ELSEVIER

Information Processing Letters 62 (1997) 17–22

Information Processing Letters

Analysis of the Hilbert curve for representing two-dimensional space

H.V. Jagadish

AT&T Research, 600 Mountain Avenue, Murray Hill, NJ 07974, USA

Received 26 March 1996; revised 10 January 1997
Communicated by G.R. Andrews

Abstract

We give closed-form expressions for the average number of runs to cover an arbitrary square region in two dimensions, using a Hilbert curve. The practical use of the derived formula is that it allows the estimation of the quality of the linearization obtained by using the Hilbert curve to map from a two-dimensional space to a one-dimensional space. Hilbert curves are used extensively as a basis for multi-dimensional indexing structures, and for declustering multi-dimensional data.

Keywords: Hilbert curves

1. Introduction

Consider a *two-dimensional image* represented as a $2^k \times 2^k$ array of 1×1 squares. Each such square is called a *pixel*.

Cover this space with a "Hilbert curve", a recursive (fractal) structure described below in Section 3.1. Using this curve, each point in the given two-dimensional space is mapped to a linear coordinate, the position on this curve.

Consider an $m \times m$ square arbitrarily positioned in this space (with no "wrap-around"). As we follow the Hilbert curve covering of the space it will enter and leave the selected square region one or more times. A *run* is a set of contiguous points on the Hilbert curve that are all in the specified region.

The question we address in this paper is how many "runs" there are in an $m \times m$ square region in a $2^k \times 2^k$ space. The answer depends upon the position of the region in the space. We seek the average over all possible positions of the region in the space. This average is a function of both m and k.

2. Motivation

There is often a need to map points from a multi-dimensional space into a single dimension. In the database context, there are two significant applications in this regard. The first is multi-attribute indexing. Elements from a multi-dimensional attribute space are hashed (or otherwise indexed) onto a linear range of block addresses on disk. Since selections may often be specified only on some of the attributes, or include ranges of attributes, it is desirable that points close together in multi-dimensional attribute space also be close together in the one-dimensional space. For example, with multi-at-

0020-0190/97/$17.00
PII S0020-0190(97)00014-8

tribute hashing [7], record signatures are mapped to buckets, which are stored on disk. We would prefer to access consecutive rather than randomly scattered buckets in response to a relational query. Similarly, with a grid file [4], there is a mapping from grid squares to disk blocks, and we would like to perform this mapping to minimize the number of disk blocks accessed, and would prefer that these blocks be sequential if possible.

A second application arises due to a multi-dimensional indexing technique proposed by Orenstein [5]. The idea is to develop a single numeric index (on a one-dimensional number line) for each point in multi-dimensional space, such that for any given object, the range of indices, from the smallest index to the largest, includes few points not in the object itself. Other applications include bandwidth reduction of digitally sampled signals [1] and graphics display generation [6].

There are many different ways in which to sequence points from a two-dimensional space. For instance, one could perform a simple "scan" – this is how two-dimensional matrices are assigned to memory locations in most programming languages. It is typically desirable, in the course of this mapping from multiple dimensions to one dimension, that "locality" be preserved as much as possible. Considering nearest neighbors along the grid axes only, we can immediately see that each grid point has 4 nearest neighbors in two-dimensional space, but only 2 nearest neighbors in the one-dimensional mapping. Therefore, the best that one can hope to do in a mapping from two dimensions to one dimension is to have two of the four nearest neighbors in the two-dimensional grid to be nearest neighbors in the linear mapping.

In terms of performance, there are many different ways in which one can measure how good a particular mapping is. An important property, for most applications, is how well a "compact" region in the two-dimensional space is represented in the sequential mapping. The number of runs in the sequential space is an important way to measure this. Experimental studies have shown [2,3] that the Hilbert curve performs the "best".

This paper provides analytical formulae for the number of runs required to cover a square region in a two-dimensional grid using a Hilbert curve.

3. Background

3.1. Construction of the Hilbert curve

The Hilbert curve is constructed as follows. Begin with Fig. 1(a) as the level 1 curve. Replicate this in each quadrant of the next level. When replicating, the lower left quadrant is rotated clockwise 90°, the lower right quadrant is rotated anti-clockwise 90°, and the *sense* (or direction of traversal) of both lower quadrants is reversed. The two upper quadrants have no rotation and no change of sense. Thus we obtain Fig. 1(b). Remembering that all rotation and sense computations are relative to previously obtained rotation and sense in a particular quadrant, a repetition of this step gives rise to Fig. 1(c). Further repetition gives Fig. 1(d).

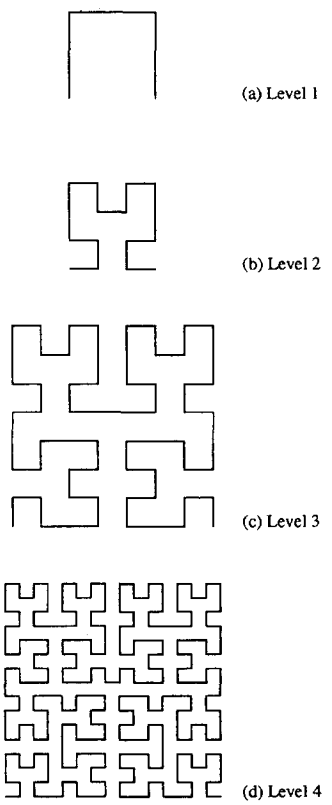

Fig. 1.

3.2. Notation

From the above construction, observe that there is a unique definition of top, bottom, and side for a Hilbert curve construction of any size. We define:

T_k = The top boundary of a Hilbert covering for a 2^k region.

B_k = The bottom boundary of a Hilbert covering for a 2^k region.

S_k = The side boundary of a Hilbert covering for a 2^k region.

These are not numerical measures, but rather just notation so we can refer to the physical boundaries. The top and bottom boundaries are different from each other, but are self-symmetric. The two side boundaries are identical, but each is asymmetric in that one end meets a bottom and the other meets the top.

As we recursively build up the Hilbert covering for a large region, we can observe the following recursion at the boundaries:

$$T_{k+1} = T_k . \circ . T_k$$

$$B_{k+1} = S_k . \circ . S'_k$$

$$S_{k+1} = B_k . \cdot . S_k$$

where $\circ$ represents that there is no connection between the two adjacent halves of the boundary, and $\cdot$ represents that there is a connection between the two halves of the boundary along the Hilbert curve. We have written S'_k to indicate a side boundary with orientation opposite that of S_k.

4. Enumerating possible positions

Consider an $m \times m$ region in a grid of size $2^k \times 2^k$. Along the left-right axis, there are $2^k - m + 1$ positions for the square region, starting from the position where the left end of the region is at the left space boundary, and ending with the position where the right end of the region is at the right space boundary. Similarly, along the top-bottom axis, there are $2^k - m + 1$ positions. The total number of positions to be considered is the product of these two. For $2^k \gg m$, we can approximate this by 2^{2k}.

5. The major recursion

We derive by induction the formula for the total number of runs using an $m \times m$ square region in

Table 1
Table of symbols used

k	Size of grid is $2^k \times 2^k$
m	Region of interest is $m \times m$
T^k	The top boundary of a Hilbert covering for a 2^k region.
B_k	The bottom boundary of a Hilbert covering for a 2^k region.
S_k	The side boundary of a Hilbert covering for a 2^k region.
$R_k(m)$	Total number of runs in all positions of an $m \times m$ region in a $2^k \times 2^k$ grid.
$tt_k(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" two T_k boundaries facing each other.
$ss_k(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" two facing S_k boundaries.
$sb_k(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" an S_k boundary facing a B_k boundary.
$e_k(m)$	$ss_k(m) + 2 * sb_k(m)$.
$cc_k(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" more than two 2^k region boundaries.
$adj_k(m)$	Savings in $R_{k-1}(m)$ when computed through a recursion on account of the edges connecting neighboring quadrants in the Hilbert curve.
$ctt(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" four T_k boundaries when two T_{k+1} boundaries face each other. The result is independent of k.
$css(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" two S_k and two B_k boundaries when two S_{k+1} boundaries face each other. Independent of k.
$csb(m)$	Total number of runs in all positions of an $m \times m$ region that "straddle" three S_k and one B_k boundaries when an S_{k+1} and a B_{k+1} boundary face each other. Independent of k.
$ce(m)$	$css(m) + 2 * csb(m)$.

every possible position of this region in a space of size $2^k \times 2^k$. (The average number of runs is simply obtained by dividing this number by the number of possible positions for the region.)

The base case for the induction is the smallest value of k – we call this k_0 – such that $2^{k_0} \geqslant m$. In this region, we manually count the number of runs in each position.

Let $R_k(m)$ be the total number of runs in all positions of an $m \times m$ square region in a grid space of size 2^k. Then, $R_{k+1}(m)$ is obtained as the summation of the following:

(1) $4 * R_k(m)$, for the four occurrences of the 2^k grid in the 2^{k+1} grid.
(2) $tt_k(m)$, accounting for all positions of the region straddling a boundary of two "tops" of 2^k size grids facing each other. tt_k is short-hand for "cost of (T_k facing T_k)".
(3) $ss_k(m)$, accounting for all positions of the region straddling a boundary of two "sides" of 2^k size grids next to each other.
(4) $2 * sb_k(m)$, accounting for all positions of the region straddling a boundary of a "side" and a "bottom" of a 2^k size grid facing each other.
(5) An additional charge to account for all positions of the region straddling more than one 2^k boundary (these positions are at the center of the 2^{k+1} grid). Call this charge $cc_k(m)$. $cc_k(m) = (m+1) \cdot (m-1)^2$ for all $k > k_0$. (Because there are $(m-1)^2$ positions of such region straddling, with an average of $m+1$ runs per position.)
(6) Minus a saving due to the connectivity between adjacent components at three places (middle of the left side boundary, top boundary, and right side boundary). At each place the savings is $m-1$. Call the total savings here $adj_k(m) = 3(m-1)$.

The specific values of the individual terms is of course, a function of m. However, it is easy to see that the first term grows by a factor of 4 each time k grows by one, the next three terms grow approximately by a factor of 2, and the last two terms change little.

Asymptotically, this means that the total number of runs in a grid space of $2^k \times 2^k$, over all positions of a square region of size $m \times m$, is asymptotically proportional to 2^{2k}, whatever be the value of m. Dividing by the number of possible different positions for the square region, we get the average number of runs to be independent of k.

Writing out the above words algebraically, we have:

$$\begin{aligned} R_k(m) &= 4R_{k-1}(m) + tt_{k-1}(m) + ss_{k-1}(m) \\ &\quad + 2sb_{k-1}(m) + cc_{k-1}(m) - adj_{k-1}(m) \qquad (1) \\ &= 4^{k-k_0} R_{k_0}(m) \\ &\quad + \sum_{j=k_0}^{k-1} 4^{k-j-1}\big(tt_j(m) + ss_j(m) + 2sb_j(m)\big) \\ &\quad + \big((4^{k-k_0}-1)/3\big)(m^2-4)(m-1), \qquad (2) \end{aligned}$$

by expanding out the recursion.

6. Minor recursions

For each of the terms in the summation of Eq. (2) (the major recursion in Section 5), we can write out a recursion, based on how the respective boundaries are constructed.

$tt_{k+1}(m) = 2 * tt_k(m) + ctt(m)$,

the cost at the connection between two top halves.

$ss_{k+1}(m) = ss_k(m) + bb_k(m) + css(m)$,

the cost at the connection at the middle of each side.

$sb_{k+1}(m) = sb_k(m) + ss'_k(m) + csb(m)$,

the cost at the connection at the middle.

$ss'_{k+1}(m) = 2 * sb_k(m) + css(m)$,

$bb_{k+1}(m) = 2 * ss'_k(m) + cbb(m)$,

the cost at the connection at the middle.

The ss cost is written different from the ss' cost due to the asymmetry of the s side. We call it ss when the two sides have the same orientation, and ss' when they have opposite orientations. All other terms involve at least one symmetric side, so there can be no difference on account of flipping orientations. Observe also that the ss' and bb terms do not directly contribute to the computation of R. However, they are required as part of the mutual recursion to compute some of the other terms, and so are included above. Finally, the tb combination is missing since it never occurs in the construction of the Hilbert curve.

For every $m \leqslant 2^k$ we observe that the number of runs in any of the *ss*, *sb*, etc., terms above can be obtained as the summation of the number of runs in each of the two regions being bounded independently. The savings due to connectivity of adjacent components is being accounted for separately, and this connectivity, when present, is only at the ends, and can never occur more than once in any $m \times m$ region.

It follows that we must have $ss_k(m) = ss'_k(m)$, and $ss_k(m) + bb_k(m) = 2 * sb_k(m)$. We use these identities to simplify the recursion above and obtain

$$tt_{k+1}(m) = 2 * tt_k(m) + ctt(m), \tag{3}$$

$$ss_{k+1}(m) = 2 * sb_k(m) + css(m), \tag{4}$$

$$sb_{k+1}(m) = sb_k(m) + ss_k(m) + csb(m), \tag{5}$$

where $ctt(m)$ represents the cost at the connection between two top halves, and $css(m)$ and $csb(m)$ similarly represent costs of appropriate connections.

The last two equations form a mutual recursion. However, we observe that what we are finally interested in are not the individual values of *ss* and *sb*, but rather the value of the summation $ss + 2 * sb$. So we write a single recursion by summing up Eqs. (4) and (5) with weights of 1 and 2 respectively and obtain

$$\begin{aligned} &ss_{k+1}(m) + 2 * sb_{k+1}(m) \\ &\quad = 4 * sb_k(m) + 2 * ss_k(m) + css(m) \\ &\qquad + 2 * csb(m). \end{aligned}$$

Writing e_k for $ss_k + 2 * sb_k$, and ce for $css + 2 * csb$ we get

$$e_{k+1}(m) = 2 * e_k(m) + ce(m). \tag{6}$$

Both Eq. (3) and Eq. (6) are straightforward recursions. The first gives a result of

$$tt_k(m) = 2^{k-k_0} * tt_{k_0}(m) + ctt(m) * (2^{k-k_0} - 1).$$

The second can be solved to yield

$$e_k(m) = 2^{k-k_0} * e_{k_0}(m) + ce(m) * (2^{k-k_0} - 1).$$

Plugging these back into the relevant terms of Eq. (2), we obtain

$$\begin{aligned} &\sum_{j=k_0}^{k-1} 4^{k-j-1} tt_j(m) \\ &\quad = \sum_{j=k_0}^{k-1} \left[2^{2k-2j-2+j-k_0} \left(tt_{k_0}(m) + ctt(m) \right) \right. \\ &\qquad \left. - 2^{2k-2j-2} ctt(m) \right] \\ &\quad = 2^{2k-k_0-2} \left[\sum_{j=k_0}^{k-1} 2^{-j} \right] \left[tt_{k_0}(m) + ctt(m) \right] \\ &\qquad - 2^{2k-2} \left[\sum_{j=k_0}^{k-1} 4^{-j} \right] ctt(m) \\ &\quad = 2^{2k-k_0-2} 2^{1-k_0} (1 - 2^{-(k-k_0)}) \\ &\qquad \times \left[tt_{k_0}(m) + ctt(m) \right] \\ &\qquad - 4^{k-1} 4^{1-k_0} (1 - 4^{-(k-k_0)}) ctt(m)/3 \\ &\quad = 2^{k-k_0-1} (2^{(k-k_0)} - 1) \left[tt_{k_0}(m) + ctt(m) \right] \\ &\qquad - (4^{k-k_0} - 1) ctt(m)/3. \end{aligned}$$

Following similar algebra, we also write

$$\begin{aligned} &\sum_{j=k_0}^{k-1} 4^{k-j-1} \left[ss_j(m) + 2 sb_j(m) \right] \\ &\quad = 2^{k-k_0-1} (2^{(k-k_0)} - 1) \left[e_{k_0}(m) + ce(m) \right] \\ &\qquad - (4^{k-k_0} - 1) ce(m)/3. \end{aligned}$$

With these terms plugged back in, Eq. (2) becomes:

$$\begin{aligned} &R_k(m) \\ &\quad = 4^{k-k_0} R_{k_0}(m) \\ &\qquad + 2^{k-k_0-1} (2^{k-k_0} - 1) \\ &\qquad\quad \times \left[tt_{k_0}(m) + ctt(m) + e_{k_0}(m) + ce(m) \right] \\ &\qquad + \left((4^{k-k_0} - 1)/3 \right) \\ &\qquad\quad \times \left[(m^2 - 4)(m - 1) - ctt(m) - ce(m) \right]. \end{aligned} \tag{7}$$

We thus have a closed-form solution that applies for all values of k, the size of the grid (which must, of course, be no smaller than k_0). There are a number of constants in this formula, all of which (including k_0) depend upon m, the specific size of region selected. We work out the exact numbers below.

7. 2×2 square regions

We drop the (m) specification in this section, writing tt_k rather than $tt_k(2)$, etc. in this section, since we know $m = 2$ through out.

$k_0 = 1$, since 2^1 is at least as large as m.

We have $R_1 = 1$; $tt_1 = ss_1 = 2$; $sb_1 = 3$; and $e_1 = ss_1 + 2 * sb_1 = 8$.

Working out the costs in the middle: $ctt = 4$; and $ce = css + 2 * csb = 2 + 2 * 3 = 8$.

Plugging in these constants into Eq. (7), we obtain:

$$\begin{aligned} R_k &= 4^{k-1} + 2^{k-2}(2^{k-1} - 1)(2 + 8 + 4 + 8) \\ &\quad - (4^{k-1} - 1)(4 + 8)/3 \\ &= 2^{2k+1} - 11 * 2^{k-1} + 4. \end{aligned} \tag{8}$$

Dividing this final closed form by the number of different positions yields:

$$(2^{2k+1} - 11 * 2^{k-1} + 4)/(2^{2k} - 2^{k+1} + 1). \tag{9}$$

It is easy to see that in the limit as k grows large, this formula asymptotically approaches a limit of 2.

Now we show that this approach to 2 is from below. We know that for $n = 1$, this formula is 1. Does it stay less than 2 forever, or does it oscillate?

$\langle numer \rangle / \langle denom \rangle \leqslant 2$ iff

$\langle numer \rangle - 2 * \langle denom \rangle \leqslant 0$.

Writing this out, we get the LHS to be $-3 * 2^{k-1} + 2$. This quantity is less than zero for all $n \geqslant 1$.

Therefore, for a 2×2 square region, the number of runs on average asymptotically approaches 2 from below as the size of the grid space is increased. The first few terms in this asymptotic approach are: 1, 14/9, 88/49, 428/225, ...

We conjecture that this number 2 is an asymptotic optimum. That is, no other linearization of two-dimensional space can do better, asymptotically, on the number of runs required to cover a 2×2 square region. Proving this conjecture is a subject for future research.

8. 3×3 square regions

We repeat the analysis of the previous section here for 3×3 regions to show how the same idea can easily be carried forward for any m.

R_1 is undefined, since $2^1 < 3$.

$k_0 = 2$; $R_2 = 10$. $tt_2 = ss_2 = 12$; $sb_2 = 14$; and $e_2 = ss_2 + 2 * sb_2 = 40$.

Working out the costs in the middle, we get $ctt = 16$; $ce = 36$.

Plugging these into Eq. (7), we obtain:

$$\begin{aligned} R_k &= 4^{k-2}(10) + 2^{k-3}(2^{k-2} - 1) \\ &\quad \times (12 + 16 + 40 + 36) \\ &\quad - (4^{k-1} - 1)(10 - 16 - 36)/3 \\ &= 4^{k-2}(10 + 52 - 14) - 104 * 2^{k-3} + 14 \\ &= 3 * 4^k - 13 * 2^k + 14. \end{aligned} \tag{10}$$

Thus, all one has to do, for any choice of m, is to work out manually the half dozen or so constants required, which can be done with a local piece of a Hilbert curve of size 2^{k_0+1}. Then one plugs these constants into the general formula.

9. Conclusions

The Hilbert curve is a popular technique for reducing the dimensionality of data while preserving adjacency as best as possible. The number of runs required to cover a region is an important measure of how well adjacency is maintained. Due to the complexity of the construction of the Hilbert curve, there have hitherto been no analyses of this number. In this paper, we present a closed-form solution for the number of runs required to cover a square region in a two-dimensional grid.

We showed that the expected number of runs for a 2×2 square region asymptotically approaches 2 from below, thereby establishing that the Hilbert curve is an asymptotically optimal linearization, at least for this case.

References

[1] T. Bially, Space-filling curves: Their generation and their application to bandwidth reduction, *IEEE Trans. Inform. Theory* 15 (6) (1969) 658–664.

[2] C. Faloutsos and Y. Rong, Spatial access methods using fractals: Algorithms and performance evaluation, Tech. Rept. UMIACS-TR-89-31, CS-TR-2214, Dept. of Computer Science, University of Maryland, 1989.

[3] H.V. Jagadish, Linear clustering of objects with multiple attributes, in: *Proc. ACM SIGMOD Conf.* (1990) 332–342.

[4] J. Nievergelt, H. Hinterberger and K.C. Sevcik, The grid file: An adaptable, symmetric multikey file structure, *ACM TODS* 9 (1) (1984) 38–71.

[5] J. Orenstein, Spatial query processing in an object-oriented database system, in: *Proc. ACM SIGMOD* (1986) 326–335.

[6] E.A. Patrick and D.R. Anderson, F.k. bechtel, *IEEE Trans. Comput.* 17 (10) (1968) 949–953.

[7] J.B. Rothnie and T. Lozano, Attribute based file organization in a paged memory environment, *Comm. ACM* 17 (2) (1974) 63–69.