

The secret of the Aleph

Karl-Ludwig Butte (LKL_Butte@web.de)

Did you ever wonder what's the use of set theory ? Do you remember those long gone schooldays when your mathematics teacher bullied you with subsets, cartesian product or even power sets and the only thing you ached for was the sound of the schoolbell to indicate the end of the lesson ? Or perhaps you wondered *who* in the world did ever come up with that and *why* ?

To answer all these questions, we have to travel to the last quarter of the 19th century to the city of Halle on the river Saale (Fig. 1) in Germany in todays federal land Saxony-Anhalt. Halle is a very old town which is first mentioned in the year 806 AD. Halle had already seen many good and bad times when Georg Cantor, son of a broker, moved there 1869 as associate professor. He was a mathematician through and through and someday – the exact date is not handed down – he came in contact with the questions and problems of infinity.



Fig. 1: Halle (Saale) around 1900

(Source: Wikipedia / public domain)

These problems have been documented between the 5th and 6th century BC and have fascinated, deranged, consumed, yes, even destroyed mathematicians for centuries to come. Cantor learned about these problems the hard way and had to get psychiatric attendance several times. He even died 1918 in a sanatorium in Halle.

The Greek philosopher Zeno of Elea (495 – 435 BC) worked hard on paradox problems without success for a long time which lead him to the idea of infinity. The most famous of these is the footrace between Achill and the turtle. Achill was the fastest runner of his time and still, Zeno argued, he shouldn't be able to overtake the turtle, which forsooth is not one of the fastest animals on earth. Because of its slowness the turtle got a head start. As Achill reached the point from where the turtle started, the turtle had moved on, too. As Achill now reached this second point, the turtle again had moved on. Because this could be carried on endlessly, Zeno came to the conclusion that Achill could never reach the turtle. Under the precondition of infinite divisibility of time and space in fact there shouldn't be any movement possible.

Zeno wasn't the first to agonise over such problems. Approximately a hundred years earlier, Anaximandros (c. 610 – c. 546 BC) introduced the term of infinity. Aristotle (384 – 322 BC) differentiated between potential and actual and even used these terms on sets. A set to which infinite objects could be added in principal he called “potential infinite” whereas a set which already contained infinite objects he called “actual infinite”. The latter he declared impossible. Actual infinity was equated with the existence of God and often was used as proof of God.

Georg Cantor ignored those limitations. For him there was a sequence of infinities with different cardinalities. These cardinalities Cantor called Aleph (\aleph), the first letter of the Hebrew alphabet and sign of God. \aleph_0 was the sign for the lowest cardinality of integer and rational numbers. But Cantor knew that the set of irrational numbers contained more numbers than the set of integers and therefore needed a higher order \aleph . Cantor examined the infinities with different cardinalities and discovered the so called *Ordinal Arithmetic*. To his regret he couldn't elate his colleagues. Merely a few theologians appreciated his work because they were familiar with Jewish ideas and the Kabbalah.



Fig.2: Georg Cantor 1894
(source:Wikipedia/Public domain)

But Cantor didn't let himself be deterred. He had just opened the gate to the wonderworld of Ordinal Arithmetic and was eager to discover their order. During the 1870s Cantor already proved that for every set there was always a set with a higher order: namely the set of all subsets of a set, called power set. E.g. the set of all subsets of the set $\{1, 2, 3\}$ is $\{ \{\}, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\} \}$. Please notice that the number of elements is always a power of 2. In our example it is $2^3 = 8$. This will be important later.

But this was just the beginning. Cantor occupied himself with the examination of the cardinality of the different \aleph and put the mathematics of his time on a completely new footing. The further development caused some furore, especially when several contradictions were discovered by others. But all these further developments are way out of scope of this article. The power set alone is a remarkable achievement and we should bide a little while and notice that it is cumbersome to write down the power set of a given set, except for those sets with only a few elements. Furthermore the question arises, if a pocket calculator could help us in doing so. On condition the calculator model we use can handle set datatypes this question can be answered with “yes” as the following program for the HP-50G demonstrates.

The basic idea for generating all subsets of a given set comes from the observation that the number of subsets is calculated as a power of 2, the cardinality therefore is $2^{\text{number of elements}}$. Therefore one can program a loop from 1 to $2^{\text{number of elements}}$ and the counter variable (I) could be converted into a binary number. The 1's of this binary number can be used to address the elements of the given set, as is shown in Fig. 3.

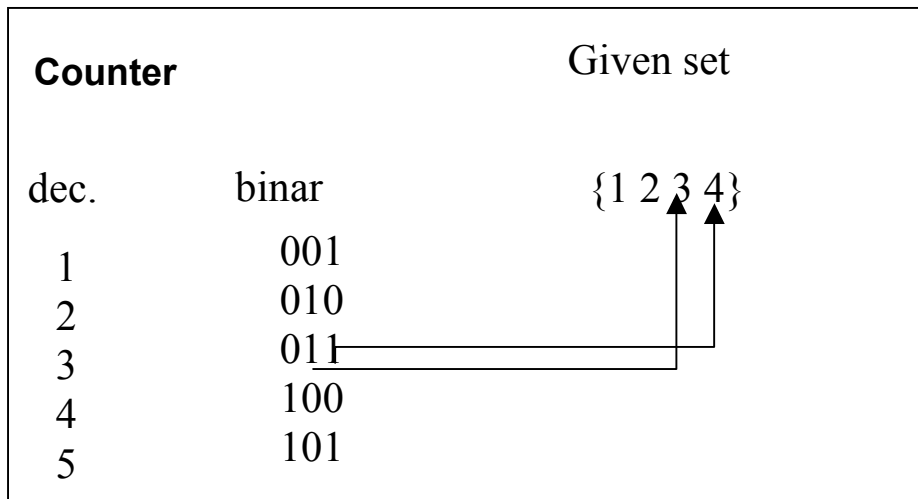


Fig. 3: Addressing the elements of the given set

Of course you cannot use several 1's at once for addressing. Therefore the binary number must be scanned sequentially for 1's. This is done with a second binary variable (X) in a loop in which X is set to values of power of 2 (1 = 0001b, 2 = 0010b, 4 = 0100b, etc.). In each cycle I and X are combined with a logical AND thereby the 0's mask out the non relevant bits (see Fig. 4). This approach makes sure that all subsets are addressed without getting too complicated loop constructions.

Alas, there is the disadvantage that the subsets are not sorted in the order of their cardinality. The SORT command of the HP-50G cannot be used because the SORT command doesn't sort sets as elements of a set and doesn't recognise their cardinality. Therefore a special sort algorithm is implemented which stores the cardinalities of the subset during their generation. After this phase, the cardinalities are sorted with a bubblesort and whenever values have to be swapped, the corresponding subsets in the power set are swapped, too.

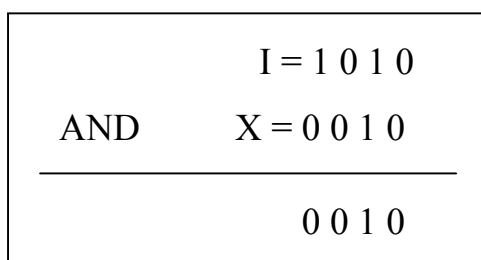


Fig. 4: Masking out non relevant bits.

The program uses the following variables:

variable	datatype	comment
AA	Int	Number of elements of the given set
MA	List	Given set
Y	Int	Number of elements of the power set
MT	Int	Cardinality of the power set
I	Int	Counter variable of the outer loop for generating the subsets (see text above) / Second use: Addressing elements of ML during the bubblesort
N	Int	Counter variable for the inner loop for generating the subsets
X	Int	Variable for scanning for 1's in variable I (see text above)
J	Int	Variable helping with addressing single elements of the given set. / Second use: Addressing single elements of ML during the bubblesort.
G	Int	Number of elements of the subset which is currently worked on.
ML	List	Set of cardinalities of the subsets
H	Int	Number of elements in ML.
ME	Liste	Power set (end result)
Z	Int	Counter variable for the bubblesort.

The program uses the following flags:

Flag	Comment
1	False = end of the outer While-loop of the bubblesort
2	False = all elements have been sorted, flag 1 can be reset
3	True = the result power set will be printed; flag will be reset automatically.

The following table shows the steps of operation for the program:

Step	Operation	Keys	Result
1	If applicable set flag 3 if the result should be printed. Advisable for sets with 4 elements and more.	3 Enter SF	
2	Input the given set		e.g. {1 2 3 4}
3	Start the program	ALPHA ALPHA POTM ALPHA Enter	Result power set will be displayed

The following program listing with comments shows how the algorithm drafted before is implemented:

Step	Instruction	Comment
1	\<< LIST\> \> AA	Dissolve the given set into single elements to get the number of elements.
2	<< AA \>LIST	Restore the given set for further use.
3	0. 0. 0. 0. 0. 0. 0. 1. { 0. }	Initialize variables
4	\> M A N I X Y G M T J H M L	
5	<< { }	Insert the empty set as first element of the power set
6	1. 'Y' STO+	Increment the number of elements of the power set.
7	2. AA ^	Calculate the cardinality of the power set.
8	'MT' STO	
9	1. MT	Initialize the outer For-loop running from 1 to the cardinality of the power set.
10	FOR I	
11	0. AA 1. -	Initialize the inner For-loop running from 0 to number of

		elements of the give set – 1.
12	FOR N	
13	2. N ^	Calculating 2^N for scanning counter variable I (see text above).
14	'X' STO	
15	I R\->B	Convert I to binary
16	X R\->B	Convert X to binary
17	AND	Calculate the logical AND of I and X (see text above).
18	IF X R\->B == THEN	If „yes“
19	N 1. +	Increment N and store in J...
20	'J' STO	
21	'MA(J)' EVAL	...to be used as address of the current element to be added to the result.
22	1.	Increment the number of elements of this subset...
23	'G' STO+	... and store it in G.
24	END	
25	NEXT	
26	G 0.	Test if the subset currently calculated is finished.
27	IF \=/ THEN	If G <> 0 ...
28	G \->LIST	...the elemnts currently on the stack will be packed into a set...
29	1. 'Y' STO+	...and the number of elements of the power set will be incremented.
30	H 1. +	The cardinality of the subset is stored in ML for later sorting.
31	'H' STO	
32	ML LIST\->	
33	DROP	
34	G H \->LIST	

35	'ML' STO	
36	END	
37	0. 'G' STO	Prepare for the next subset.
38	NEXT	
39	Y \->LIST	At last all subsets are packed into one set...
40	\-> ME	...and stored in ME.
41	\<< 1. SF	Sorting the elements of the power set according to their ascending cardinality. (bubblesort).
42	WHILE 1. FS? 1. == REPEAT	Outer While-loop as long as flag 1 is set.
43	1. Y 1. -	Inner For-loop...
44	FOR Z	For all elements of the set of cardinalities.
45	Z 'T' STO	Address of the first element to be compared.
46	Z 1. + 'J' STO	Address of the second element to be compared.
47	ML I GET	Get the elements
48	ML J GET	
49	IF > THEN	If the elements must be swapped
50	ML I GET	then the first swap occurs in ML...
51	ML J GET	
52	SWAP	
53	ML SWAP	
54	J SWAP	
55	PUT	
56	SWAP	
57	I SWAP	
58	PUT	
59	'ML' STO	
60	ME I GET	...and then in the power set

61	ME J GET	
62	SWAP	
63	ME SWAP	
64	J SWAP	
65	PUT	
66	SWAP	
67	I SWAP	
68	PUT	
69	'ME' STO	
70	2. SF	Flag 2 is set to indicate that a swap has been made gesetzt
71	END	
72	NEXT	
73	IF 2. FS?C 0. == THEN	After one cycle without any swap...
74	1. CF	...flag 1 is reset to indicate the...
75	END	...end of the bubblesort.
76	END	
77	IF 3. FS?C 1. == THEN	If flag 3 is set...
78	ME LIST\>	...the power set is printed out
79	DROP	
80	PRST	
81	CLEAR	
82	END	
83	ME	Output of the power set to the display.
84	\>>	
85	\>>	
86	\>>	
87	\>>	

Be sure to switch on the handshaking for printing on the HP-82240 infrared printer. Otherwise it is possible that some data maybe lost.

Readers, who would like the program in electronic form may send a request via email to the author.

Last but not least the following reference list completes this article:

Aczel, Amir D.: „The Mystery of the Aleph, Mathematics, the Kabbalah, and the Search for Infinity“, Four Walls Eight Windows, New York/London © Amir D. Aczel 2000

http://en.wikipedia.org/wiki/Power_set

Malle, H. „Handbuch der Mathematik“, Nikol Verlagsgesellschaft Hamburg, 2006