

Data Compression 1 – DatComp1

has always been actual for data to computers. 20 years ago because storage was limited and expensive, today because of the tremendous amount of data that we are working with. This amount grows more than exponentially every year !

Bits & ...

To understand the basic about compression we must look at how computers work. And to understand that, we will begin with numbers, as we know them with ciphers from 0 to 9.

10.000	1.000	100	10	1	
*	*	*	*	*	
0	0	0	0	0	
1	1	1	1	1	
2	2	2	2	2	
3	3	3	3	3	
4	4	4	4	4	
5	5	5	5	5	
6	6	6	6	6	
7	7	7	7	7	
8	8	8	8	8	
9	9	9	9	9	

If we want to represent the number 5042, we can do it as shown, utilizing the position of the cipher to give a value. We never write the uppermost line (10.000 1.000, 100, 10, 1) – it is implicit or understood by everyone.

In a computer (due to construction limitations) it is only possible to have 2 ciphers 0 and 1. It is called a bit.

It ain't much, but with this information, we can represent any number, just as we did above. We must use more ciphers to represent a given number, but space is not a problem. F.ex. if we have 5 bits :


16	8	4	2	1	=
*	*	*	*	*	
0	0	0	0	0	
1	1	1	1	1	13

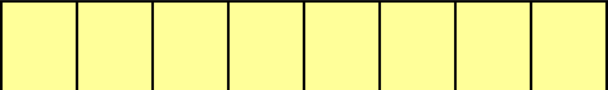
we can represent the number 13 as shown.

Basically there are no difference between this and the 10 cipher system we know. We just need more columns.

... *bytes*

are 8 bits together. That is for the moment, it will change upward with time because of bigger and bigger computer systems.

possibilities
 Representing 0 $\rightarrow 1 = 2^1 - 1$

 Representing 0 $\rightarrow 255 = 2^8 - 1$

1 char = 2 Bytes = 16 bits

i.e. a char variable in Java can represent values from 0 $\rightarrow 65535 = 2^{16} - 1$.

When we consider, that Unicodes usual values (non printed as well as printed ciphers and characters) has values from 0 $\rightarrow 127 = 2^7 - 1$ this Java char representation is overkill and waste of space. It reserves 16 bit, but utilize only 7.

Java's API offer us several classes with methods to act upon bits, bytes, chars and Strings.

KEA Digital - jart@kea.dk asbc@kea.dk

DatComp1 ***EXERCISE :***

- 1) Plan and implement a way to read in Alice In Wonderland (*.txt file from www.gutenberg.org on Fronter) and compress it to a new, smaller *.txt file.
- 2) Implement the opposite too. That is reading the compressed *.txt-file and restore it to the original.
- 3) With 1) & 2) done, reconsider your compressing method. Can it be more efficient ? (hint: usually the answer is YES!)
- 4) Redo 1) and 2) with this better method.

Who has the best compression of Alice in Wonderland in February ?