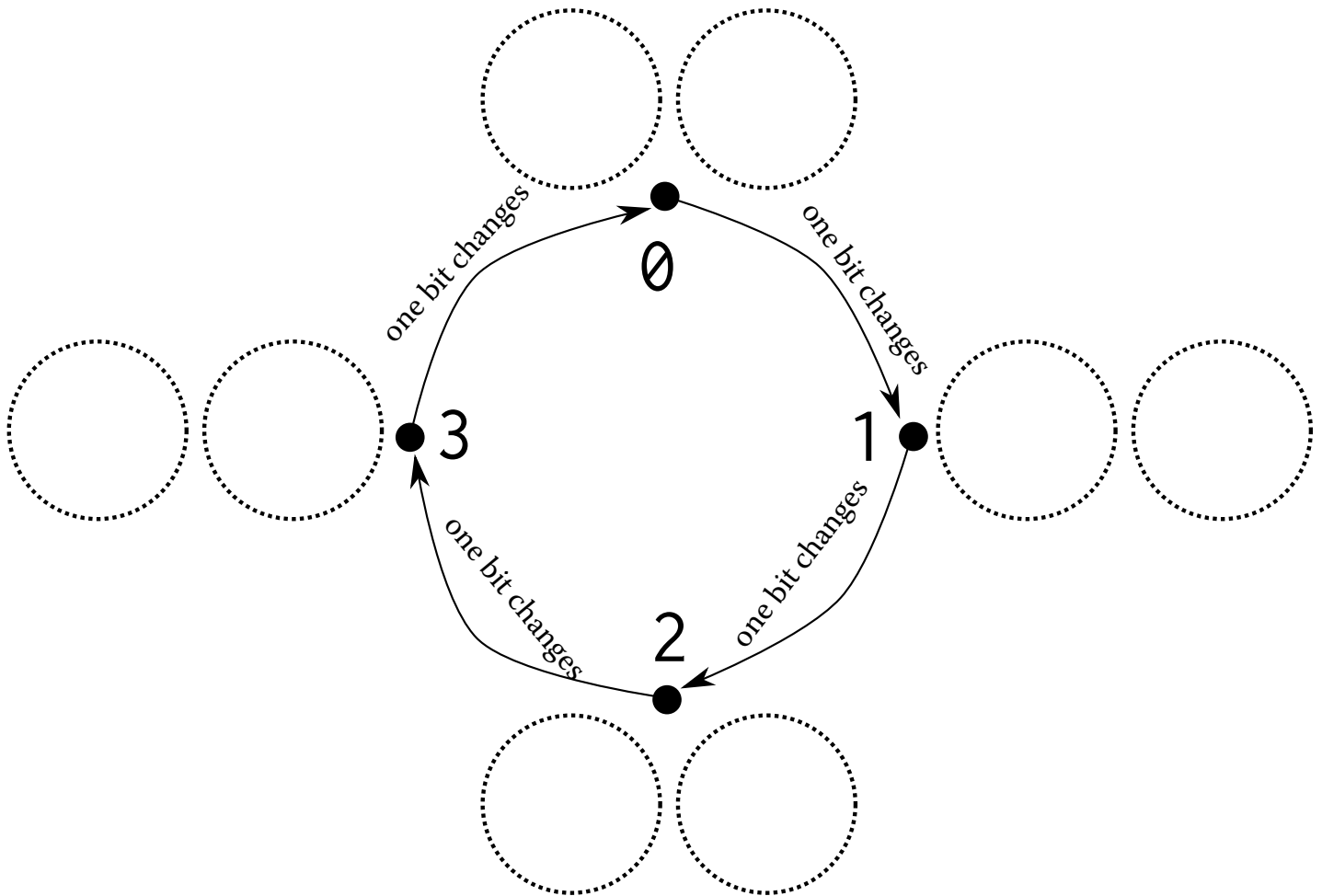
- Number of distinct 6-bit patterns: ____
- Number of distinct 7-bit patterns: ____
- Number of distinct 8-bit patterns: ____
- Number of distinct 9-bit patterns: ____
- Number of distinct 10-bit patterns: ____

8. **Instead of** requiring that at most two of bits are ones, let's invent a representation where there are **no consecutive ones**. For example, with 5 bits, 10101 would be allowed, but not 11000.

- Number of distinct 4-bit patterns: ____
- Number of distinct 5-bit patterns: ____
- Number of distinct 6-bit patterns: ____

A 2-bit uniform-change code

Place red/white counter chips on the dotted circles. Let the white side represent zero, and the red side represent one. With each transition (marked by the arrows), exactly one bit should change color.



A 3-bit uniform-change code

