Verhoeff algorithm

From Wikipedia, the free encyclopedia

The **Verhoeff algorithm**[1] is a checksum formula for error detection developed by the Dutch mathematician Jacobus Verhoeff and was first published in 1969.[2][3] It was the first decimal check digit algorithm which detects all single-digit errors, and all transposition errors involving two adjacent digits,[4] which was at the time thought impossible with such a code.

Goals [edit]

Verhoeff had the goal of finding a decimal code—one where the check digit is a single decimal digit—which detected all single-digit errors and all transpositions of adjacent digits. At the time, supposed proofs of the nonexistence[5] of these codes made base-11 codes popular, for example in the ISBN check digit.

His goals were also practical, and he based the evaluation of different codes on live data from the Dutch postal system, using a weighted points system for different kinds of error. The analysis broke the errors down into a number of categories: first, by how many digits are in error; for those with two digits in error, there are *transpositions* (*ab* → *ba*), *twins* (*aa* → 'bb'), *jump transpositions* (*abc* → *cba*), *phonetic* (*1a* → *a0*), and *jump twins* (*aba* → *cbc*). Additionally there are omitted and added digits. Although the frequencies of some of these kinds of errors might be small, some codes might be immune to them in addition to the primary goals of detecting all singles and transpositions.

The phonetic errors in particular showed linguistic effects, because in Dutch, numbers are typically read in pairs; and also while 50 sounds similar to 15 in Dutch, 80 doesn't sound like 18.

Taking six-digit numbers as an example, Verhoeff reported the following classification of the errors:.

Digits in error	Classification	Count	Frequency
1	Transcription	9,574	79.05%
2	Transpositions	1,237	10.21%
	Twins	67	0.55%
	Phonetic	59	0.49%
	Other adjacent	232	1.92%
	Jump transpositions	99	0.82%
	Jump Twins	35	0.29%
	Other jump errors	43	0.36%
	Other	98	0.81%
3		169	1.40%
4		118	0.97%
5		219	1.81%
6		162	1.34%
Total		**12,112**	

Description [edit]

Verhoeff devised his algorithm using the properties of the dihedral group of order 10 (a non-commutative

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Русский
Edit links

system of operations on ten elements, which corresponds to the rotation and reflection of a regular pentagon), combined with a permutation. He claimed that it was the first practical use of the dihedral group, and confirmed the principle that in the end, all beautiful mathematics will find a use,[6] even though in practice the algorithm will be implemented using simple lookup tables without needing to understand how to generate those tables from the underlying group and permutation theory.

This is more properly considered a family of algorithms, as there are other permutations possible, and discussed in Verhoeff's treatment. He notes that this particular permutation, $\begin{pmatrix}0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9\\ 1 & 5 & 7 & 6 & 2 & 8 & 3 & 0 & 9 & 4\end{pmatrix} = (1\ \ 5\ \ 8\ \ 9\ \ 4\ \ 2\ \ 7\ \ 0)(3\ \ 6)$, is special as it has the property of detecting 95.3% of the phonetic errors.[7]

The strengths of the algorithm are that it detects all transliteration and transposition errors, and additionally most twin, twin jump, jump transposition and phonetic errors.

The main weakness of the Verhoeff algorithm is its complexity, and the calculations required cannot readily be performed by hand. A similar code is the Damm algorithm, which has similar qualities.

Table-based algorithm [edit]

The Verhoeff algorithm can be implemented using three tables: a multiplication table *d*, an inverse table *inv*, and a permutation table *p*.

d(j,k)[8]		**k**									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
j	**0**	0	1	2	3	4	5	6	7	8	9
	1	1	2	3	4	0	6	7	8	9	5
	2	2	3	4	0	1	7	8	9	5	6
	3	3	4	0	1	2	8	9	5	6	7
	4	4	0	1	2	3	9	5	6	7	8
	5	5	9	8	7	6	0	4	3	2	1
	6	6	5	9	8	7	1	0	4	3	2
	7	7	6	5	9	8	2	1	0	4	3
	8	8	7	6	5	9	3	2	1	0	4
	9	9	8	7	6	5	4	3	2	1	0

		inv(j)
j	**0**	0
	1	4
	2	3
	3	2
	4	1
	5	5
	6	6
	7	7
	8	8
	9	9

p(pos,num)		**num**									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
pos (mod 8)	**0**	0	1	2	3	4	5	6	7	8	9
	1	1	5	7	6	2	8	3	0	9	4
	2	5	8	0	3	7	9	6	1	4	2
	3	8	9	1	6	0	4	3	5	2	7
	4	9	4	5	3	1	2	6	8	7	0
	5	4	2	8	6	5	7	3	9	0	1
	6	2	7	9	3	8	0	6	4	1	5
	7	7	0	4	6	9	1	3	2	5	8

The first table, ***d***, is based on multiplication in the dihedral group D_5.[9] and is simply the Cayley table of the group. Note that this group is not commutative, that is, for some values of *j* and *k*, $d(j,k) \neq d(k, j)$.

The inverse table ***inv*** represents the multiplicative inverse of a digit, that is, the value that satisfies $d(j, inv(j)) = 0$.

The permutation table ***p*** applies a permutation to each digit based on its position in the number. This is actually a single permutation (1 5 8 9 4 2 7 0)(3 6) applied iteratively; i.e. $p(i+j,n) = p(i, p(j,n))$. (See [*clarification needed*]

The Verhoeff checksum calculation is performed as follows:

1. Create an array *n* out of the individual digits of the number, taken from right to left (rightmost digit is n_0, etc.).
2. Initialize the checksum *c* to zero.
3. For each index *i* of the array *n,* starting at zero, replace *c* with $d(c, p(i \bmod 8, n_i))$.

The original number is valid if and only if $c = 0$.

To generate a check digit, append a *0*, perform the calculation: the correct check digit is $inv(c)$.

Examples [edit]

Generate a check digit for *236*:

i	n_i	$p(i,n_i)$	*c*
0	0	0	0
1	6	3	3

Validate the check digit *2363*:

i	n_i	$p(i,n_i)$	*c*
0	3	3	3
1	6	3	1

2	3	3	1
3	2	1	2

c is *2*, so the check digit is *inv(2)*, which is *3*.

2	3	3	4
3	2	1	0

c is zero, so the check is correct.

References [edit]

1. ^ Verhoeff, J. (1969). *Error Detecting Decimal Codes (Tract 29)*. The Mathematical Centre, Amsterdam. doi:10.1002/zamm.19710510323.
2. ^ Kirtland, Joseph (2001). *Identification Numbers and Check Digit Schemes*. Mathematical Association of America. p. 153. ISBN 0-88385-720-0. Retrieved August 26, 2011.
3. ^ Salomon, David (2005). *Coding for Data and Computer Communications*. Springer. p. 56. ISBN 0-387-21245-0. Retrieved August 26, 2011.
4. ^ Haunsperger, Deanna; Kennedy, Stephen, eds. (2006). *The Edge of the Universe: Celebrating Ten Years of Math Horizons*. Mathematical Association of America. p. 38. ISBN 978-0-88385-555-3. LCCN 2005937266. Retrieved August 26, 2011.
5. ^ Sisson, Roger L., An improved decimal redundancy check, Communications of the ACM Vol. 1, Iss. 5, May 1958, pp10-12, DOI: 10.1145/368819.368854.
6. ^ Verhoeff, J. (1975). *Error Detecting Decimal Codes (Tract 29), second printing*. The Mathematical Centre, Amsterdam.
7. ^ Verhoeff 1969, p. 95
8. ^ Verhoeff 1969, p. 83
9. ^ Gallian, Joseph A. (2010). *Contemporary Abstract Algebra* (7th ed.). Brooks/Cole. p. 111. ISBN 978-0-547-16509-7. LCCN 2008940386. Retrieved August 26, 2011.

External links [edit]

- Detailed description of the Verhoeff algorithm
- A description using lookup tables
- Verhoeff implementation in Perl (from CPAN)
- Verhoeff implementation in FileMaker Pro
- Verhoeff implementation in MS SQL Server Transact SQL
- Biographical sketch of Jacobus Verhoeff
- Verhoeff validation and generation code in C++
- Verhoeff validation & generation code in Javascript
- Verhoeff validation & generation code in C#, VB.NET, VBA, Java, Python, D, PHP, ActionScript and Pascal/Delphi

Wikibooks has a book on the topic of: ***Algorithm_Implementation/Checksums/Verhoeff_Algorithm***

Categories: Modular arithmetic | Checksum algorithms | Error detection and correction