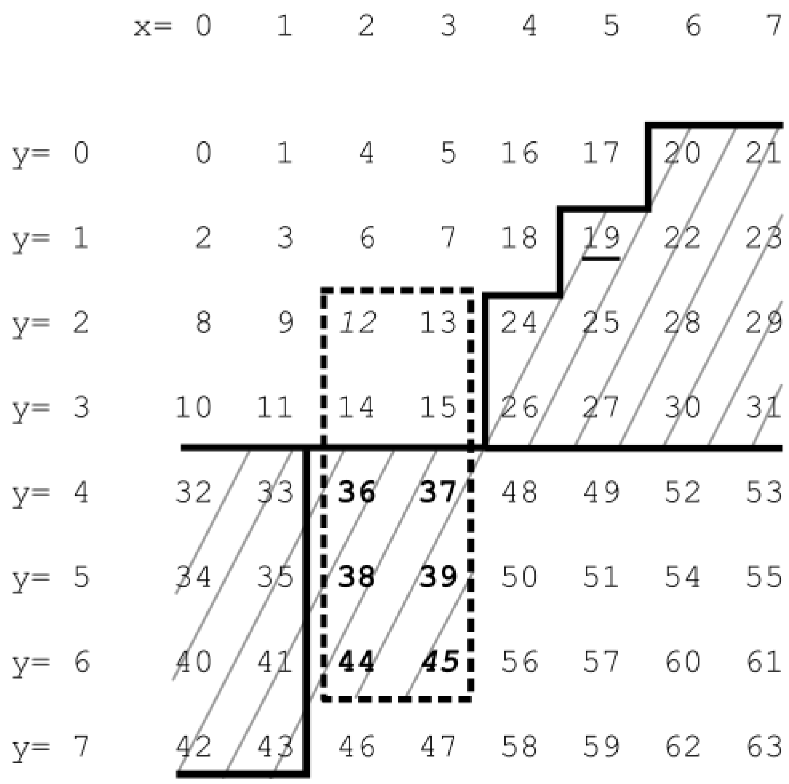
The following Details are found in englisch Wikipedia „Z-order curve“ content “Use with one-dimensional data structures for range searching”
or in German wikipedia „z-kurve“, content „Anwendungen in der Informatik“

Use with one-dimensional data structures for range searching

By bit interleaving, the database records are converted to a (possibly very long) sequence of bits. The bit sequences are interpreted as binary numbers and the data are sorted or indexed by the binary values, using any one dimensional data structure, as mentioned in the introduction. However, when querying a multidimensional search range in these data, using binary search is not really efficient. Although Z-order is preserving locality well, for efficient range searches an algorithm is necessary for calculating, from a point encountered in the data structure, the next possible Z-value which is in the multidimensional search range:


x= 0 1 2 3 4 5 6 7
y= 0 0 1 4 5 16 17 20 21
y= 1 2 3 6 7 18 19 22 23
y= 2 8 9 12 13 24 25 28 29
y= 3 10 11 14 15 26 27 30 31
y= 4 32 33 36 37 48 49 52 53
y= 5 34 35 38 39 50 51 54 55
y= 6 40 41 44 45 56 57 60 61
y= 7 42 43 46 47 58 59 62 63

In this example, the range being queried (*x* = 2, ..., 3, *y* = 2, ..., 6) is indicated by the dotted rectangle. Its highest Z-value (MAX) is 45. In this example, the value *F* = 19 is encountered when searching a data structure in increasing Z-value direction, so we would have to search in the interval between *F* and MAX (hatched area). To speed up the search, one would calculate the next Z-value which is in the search range, called BIGMIN (36 in the example) and only search in the interval between BIGMIN and MAX (bold values), thus skipping most of the hatched area. Searching in decreasing direction is analogous with LITMAX which is the highest Z-value in the query range lower than *F* (15 in the example).. The BIGMIN problem has first been stated and its solution shown in Tropf and Herzog (**https://hermanntropf.de/media/multidimensionalrangequery.pdf**). Details are found in chapter 4, with hints to „Range search" and „computing Litmax" and the tables "Litmax decision table" and "Bigmin Decision Table".

An extensive explanation of the LITMAX/BIGMIN calculation algorithm, together with Pascal Source Code (3D, easy to adapt to nD) and hints on how to handle floating point data and possibly negative data, is provided 2021 by Tropf (**https://hermanntropf.de/media/DBCode_mit_Erlaeuterung.txt**).