It's All Just Bits

Notes on Digitization for MA/CS 109

Leo Reyzin with the help of Nick Benes and Ray Sweha

Defining "Computer Science" is tricky, and there are many competing definitions. My view is that Computer Science is primarily concerned with "*HOW* to do things with information?" In fact, many languages don't even call our discipline Computer Science; for example, it's "informática" in Spanish, "Informatik" in German, and "informatique" in French.

It's usually a good idea to start simple. What is the simples information we can have? An answer to a yes/no question seems like the simplest information we can have. We will represent a "no" as 0, and "yes" as 1. We will call a single digit that can be either 0 or 1 a *bit*. The term "bit" was coined by John Tukey at AT&T's Bell Labs in 1947 and stands for binary digit.

We will now proceed to demonstrate that a large variety of information can represented as bits. This process is known as digitization. It has been transforming our society at a rapid pace and its implication are still not well-understood. In fact, our intuition, used to thinking of information in the analog world, often lets us down when dealing with digital information. We will provide some examples of that below.

Numbers are (just) bits

How is

10100101

a number?

We treat each bit position like a coin worth higher and higher powers of 2, from right to left: $1 = 2^0$, $2 = 2^1$, $4 = 2^2$, $8 = 2^3$, ... Then we add the coins corresponding to the positions where the bit is equal to 1.

128	64	32	16	8	4	2	1	
1	0	1	0	0	1	0	1	$= 128 + 32 + 4 + 1 = 165$
0	1	1	0	0	1	0	0	$= 64 + 32 + 4 = 100$
0	0	0	0	0	0	0	1	$= 1$
0	0	0	0	0	0	1	0	$= 2$
0	0	0	0	0	0	1	1	$= 2 + 1 = 3$
0	0	0	0	0	1	0	0	$= 4$
0	0	0	0	0	1	0	1	$= 4 + 1 = 5$
0	0	0	0	0	1	1	0	$= 4 + 2 = 6$
0	0	0	0	0	1	1	1	$= 4 + 2 + 1 = 7$
0	0	0	0	1	0	0	0	$= 8$

Notice that while we're counting up, each time we fill up the tail portion with 1s, we need a new coin. So $11 = 3$, but we need a new coin (digit) to get to 4.

It should be fairly clear how to convert every bit string to a whole number. It is less clear, at this point, how to convert a whole number to a bit string, or even why every whole number can be represented as a bit string. You will be doing this on the homework, and

we will be discussing it in a few lectures. For now, to develop some intuition, observe that $1+2=4-1$, $1+2+4=8-1$, $1+2+4+8=16-1$, $1+2+4+8+16=32-1$, and so on. Generally, $1+2+4+\cdots+2^{k-1}=2^k-1$ (there are a few different ways to prove this—induction is one of them). In other words, the biggest number that can be possibly made using k bits is just one below the value of the $(k+1)$th coin. So just as you made all you can make using k coins, the next coin comes to the rescue. Also, observe that the number of possible bit strings of length 1 is 2: the string 0 and the string 1. The number of possible bit strings of length 2 is 4: the strings $00, 01, 10$, and 11. In general, each time you increase the length by one, the number of strings doubles (again, this needs to be proven, and induction works). So the number of bit strings of length k is 2^k. In fact, as you will show on the homework, every integer between 0 and 2^k-1 can be represented with a k-bit string (equivalently, it takes $\lceil \log_2 N \rceil$ bits to represent any positive integer less than N). So the eight coins shown above can represent any integer between 0 and 255.

Text is (just) bits

In fact, letters have numerical code, and numbers, as we now know, are bits.

There is a table of numerical codes for letters and other symbols, called ASCII (American Standard Code for Information Interchange, presented in Figure 1)

1 ┌	33 !	65 A	97 a	129 ü	161 ¡	193 Á	225 á
2 ┐	34 "	66 B	98 b	130 ,	162 ¢	194 Â	226 â
3 └	35 #	67 C	99 c	131 ƒ	163 £	195 Ã	227 ã
4 ┘	36 $	68 D	100 d	132 „	164 ¤	196 Ä	228 ä
5 ├	37 %	69 E	101 e	133 …	165 ¥	197 Å	229 å
6 ─	38 &	70 F	102 f	134 †	166 ¦	198 Æ	230 æ
7 •	39 '	71 G	103 g	135 ‡	167 §	199 Ç	231 ç
8 ◘	40 (	72 H	104 h	136 ˆ	168 ¨	200 È	232 è
9	41)	73 I	105 i	137 ‰	169 ©	201 É	233 é
10	42 *	74 J	106 j	138 Š	170 ª	202 Ê	234 ê
11 ♂	43 +	75 K	107 k	139 ‹	171 «	203 Ë	235 ë
12 ◘	44 ,	76 L	108 l	140 Œ	172 ¬	204 Ì	236 ì
13	45 -	77 M	109 m	141 ü	173 -	205 Í	237 í
14 ♫	46 .	78 N	110 n	142 Ž	174 ®	206 Î	238 î
15 ☼	47 /	79 O	111 o	143 ü	175 ¯	207 Ï	239 ï
16 ►	48 0	80 P	112 p	144 ü	176 °	208 Ð	240 ð
17 ◄	49 1	81 Q	113 q	145 '	177 ±	209 Ñ	241 ñ
18 ↕	50 2	82 R	114 r	146 '	178 ²	210 Ò	242 ò
19 ‼	51 3	83 S	115 s	147 "	179 ³	211 Ó	243 ó
20 ¶	52 4	84 T	116 t	148 "	180 ´	212 Ô	244 ô
21 §	53 5	85 U	117 u	149 •	181 µ	213 Õ	245 õ
22 ▬	54 6	86 V	118 v	150 –	182 ¶	214 Ö	246 ö
23 ↨	55 7	87 W	119 w	151 —	183 ·	215 ×	247 ÷
24 ↑	56 8	88 X	120 x	152 ˜	184 ¸	216 Ø	248 ø
25 ↓	57 9	89 Y	121 y	153 ™	185 ¹	217 Ù	249 ù
26 →	58 :	90 Z	122 z	154 š	186 º	218 Ú	250 ú
27 ←	59 ;	91 [	123 {	155 ›	187 »	219 Û	251 û
28	60 <	92 \	124 |	156 œ	188 ¼	220 Ü	252 ü
29	61 =	93]	125 }	157 ü	189 ½	221 Ý	253 ý
30	62 >	94 ^	126 ~	158 ž	190 ¾	222 Þ	254 þ
31	63 ?	95 _	127 ⌂	159 Ÿ	191 ¿	223 ß	255 ÿ
32	64 @	96 `	128 €	160	192 À	224 à	

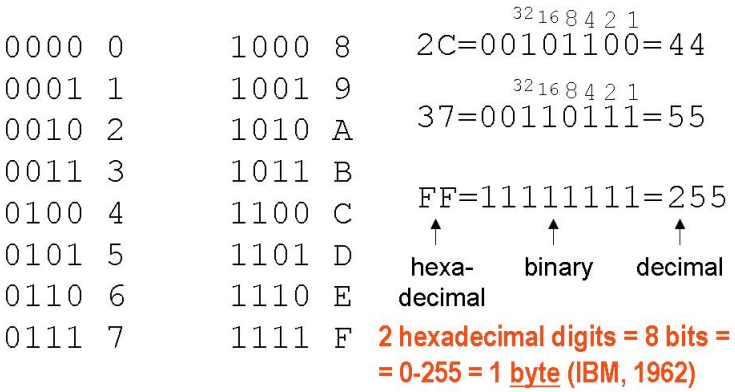
Figure 1: ASCII Table from `http://people.revoledu.com/kardi/resources/Converter/ASCII-table.html`

The table only goes up to 255, because it was decided at some point (apparently, at IBM

in 1962) that a computer will work with chunks of 8 bits at a time. 8 bits are called a *byte* (the smallest amount the computer can *byte* off).

Seeing bits of a file

Writing out individual bits is a pain. We have a shorthand for 4 bits at a time:



Notation: Writing with 0's and 1's is called *binary*. Our good old 0 through 9 is called *decimal*, and adding A through F is called *hexadecimal*. A byte can be represented by 8 bits or two hexadecimal digits.

There is software that allows you to see bytes of a file, typically displaying each byte in hexadecimal notation. For example, we can look at a plain-text (.txt) file and see the ASCII code of each letter. A MSWord document (.doc) has the text in ASCII, as well as other bytes containing information that MSWord uses for formatting (such as which fonts to use, where to make the text bold, etc). The meaning of those bytes is understood by MSWord. However, we can look at it with other software, such as the software that allows you see the actual bytes of a file, and sometimes find surprises. For instance, when you delete something in Word, it sometimes doesn't actually delete the text, but merely marks it as deleted in the formatting information (just like it can mark the text bold, or 10-point, etc.). This may be done for the convenience of the user (it allows you to see who changed what, comment on the changes, and undo them), or for other considerations—for instance, to make the program run faster.

Not surprisingly, the disconnect between what see in MSWord and what is actually in the file may lead to problems. For example, in the public UN report on the investigation into the assassination of Rafik Hariri, the Word document mentions unnamed "senior Lebanese and Syrian officials" allegedly involved in the plot. But looking at the actual bytes of the .doc file, we can see the actual names of the officials. Someone deleted the names during the editing of the report, but the names are still there, just hidden.

The problem is not unique to MSWord. A similar example for a Adobe Acrobat is a US military report on the accidental killing of an Italian intelligence agent Nicola Calipari by US troops in Iraq. The .pdf file is redacted (i.e. classified information is blacked out to produce the publicly released unclassified version), but if we cut and paste the redacted paragraphs into a word processor, we can see the Secret information that had been blacked

out. Someone simply added a layer of black rectangles on top of the words—but the words are still there.

These examples demonstrate that digital information is often hard to get rid of, which contradicts our intuition, developed for analog information, about what it means to delete information. More generally, the meaning of particular bits (such as the bits of an MSWord document or an Adobe .pdf file) depends on the program that is used to display them to us.

Sometimes, the meaning of bits can be lost, because the program to interpret them may be gone. Just try opening a document from a 1980s word processor on today's computers. For a more prominent example, consider the BBC Domesday Project, which produced a digital record of 1986 United Kingdom (getting input from over a million people) to commemorate 900th anniversary of the completion of the Domesday Book, a comprehensive survey of England completed in 1086. In 15 years, the computers and the software to read the laser discs were gone, and the encodings that were used to convert information into bits were forgotten. It took a good deal of effort to reconstruct them. This example highlights the (yet unsolved) challenges of digital preservation. Book preservation has its own challenges, but the 1086 Domesday is still readable today—almost 1,000 years later.

Picture are (just) bits

We perceive color because our eye has sensors (called "cones") for red, green, and blue light. Thus, to display a color to a human, we need to specify the intensity of its red, green, and blue (RGB) components. We can use numbers for that. Typically, we use one byte (i.e., intensity between 0 and 255) per component, thus using three bytes, or 24 bits, to encode a color. See, e.g., `http://colormixers.com/mixers/cmr/`. Because there are 24 bits, this representations allows for $2^{24} = 16,777,216$ possible colors. Another way to see the number of possible colors is to use the multiplication principle: observe that each of the three components has 256 possible values independent of the other components—thus, there are $256 \cdot 256 \cdot 256 = 16,777,216$ possible combinations of the three components.

What about an entire picture? A picture like the one in Figure 2
is made up of tiny individual squares, or *pixels*. When people speak of a resolution, they usually speak of the number of pixels. For instance, a 6-megapixel digital camera may have 2,000 rows and 3,000 columns of pixels, for a total of 6 million ("mega" means million). A 1,200 dots-per-inch (dpi) printer puts 1,200 by 1,200 pixels on every square inch of paper.

Each pixel is colored with a single color (which, as above, consists of three components—red, green, and blue). This means that to record a picture as bits, we need to record the red, green, and blue values for each pixel. A bitmap (.bmp) file does just that:

```
38 53 4c 38 54 4d 39 54 4d 39  SL8SL8SL8SL8SL8SL8SL8SL8TM9TM9
37 55 50 37 55 50 37 55 50 37  UN:UN:VO;VP9UO8UP7UP7UP7UP7UP7
35 53 4e 35 54 4f 36 56 4f 36  UP7UP7RM4RM4SN5SN5SN5SN5TO6VO6
36 57 49 36 55 47 34 55 47 34  WL8YK8XJ7WI6WI6WI6WI6WI6UG4UG4
3a 57 4e 3a 57 4e 3a 57 4e 3a  UG4VH5VH5WI6WI6VK7YN:WN:WN:WN:
39 57 4e 3a 57 4e 3a 57 4e 3a  WN:WN:WN:WN:UL8VM9VM9WN:WN:WN:
34 52 49 35 54 4b 37 55 4c 38  WN:VM9[R>YP<UL8RI5QH4RI5TK7UL8
33 4e 46 2f 51 4b 2e 53 4e 2f  SJ6PG3OF2OF2QH4RI5PG3NF/QK.SN/
33 4e 44 33 4e 44 32 4e 44 32  UN3WP5WO8TL5QG5OE3ND3ND3ND2ND2
2f 57 4c 31 57 4e 33 59 50 35  NE1NE1NF/NG.SG/TI.UJ/WL1WN3YP5
```

Figure 2: An Example Picture (source and subject to be revealed later in the text)

Notice that nearby pixels will be about the same color, so they should have about the same sequence of 3 bytes repeated over and over. For instance, after the cursor we see "57 4e 3a" repeated several times. The beginning of a bitmap file contains a small amount of control information, such as resolution, the number of bits per color, etc. Other so-called "raw" or "uncompressed" formats (described, for example, at `http://en.wikipedia.org/wiki/Raw_image_format`) work similarly.

Color can be represented in other ways than one byte per color component for three components. For instance, we can just use one bit per pixel: 0 if it's black and 1 if it's white. This will give us a black-and-white picture. The number of bits used for each pixel is called *color depth*.

The size a bitmap file (or other raw image file) can computed quite simply: it is the number of pixels times the number of bits per pixel, plus the small amount needed for the additional information (usually this additional amount is tiny compared to the bulk of the file, so we will ignore it in subsequent calculations). For example, a $2,000 \times 3,000$-pixel picture at 24-bit color depth will take up a little over $2,000 \cdot 3,000 \cdot 24 = 144,000,000$ bits, or $18,000,000$ bytes. This is 18 megabytes or 17.166 megabytes, depending on whether you view a megabytes as $1,000,000$ bytes or $2^{20} = 1,048,576$ bytes (see "How many bits" section below).

The higher the resolution or color depth, the larger the file is. Let us consider see the rate of growth of the file size as a function of color depth and resolution.

First let us study the effect of color depth on the file size. Figure 3 shows the same picture with 4 different color depths. In the top left corner the color depth is one, i.e. each pixel has one bit to denote its color, in this case black or while. If it is a $2,000 \times 3,000$-pixel image, it will take $2,000 \cdot 3,000 \cdot 1 = 6,000,000$ bits, or $750,000$ bytes. Note that this is 24

Figure 3: The same picture in 2-color (top left), 16-color (top right), 256-color (bottom left), and 16,777,216-color (bottom right) versions

times smaller than the original image.

In the top right corner the color depth is four, i.e. each pixel has four bits to denote its color, which means that there are $2^4 = 16$ possible colors. This increase in color depth quadruples the size of the file (because it is now $2,000 \cdot 3,000 \cdot 4$ bits), but gives us sixteen times more colors to represent the picture.

The bottom left picture has color depth 8; thus each pixel can be any of $2^8 = 256$ different colors. Thus increases the file by another factor of 2, because it is now $2,000 \cdot 3,000 \cdot 8$ bits. Thus, doubling the file size allowed us to go from 16 to 256 colors.

The bottom right picture has the original color depth of 24 bits (a byte for each of the three color components: red, green and blue), which means that each pixel can be one of 2^{24} (over 16 million!) different colors. Thus, as we went from the 1 bit per pixel to 24 bits per pixel, we increased the file size by a factor of 24, but we got a picture with 16 million colors rather than just 2 colors. In general, because k bits can represent 2^k different colors, the number of colors grows exponentially with the number of bits per pixel. Conversely, because X colors can be represented by $\log_2 X$ bits, the file size grows logarithmically in of the number of colors.

The effect of resolution on the file size is quite different. For example, if we double the resolution of a from $2,000 \times 3,000$ image to $4,000 \times 6,000$, the number of pixels will go up from $6,000,000$ to $24,000,000$, which means the file size will be quadrupled. In general, if we increase the number of pixels per row by a factor of c and the number of pixels per column by a factor of c, we will increase the total number of pixels and, thus, the file size,

by a factor of c^2. Thus, the file size grows quadratically with (linear) resolution.

We often see other picture formats, such at the commonly used JPEG (.jpeg or .jpg). They are more complicated—essentially, they take information in a bitmap and compress it to save on file sizes. However, logarithmic growth as a function of the number colors and quadratic growth as a function of resolution typically remain for other formats. Other formats are decompressed into a bitmap in order to be displayed or printed.

Sound is (just) Bits

Sound is just air pressure acting on our eardrum. We can view it as a graph of pressure as a function of time:

In analog systems, we just record this graph by etching it into a vinyl disc (LP) or magnetizing a tape according to the shape of the graph. To turn it into a digital recording, we can write down numerically where the graph is (i.e., the value on the vertical axis) at regular intervals. Thus, instead of recording a continuous graph, we record a sequence of points. Each point is just a number, and we already know that numbers can be recorded as bits. Then, to reconstruct the original graph, we simply connect the points, like in a child's game of connect-the-dots.

The more frequent we make the points, the more closely we can get to the original graph (see Figure 4 On a stereo CD, there are 44,100 points per second for the left channel and the same number of points for the right one. (Because the human ear does not hear frequencies much above 20,000 Hertz, recording more points per second will not result in an audible difference; do some research on the Nyquist criterion if you want to learn more.) On other hand, a poor quality phone line may carry less than a tenth of that—perhaps as few as 4,000 points per second. If you ever noticed how horrible hold music sounds, that's one of the reasons.

Some Social Implications of Digitization

The implications are many; here we barely scratch the surface. A good source of thoughtful discussion on the effects on digitization is the book Blown to Bits by Abelson, Ledeen, and Lewis, available for free at `http://bitsbook.com` or in hardcopy at your library/bookstore.

Universal Devices

Because everything is reduced to bits, you do not need different kinds of storage for different information: a harddrive or a flash memory stick can store anything. Similarly for networks. We had to build the telegraph network for text, the telephone network for voice, and the cable network for video. Now that we have the Internet, we no longer need a separate network for each type of data. Instead, the Internet carries bits, and every type of data is reduced to bits. Of course, the bits themselves may be delivered differently—e.g., by electricity, light, or radiowaves—but the mode of delivery is no longer tied to the content.

However, regulations have not caught up with the universality of devices and information. For instance, anyone can have an 800 number—a phone company cannot refuse to give you because it doesn't like your political cause. Yet Verizon (initially) rejected a request by

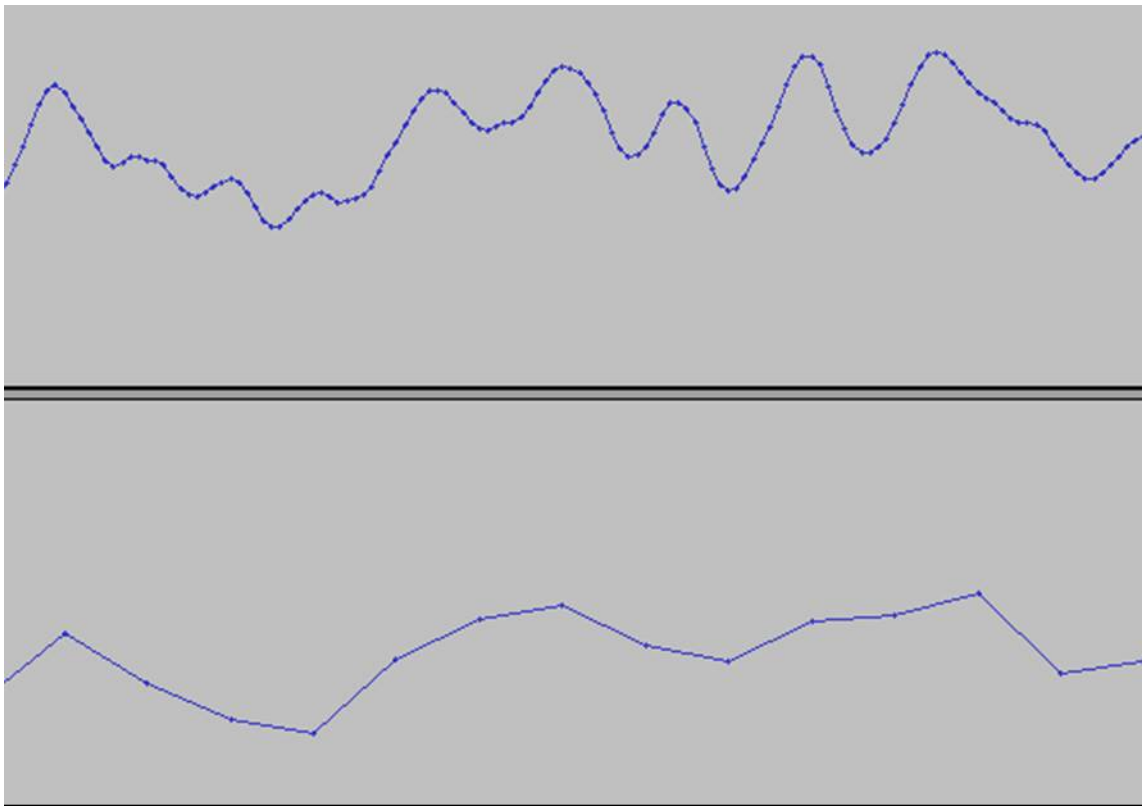


Figure 4: The soundwave resulting from connecting the dots made by digitization (as displayed by the Audacity software package, `http://audacity.sourceforge.net/`). Note that the same piece of music looks very different when the dots are made 44,100 times per second (top) and 4,410 times per second (bottom). The sound quality is also quite different, with the bottom version sounding very flat. This soundwave is for about 1/200th of a second of Vivaldi's Spring (performed by John Harrison and Wichita State University Chamber Players, Robert Turizziani, conductor).

NARAL pro-choice America for a special short text message (SMS) number because they were concerned about the controversy surrounding abortion. Yet text messages are bits the same as voice messages.

There are current debates about so-called "net neutrality"—whether companies that carry bits on the Internet should be allowed to discriminate based on what bits they are and where they are going (for example, giving preferential treatment to movies from certain studios to make them download faster, or giving poor treatment to voice-over-IP phone calls to entice users to sign up for phone service).

Regulations about surveillance (both government and private) are also different for different types of data (it should be observed that, in general, digitization makes vast surveillance easier even as it gives individuals new tools, such as encryption, to protect themselves).

Similarly, regulations about liability differ. It is well accepted that the phone company is not liable for evils committed over the phone wires. No such universal understanding exists for internet service providers, particularly those that are in the business of hosting user-generated content (e.g., YouTube or craigslist).

Public Information is a Lot More Public

The fact that pictures are bits changes our view of the world dramatically. In the US and most other places, anyone can take pictures in public places. But there has been lots of controversy behind Google Street view, for example, despite the fact that it's just taking pictures in public places. The difference is that pictures are ending up on-line rather than in a library or a book. Public information becomes much more public when it is digitized—because instead of traveling to a library (possibly halfway around the world) or searching for a book (possibly out of print) you can get it on-line from anywhere in the world with little effort.

Let's go back to the picture in Figure 2. This picture was collected by Kenneth and Gabrielle Adelman as part of California Coastline Project, documenting the state of the coast, its erosion, and environmental hazards. They simply flew up and down the coast in a helicopter and took pictures of the ground at fixed intervals. In Feb 2003 Barbra Streisand filed a lawsuit to stop this picture from appearing on the Internet, because it shows her estate in Malibu. One of the complaints in the lawsuit was the picture shows everything in minute detail, down to "positioning of her parasols and deck chairs." Of course, whether the picture actually shows such detail depends on the resolution—i.e., on the quality of digitization. Should the quality of digitization acquire legal significance?

If this was some research institute taking analog photos and putting them in a paper archive at some university (even one publicly accessible), there probably would have been no lawsuit. But digitization enables us to easily make a *very* public archive. Our laws generally do not distinguish between different notions of "public": on-line digital or off-line paper-based. Should they?

Note another thing about bits: they are hard to kill by lawsuit. They don't respect jurisdiction and travel freely around the world. Attempts to silence them tend to magnify them, because they can be copied so easily. This photograph became much more widely known as a result of the lawsuit.

Streisand's lawsuit and the attention it drew to the photograph of her estate have given

us a new term: "the Streisand effect." It refers to the fact that once bits are out on-line, attempts to put them away tend to only draw more attention to them. People still don't get it, and examples of this happen all the time. Most recent famous ones include the leaked internal memos of election machine company Diebold (which were rather incriminating) and an attempt by British ISPs to block a Wikipedia page with an album cover that depicted a nude child. Both efforts backfired: both the Diebold memos and the Wikipedia page got much more attention as a result.

It is debatable whether the Streisand effect is an inevitable outcome of digitization. Information tends to find ways around censorship, but new regulations and technical efforts may make it much harder. For instance, Australia, which we consider to be a free-speech-loving Western democracy, is considering a countrywide filtering system.

Copying and Copyright

Copyright is based on providing the creator a limited-term monopoly on making copies. But making copies is what computers do; there's no distinction between a copy and an original when you are talking about bits (in particular, copies of digital information are perfect). To even display a picture or play a song stored on your harddrive, the computer needs to make a copy in memory. When your computer goes to another computer and downloads a song, who is making the copy? If you want to loan your friend a digital book you purchased, how can you do it without making a copy? If you have a song that you purchased to play on one device (e.g., the iPod), should you be able to convert it to play it on another device (e.g., if the iPod breaks or you want to upgrade)?

Society is struggling with these issues without a single overriding principle, but rather through a collection of ad hoc moves, in which the voices heard are often those of the best-organized or most incentivized players.

How Many Bits

Using prefixes from the metric system, a kilobit (kb) is 1,000 bits, and a kilobyte (kB) is 1,000 bytes, or 8,000 bits (note that lower case b usually means bit, and uppercase B usually means byte in such abbreviations). Similarly, a megabyte is a million (10^6) bytes, a gigabyte is a billion (a thousand million or 10^9) bytes, a terabyte is a trillion (a thousand billion, or 10^{12}) bytes, and a petabyte is a quadrillion (a thousand trillion or 10^{15} bytes). However, because in computer science powers of two come much more naturally than powers of ten, and because $2^{10} = 1,024$ is close to $10^3 = 1,000$, people often refer to a kilobyte as $2^{10} = 1,024$ bytes, a megabyte as $2^{20} = 1,048,576$ bytes, a gigabyte as 2^{30} (a little over a billion) bytes, a terabyte as 2^{40} (a little over a trillion) bytes, and a petabyte as 2^{50} (a little over a quadrillion) bytes. It doesn't make a difference in order of magnitude calculations, but can on occasion cause confusion.