Indian Journal of Science and Technology, *Vol 7(9), 1387–1390, September 2014*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Calculations of Mapping from Two Dimensional Plane to Integer Line and the Reverse using Hilbert Curve

S. Santhosh Baboo[1] and V. Narmadha[2*]

[1]Department of Computer Science, D.G. Vaishnav College, Chennai, Tamil Nadu, India
[2]Department of Computer Science, Women's Christian College, Chennai, Tamil Nadu, India; vnarmadha@yahoo.co.in

Abstract

Location management systems require multi-dimensional access methods to allow efficient handling of spatial queries. Because there is no total ordering of locations that preserves the spatial locality between objects, it is difficult to design multi-dimensional access method in the way as traditional one-dimensional access methods. However, mapping multi-dimensional data into a single dimension makes it possible to answer these queries in an efficient way. Space filling curves are used to connect all the points on a plane. Hilbert curve preserves locality than any other space filling curves. Hilbert curve is used for sequencing the points in a plane and use the order for storage. This paper discusses the efficient way of calculating the mapping from two dimensional planes to integer line or real line and the vice versa. The time complexity for encoding and decoding using the proposed algorithm is less compared with existing algorithm. The memory requirement is almost constant compared to state machine approach. This algorithm can be used for storing spatial data in efficient way so that the access time becomes minimal.

Keywords: Efficient Storage Management, Hilbert Curve, Indexing, Spatial Data Structure

1. Introduction

A space filling curve is a continuous, onto mapping from R to R^d. It was not always clear that such a mapping would exist for d > 1. In 1878 Cantor exhibited a one to one map from the unit interval I = [0, 1] onto unit square S = [0, 1] X [0, 1] and thus proving that I and S have same cardinality. Later in the late 19th century Peano showed that it is possible to find a continuous map which is not one to one but onto for d = 2 and d = 3. An example given later by Lebesgue makes use of the standard Cantor set I. Any number t in has a expansion given by

$t = \sum_{i=0}^{\infty} ai/3i$, where each ai takes one of the values 0, 1, or 2.

The Cantor set C is defined as

$C = \{t : t = \sum_{i=0}^{\infty} ai/3i$ with ai = 0 or 2$\}$, C is known as fractal.

Another (geometric) way to arrive at *C* is the following: from *I*, first remove the open middle one-third interval G_I = (1/3, 2/3) and call what remains as F_1. Thus F_1= [0, 1/3] U [2/3, 1]. From each of the two intervals in F_1, remove their open middle one-third intervals (1/9, 2/9) and (7/9, 8/9) and call what is left as F_2. From each of the four intervals in F_2, remove their open middle one third intervals and call what remains as F_3 and so on. What finally remains is the Cantor set C = n *Fn*. Lebesgue's construction can now be given as follows:

for t ∈ C with $t = \sum_{i=0}^{\infty} ai/3i$,

*Author for correspondence

Define

$$x(t) = \sum_{i=1}^{*} \frac{b2i - 1}{2i} \text{ and } y(t) = \sum_{i=1}^{*} \frac{b2i}{2i} \text{ where } b_i = a_i / 2.$$

The popularity of space filling curve is due to the geometric construction given by the Germen mathematician David Hilbert. His basic idea was that if the unit interval should fill the whole of S, then $1/4^{th}$ of I will fill a corresponding sub square of S of area ¼ with continuity in neighbouring squares. Next I and S can be replaced by an interval of length ¼ and sub square area of ¼ respectively and the process can be repeated. Hence for each n > 1, I and S are subdivided into 4^n closed intervals and 4^n closed sub squares. The first three stages are shown in Figure 1. At each stage, centers of the sub square are joined by consecutive straight lines in the shown in Figure 1. This procedure defines a sequence of continuous functions from I to S. Since the length of the sides of the square tends to 0, the sequence converges to a limit function which is therefore continuous. This limit function is called Hilbert Curve.

Since then, quite a number of space filling curves have appeared in the literature. During the early days space filling curves were primarily seen as a mathematical curiosity. Today however, space filling curves are applied in areas as diverse as load balancing for grid computing, colour space dimension reduction, small antenna design, I/O-efficient computations on massive matrices, and the creation of spatial data indexes. In this paper, we focus on the application of space filling curves to the creation of query-efficient spatial data indexes using Hilbert curve.

2.1 Hilbert Curve

Among the space filling curves as Hilbert curve has the property of preserving the locality. A Hilbert curve[2] is a particular form of the space filling curve that traverses a $2^N \times 2^N$ array of points while never maintaining the same direction for more than three consecutive points, where N is the resolution, level or depth. Let us see how in 2-D Hilbert curve drawn.

A square is initially divided into 4 sub-squares which are then ordered such that any pair of consecutive sub squares shares a common edge. The ordering is illustrated by drawing a line through their centre-points and this line is called a first-order curve. Figure 1(b) shows the next step in which each sub-square is then divided into 4 sub-squares. The sub squares within the first and last squares of the first step are ordered differently to ensure the adjacency property is always preserved.

The three diagrams in Figure 1(a, b, c) illustrate the way in which the Nth Hilbert curve H_N is obtained from H_{N-1}. The curve H_{N-1} is replicated and moved into the four quadrants of a larger square after a suitable rotation and these four curves are joined by three line segments.

To describe vertex labeling algorithm conveniently, the four quadrants are numbered as follows: we define the lower-left quadrant as quadrant 0, the upper-left quadrant as quadrant 1, the upper-right quadrant as quadrant 2 and the lower-right quadrant as quadrant 3; see Figures 1a. Consequently, quadrants 1 and 2 of H_N are the copies of H_{N-1}, quadrant 0 is a copy of H_{N-1} rotated by 90° clockwise and quadrant 3 is a copy of H_{N-1} rotated by 90° counter-clockwise.

By converting the decimal digits in Figure 1 to their quaternary digits as shown in Figure 2, we obtain the following replication rules: for quadrants 0, 1, 2 and 3, the highest digit of each order is always 0, 1, 2 and 3, respectively, for all resolutions; quadrant 0 of H_N is a copy of H_{N-1} reflected on the minor diagonal, quadrant 1 is a copy of H_{N-1} with each element increased by 4^{N-1}, quadrant 2 is a copy of H_{N-1} with each element increased by $2 \times 4^{N-1}$ and quadrant 3 is a copy of H_{N-1} reflected on the major diagonal with each element increased by $3 \times 4^{N-1}$.

In this paper, we implement a new iterative algorithm for encoding and decoding the Hilbert order based on a replication process of the Hilbert matrix proposed in[8] and make a comparison with other encoding and decoding procedure.

2.2 Hilbert Code Encoding and Decoding

Given the coordinates of a particular point P with pair (X, Y) in a plane, the corresponding Hilbert order H is to be determined. This procedure is called *encoding*. Conversely, given H, the corresponding coordinate (X, Y) is to be determined. This procedure is called *decoding*.

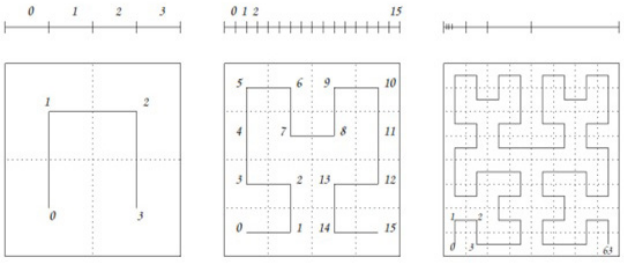

Figure 1. Hilbert curve in 2 dimensions.

Butz uses an iterative algorithm[1] to compute a mapping function with byte-oriented technique such as exclusive OR, shifting etc. Sagan presents an arithmetic method for the generation of the nodes and produces an approximating polygon to represent the Hilbert space filling curve[4]. Hilbert space-filling curves can be explained with the Lindenmayer system which can be used to generate self-similar fractals[5]. Bartholdi presents an algorithm for computing all addresses of scanning path by recursive procedure[6]. A non-recursive algorithm was reported[3] for N dimensional Hilbert space-filling curve using look-up tables. Using tensor product formulation present, Lin and Chen[7] designed both recursive and iterative coding algorithms which scan all space points of two-dimensional and three dimensional Some application problems, such as finding nearest neighbor points and retrieving partial of satellite picture in geographic information system, are not required to scan all data elements of a Hilbert space-filling curve. Now we will introduce a simple mathematical method to construct the encoding and decoding procedures based on the replication of the Hilbert matrix given in [8].

2.3 Modified Algorithm for Hilbert Space-Filling Curve

We use the Cartesian coordinate system to express the positions of all the elements in the Hilbert matrix. Element 0 in the Hilbert matrix is the origin of the Cartesian coordinate system; the direction from left to right is the positive direction of the X -axis; the direction upwards is the positive direction of the Y -axis. Given the coordinates of a particular point P with pair (x, y) in D, the corresponding Hilbert order z is to be determined. This procedure is called encoding. For example, element 8 of resolution 2 in Figure 2 corresponds to the coordinate pair (2, 2). Conversely, given z, the corresponding coordinate (x, y) is to be determined. This procedure is called decoding.

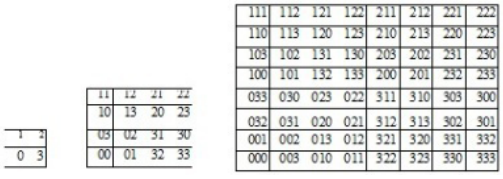

Figure 2. The traditional Hilbert orders of resolutions 1, 2 and 3.

2.3.1 Encoding Algorithm

Encode (input: (x, y) Cartesian coordinate of the point, n: Resolution of Hilbert curve output: H: H-code, (x, y) Cartesian coordinate of the point)

1. Input the location(x, y) and maximum order of the Hilbert curve.
2. Calculate $R_{min} = \log_2(\max(x, y))$ and set width = $2^{Rmin - 1}$
3. If parities of n and R_{min} are not the same, exchange the value of x and y
4. Determine in which quadrant q of the Hilbert code the point lies with R_{min} as resolution
5. Concatenate q to Z
6. Reduce the width by its half.
7. Reduce R_{min} by 1
8. Find new (x, y) as follows depending on the quad
9. If (x, y) lies in the 0^{th} quadrant, interchange x and y
10. If it lies in the first quadrant, reduce y by the width
11. If it lies in the second quadrant, reduce both x & y by the width of the current resolution.
12. If it lies in the third quadrant, calculate new coordinate using the following x = width − y − 1; y = 2*width − x − 1
13. Reduce R_{min} by 1; width has to be reduced to its half value.
14. Repeat steps 4 through 13 till $R_{min} = 0$

2.3.2 Decoding Algorithm

It is the reverse of what is done in the encoding algorithm.

Decode(input: H: H-code, n : Resolution of Hilbert curve output: (x,y) Cartesian coordinate of the point)

1. Covert H to quaternary digits.
2. Extract the least significant digit from H and call it r
3. Set (x, y) to the quadrant position in hilbert curve order 1. (I.e) Set (x, y) = (0, 0) if r = 0. Set it to (0, 1) if r = 1. Set it to (1, 1), if r = 2. Otherwise set (x, y) to (1, 0)
4. Extract the least significant digit from H and call it r
5. Set (x,y) depending on r as follows
   r = 0 : interchange x and y
   r = 1: y = y + width
   r = 2 : x = x + width, y = y + width
   r = 3: x = 2*width − y − 1, y = width = x − 1
6. width = 2*width
7. If parities of n and R_{min} are not the same, exchange the value of x and y .

3. Results and Discussion

The above algorithm is implemented in C language. The program is tested for different order of Hilbert curve. Along with this, Lawder's Algorithm[9] & Butz's Algorithm are also tested. For these two Algorithms we used the code available by Lawder. It has been recorded for point (4, 8) in different depth varying from 1 to 30. The result is shown in Figure 3.

Though the complexity is n when the point lies at nth row/column, it uses less memory and simple calculations. Butz's algorithm creates table for each order of Hilbert curve and the table is searched. So time taken is bounded by the order of the Hibert curve.

Lawder uses state diagram for each order. So the time taken for coding is more compared to the Butz's even though it uses less amount of memory. As order increases the time taken to code and decode also increases.

Our modified algorithm is bounded by calculations and memory as the order increases. So time taken to execute our algorithm is bounded by position of point in the two dimensional plane. It does not create any table or states which involves lot memory as the dimension increases. The algorithm works for Hilbert curve order 32.

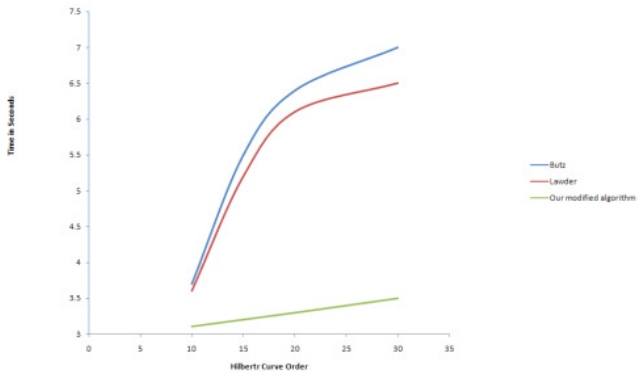

Figure 3. Time taken of encoding and decoding Hilbert code for different order.

4. Conclusion

The algorithm which is discussed in this paper is efficient in terms of memory and execution time. The mapping leads to efficient way of answering the queries like nearest facility, facilities in the in route and efficient planning of trip in a transportation network. It may also help in planning for disaster management during floods or cyclone.

5. References

1. Butz AR. Alternative algorithm for Hilbert's space-filling curve. IEEE Trans Comput. 1971; 20:424–6.
2. Cole AJ. A note on space filling curves. Software Pract Ex. 1983; 13:1181–9.
3. Kamata S, Pacrez A, Kawaguchi E. A method of computing hilbert curves in two and three dimensional spaces. Trans Inst Electron Inform Comm Eng. 1991; J74-D-II(9):1217–26.
4. Sagan H. On the geometrization of the Peano curve and the arithmetization of the Hilbert curve. Int J Math Educ Sci Tech. 1992; 23(3):403–11.
5. Ohno Y, Ohyama K. A catalog of symmetric self-similar space-filling curves. J Recreational Math. 1991; 23:161–73.
6. Bartholdi JJ III, Goldsman P. Vertex-labeling algorithms for the Hilbert space filling curve. Software Pract Ex. 2001; 31(5):395–408.
7. Lin SY, Chen CS, Liu L, Huang CH. Tensor product formulation for Hilbert space-filling curves. Proceedings of the International Conference on Parallel Processing (ICPP 2003); 2003 Oct 6–9; Kaohsiung, Taiwan. Los Alamitos, CA: IEEE Computer Society Press; 2003, 99–106.
8. Chen N, Wang N, Shi B. A new algorithm for encoding and decoding the Hilbert order. Software Pract Ex. 2007; 37:897–908.
9. Lawder JK. Using State Diagrams for Hilbert Curve Mappings. Technical Report. Birkbeck College, University of London: 2000. Reprt No.: JL2/00