SPACE-FILLING CURVES FOR DATA VISUALIZATION

SARAH LYELL

1. ACKNOWLEDGEMENTS

Most of this document was compiled during the background reading phase of my summer internship at the Institute for Human and Machine Cognition (IHMC) in Pensacola, Florida during the summer of 2023.

I am grateful to Larry Bunch, Jairun Diemert, Micael Vignati, my fellow interns, IHMC, and the Williams Alumni Sponsored Internship Program for their guidance and support this summer.

2. PRELIMINARIES

In this document, I aim to examine different layout options for large data sets, primarily through the use of (pseudo) space-filling curves. I also briefly overview another avenue: space-filling trees. As stated above, this document primarily corresponds to the background and research phase of my internship. However, the primary goal of the summer was to implement data visualization techniques informed by space-filling curves, not to pursue mathematical research. As such, this overview necessarily lacks mathematical depth and rigor and is aimed towards application. I also hope that it will be helpful to future interns who may want to continue the project regardless of mathematical background. Despite the focus on background, I do take the time towards the end of this document to review some of the avenues implementation took.

3. THE PROBLEM

The aim of this overview is somewhat general, but I will describe a potential problem we might wish to address. We are currently in the lengthy process of transitioning from IPv4 to IPv6, which presents new challenges for the visualization of network data. IPv6 theoretically allows for 2^{128} addresses (though the actual number is smaller) in comparison with IPv4, which theoretically allows for 2^{32} unique addresses. The transition to IPv6 has occurred for numerous reasons, one of which is the increasing problem of the relatively small address space of IPv4. IPv6 theoretically allows for 2^{96} times as many addresses, which is many magnitudes greater than the actual increase in size needed to accommodate the world's current need for unique IP addresses. As a result, the IP addresses in use in IPv6 are relatively sparsely distributed compared to those in use in IPv4. This presents a new challenge when it comes to visualizing IP data.

We also seek to look at methods for visualizing data that will preserve some degree of locality from index (think of as the number line) value to coordinate values in the plane determined by the curve. For instance, we would hope to find some kind of visualization that maps IP addresses corresponding to devices on the same network to be nearby in our two dimensional (or even three dimensional) visualization.

As it turns out, space-filling curves have these (and often other) nice properties. There are, however, many different kinds of space-filling curves. Part of my task for the summer was to determine what kinds of space-filling curves would be useful to the kinds of visualizations already being done by the research group I was working with at IHMC. But before we get to specific curves, I will provide a short background for those unfamiliar. Be sure to also see the last section of this paper for some helpful introductory materials.

Date: June 2023.

4. What is a Space-Filling Curve?

There are a few different ways we can think of a space-filling curve. Formally, a space-filling curve is a curve whose range reaches every point in a higher dimensional region.

The textbook Space-Filling Curves: An Introduction With Applications in Scientific Computing gives the following definitions:

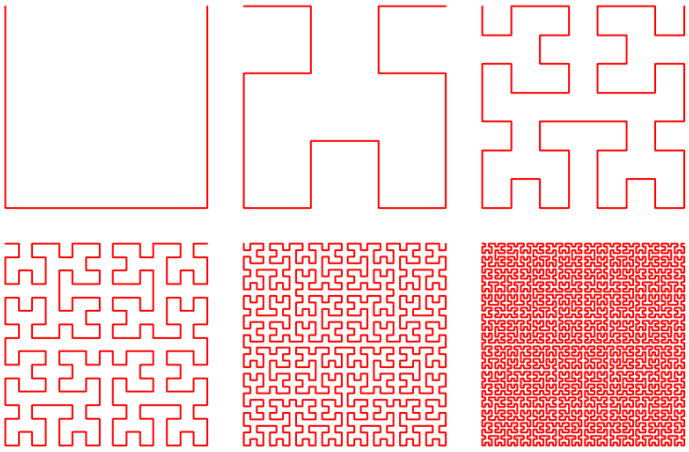
Definition 4.1 (Curve)**.** Let $f : \mathcal{I} \to \mathbb{R}^n$ be a continuous mapping of the compact set $\mathcal{I} \subset \mathbb{R}$ into $\mathbb{R}^n$. The respective image $f_*(\mathcal{I})$ of such a mapping is then called a **curve**, and $x = f(t), t \in \mathcal{I}$ is called a **parameter representation** of the curve.

Definition 4.2 (Space-filling curve)**.** Given a curve $f_*(\mathcal{I})$ and the corresponding mapping $f : \mathcal{I} \to \mathbb{R}^n$, then $f_*(\mathcal{I})$ is called a **space-filling curve** if $f_*(\mathcal{I})$ has a Jordan content (area, volume, ...) larger than 0.

We can think of a space-filling curve as mapping a lower dimension into a higher dimension or as mapping a higher dimension into a lower dimension. For our purposes, these will be equivalent. For instance, we can think of a Moore curve as a mapping from a line to a plane or as a mapping from a plane to a line. Doing so allows us to make use of various literature, some of which defines space-filling curves as mapping from higher to lower dimensions and some of which defines space-filling curves as mappings from lower to higher dimensions.

If these definitions are not useful to you, don't worry too much. They won't be strictly necessary to understand the rest of this document, but I thought I ought to include them for completion.

In this document I will often refer to a pseudo space-filling curve as a space-filling curve for simplicity, even though this is not technically correct. A lot of space-filling curves are constructed as the limit of the iterations of the pseudo space-filling curve (informally). For example, in the graphic below we see a number of iterations of a pseudo space-filling curve. The Hilbert space-filling curve is really the limit of the iterations of such curves.



Source: `http://www.datagenetics.com/blog/march22013/`

Since we will be dealing with a discrete number of data points, we are not as much concerned with the "real" space-filling curve as we are with whatever order of the corresponding pseudo space-filling curve that provides an adequate level of refinement tracing a path through a discrete number of points.

5. Past Uses

Currently, researchers from many different fields have made use of space-filling curves to visualize large data sets. The Hilbert curve in particular, which will be illustrated later, has been widely used. In this section, I will provide a few examples of past uses of space-filling curves for data visualization to motivate the further analysis and to provide inspiration for possible further or more relevant applications.

5.1. **Genomic data visualization.** In one case of genomic data visualization, the use of a Hilbert curve was primarily in providing an overview of data to help inform further analysis and visualization techniques. In cases with many data points (the example uses approximately 1.8 million), simply plotting the data on a vector does not provide an easy way to visualize peaks or clusters in the data. Simply zooming in, or only viewing one section of the plot, would not allow for an overview of all or enough data to spot patterns easily. The benefit of something like a Hilbert curve plotting technique is that it allows for an overview of a very large section of data while still providing sufficient detail to pick out relative peaks or clusters in the data without too much difficulty.

Visualization of Genomic Data with the Hilbert Curve [reference] provides the following figure to illustrate this property.

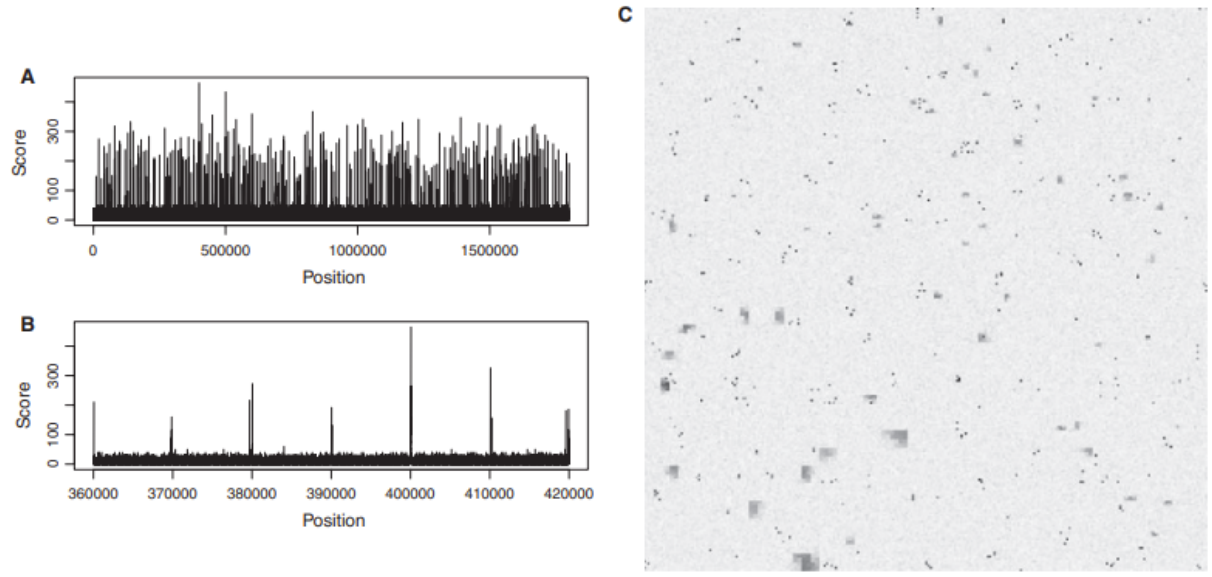


Fig. 1. A data vector with constructed example data to demonstrate the use of HCV. **(A)** A standard plot of the whole vector only shows that the data seems to contain many peaks. **(B)** A zoom-in offers only limited insights. **(C)** The HCV shows (see main text) that the data contains narrow, tall peaks which appear in clusters, as well as wide peaks that are distributed randomly but have even larger width in one quarter of the vector.

The paper from which this example was drawn suggests that a Hilbert curve is the optimal space-filling curve for such examples, though, as we will see, other sources suggest this may not necessarily be true. Still, the fact that the Hilbert curve has been successfully implemented in other software programs is promising.

5.2. **IP address visualization.** Hilbert curves have also been used to visualize IP address data.

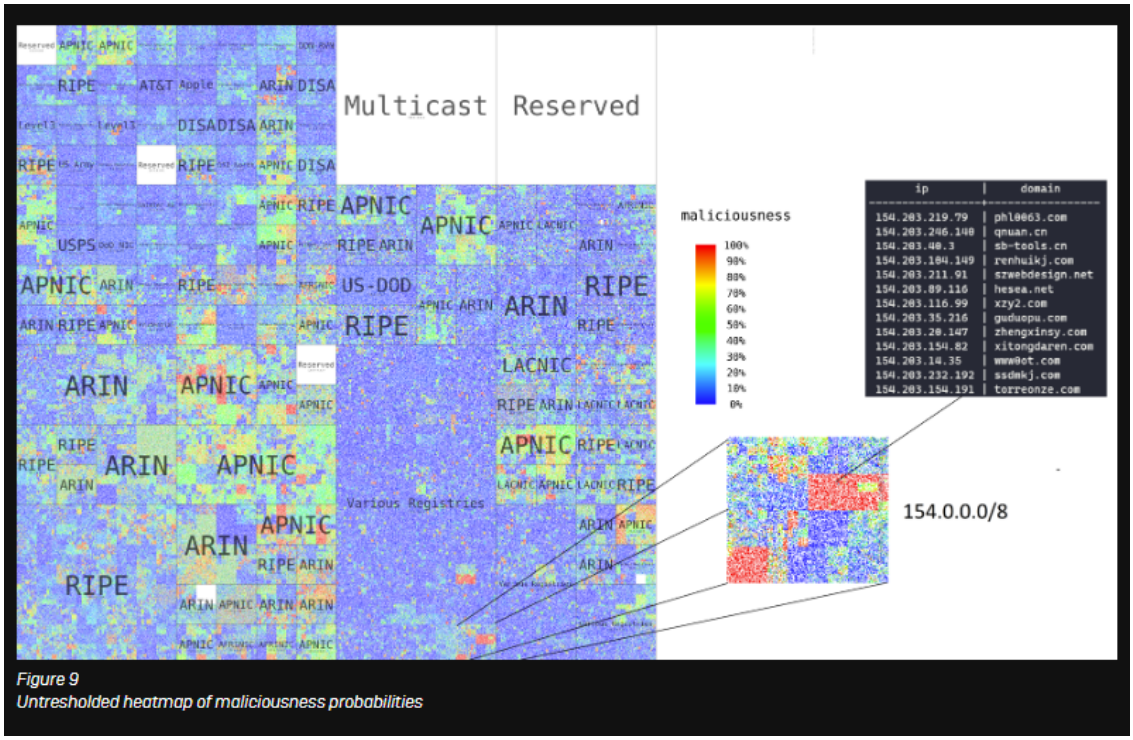


Figure 9
Untresholded heatmap of maliciousness probabilities

Source: Sophos AI blogpost: A machine learning approach to inferring the maliciousness of unknown IP addresses, autonomous systems, and ISPs

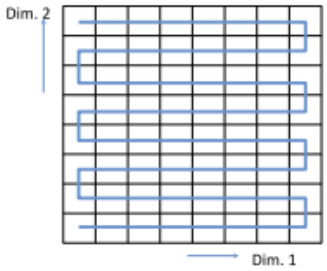
Though this might not be exactly the way that we would want to visualize something like IP address data, it provides an example of what such a visualization might look like.

There are a few other example visualizations. The link directly above includes more examples, and even more can be found at the following links:

- IPv4 heat maps using Hilbert and Morton curves (presentation by Roy Arends for Nominet UK)
- Short explainer on visualizing IP data with space-filling curves
- Blog post: Mapping IPv4 Address (with Hilbert curves) in R
- Mapping the whole internet with Hilbert curves

6. What is Actually Special About Space-Filling Curves?

To fully appreciate the potential usefulness of space-filling curves, we will consider a non space-filling curve, the so-called "snake curve" or "row-major" curve.



(b) the row-major SFC

Source: Optimality of Clustering Properties of Space-Filling Curves

I've read some literature by computer scientists who seem to tout the "snake curve" or other curves like the so-called "onion curve" as being the best space-filling curves for locality preservation. I found this to be odd. First of all, these are technically not space-filling curves, though this might not be of too much importance to a computer scientist who is looking for something that might be called screen-filling over space-filling. After all, everything a computer deals with is discrete, so it could be forgiven that these constructions are not technically space-filling. However, there is a reason these kinds of curves are not really useful to us.

Space-filling curves have the nice property that with increasing order (of pseudo space-filling curves), the output of whatever function takes an index point to a two-dimensional point on the plane will approach a fixed point.

As an example, working over the real numbers, if we wanted to see where the point corresponding to $.25$ (if we are mapping from $[0,1] \to [0,1]^2$ for different numbers of points (i.e. for different orders of pseudo Hilbert curves), the Hilbert curve (and all truly space-filling curves) have the nice property that $HilbertMap(.25, n)$ (if we think of the Hilbert curve as a function that takes a number $a \in [0,1]$ and an iteration number, n) approaches a fixed point in two dimensions. Conversely, the snake curve does not have this property. Though $SnakeMap(.25, n)$ will generally map $.25$ to some (x, y) value where y is about $.25$, its x value will oscillate. This is illustrated quite well in a 3Blue1Brown video linked at the end of this document.

Why do we care? Well, if you are a human using a particular visualization tool, it would be nice if you could develop intuitions about the data you are looking at regardless of how many points you are looking at. Your intuitions for a particular curve would scale across different sizes of data. Or, for instance, if you are trying to map out IP addresses, the layout of a group of IPs you have already mapped will be consistent. If you map four IP addresses (say with IPs 0,1,2,3) with an order 1 Z-curve and then decide you want to map 16 addresses with an order 2 curve, the original 4 addresses will still be laid out in the same relation to each other, though now they will make up only a quarter of the overall space. In particular, due to the specific construction of it, IP addresses on the same network will continue to be laid out near each other over successive iterations, which is quite nice.

7. Some Candidates

The following are some potential space-filling curves on which we can model our visualization. Our initial focus is curves that fill a plane, though some of these curves can be extended to three dimensions.

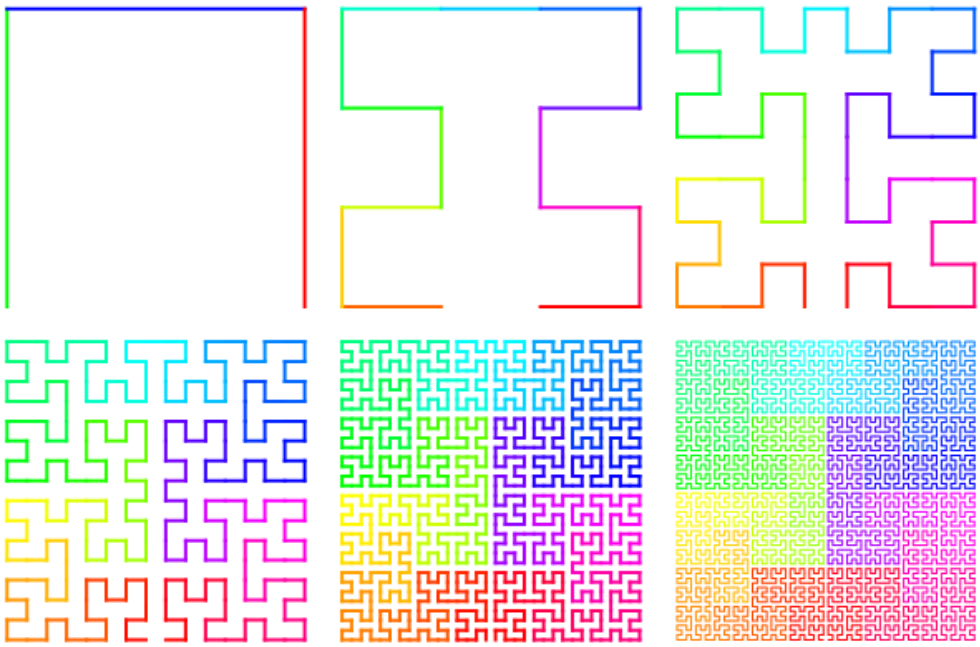
- Moore curve
- Morton curve
- Hilbert curve
- Dragon curve
- Peano curve
- Gosper curve
- 3D Moore curve
- 3D Morton curve
- 3D Hilbert curve

The above were chosen primarily for the abundance of research available on their properties and their current uses in applications. This is, of course, far from an exhaustive list, and it is quite possible that there are useful curves that have been omitted.

8. Visualiazing Our Options

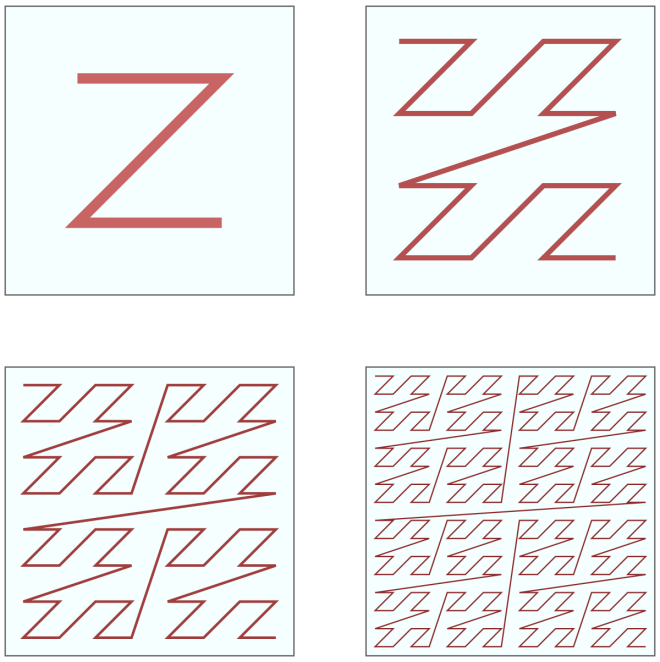
In this section, you can find some visual representations of the space-filling curves given above with sources provided. These have been retrieved from various internet resources, so they are intended to provide visual intuition for some of the different curves available. However, they should not be used for the purpose of direct comparison. If a more direct comparison between a specific set of curves at a specified resolution, it is fairly straightforward to generate such images using a software such as mathematica, though I have not done so here.

8.1. **Moore curve.** The following is an illustration of stages 0 through 5 of the Moore curve



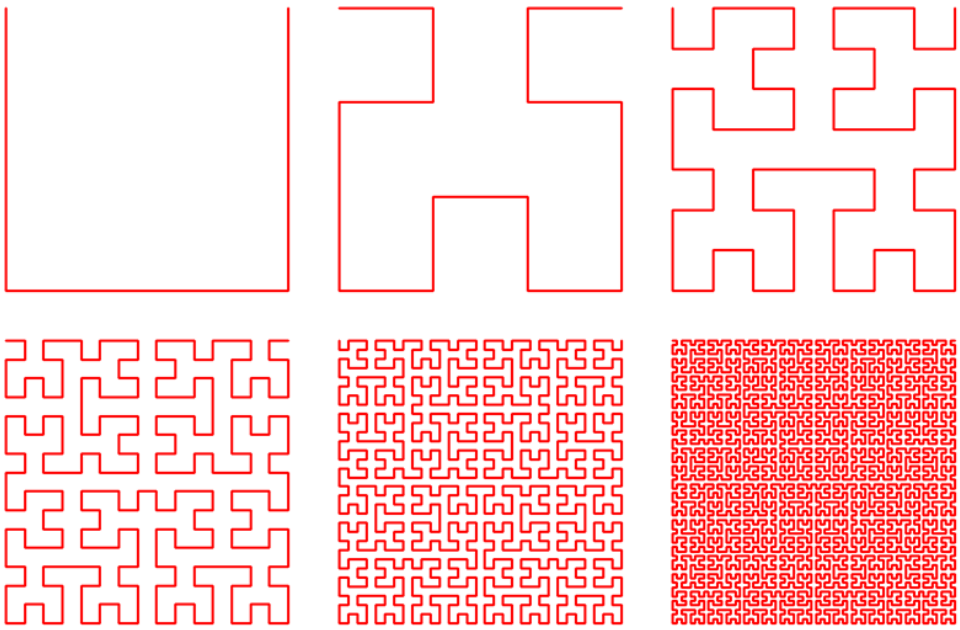
Source:
https://en.wikipedia.org/wiki/File:Moore-curve-stages-0-through-5.png

8.2. **Morton curve (Z-curve).** The following graphic displays four iterations of the Z-curve or Morton curve



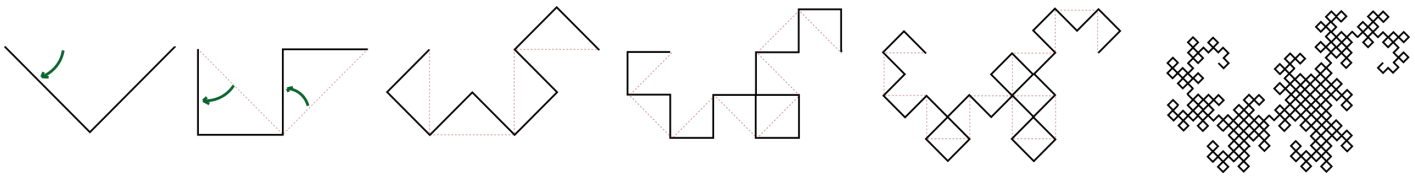
Source:
https://en.wikipedia.org/wiki/Z-order_curve#/media/File:Four-level_Z.svg

8.3. **Hilbert curve (2D).** The following graphic displays 6 iterations of a two-dimensional Hilbert curve



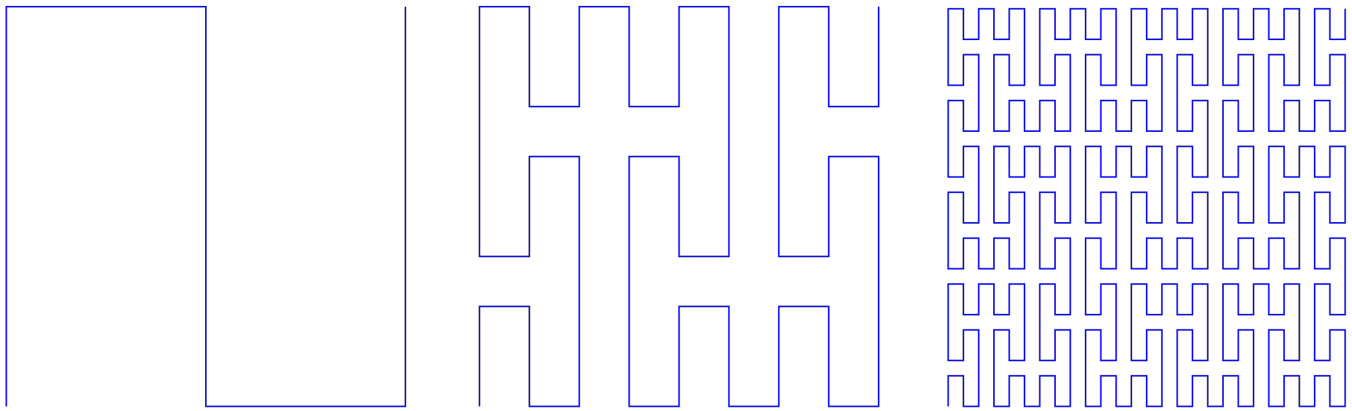
Source: http://www.datagenetics.com/blog/march22013/

8.4. **Dragon curve.** The following graphic displays a few iterations of the dragon curve



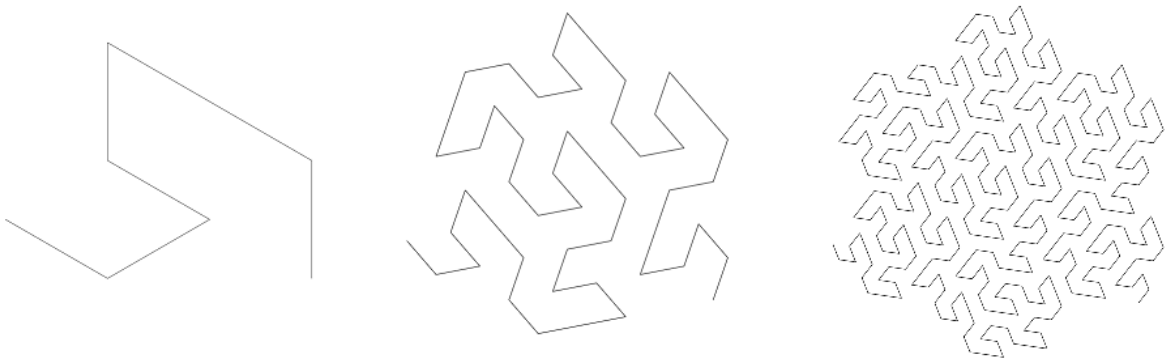
Source: https://en.wikipedia.org/wiki/Dragon_curve#/media/File:Dragon_curve_iterations_(2).svg

8.5. **Peano curve.** The following graphic displays a few iterations of a Peano curve



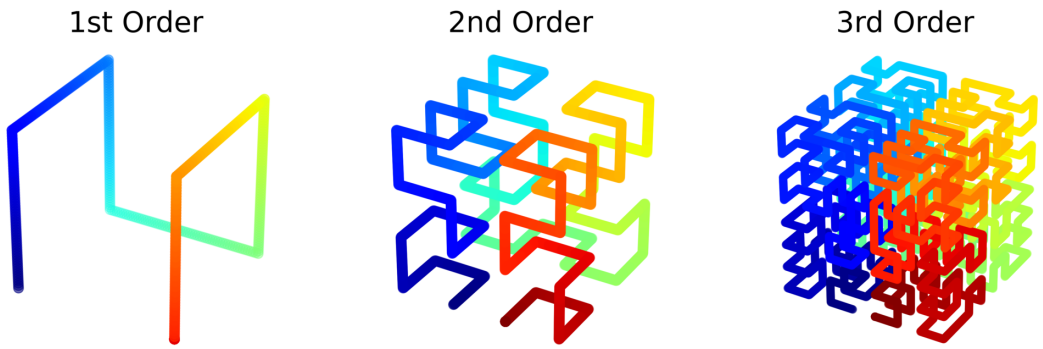
Source: https://en.wikipedia.org/wiki/Peano_curve#/media/File:Peanocurve.svg

8.6. **Gosper curve.** The following graphic shows a few iterations of a Gosper curve



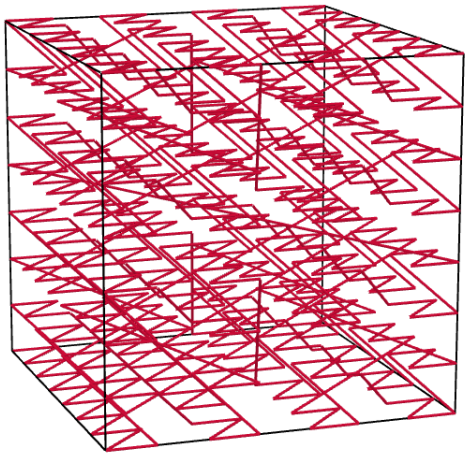
Source: Screen shots taken from this interactive tool https://www.geogebra.org/m/T38QSY7Q

8.7. **3D Moore curve.** The following graphic illustrates a few iterations of a 3D Moore curve



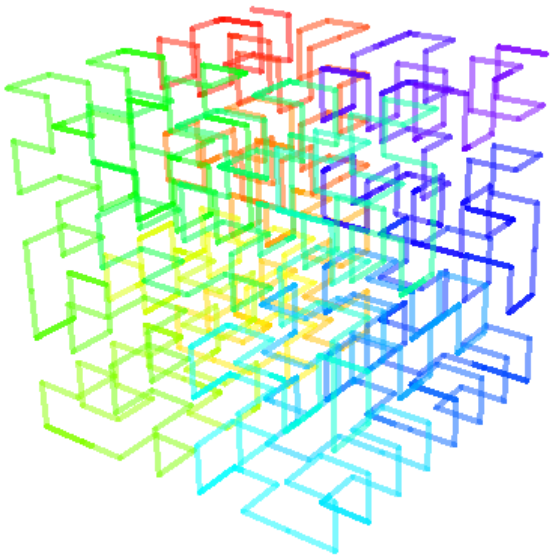
Source:
`https://en.wikipedia.org/wiki/Moore_curve#/media/File:Moore_curves_3D.png`

8.8. **3D Lebesgue curve.** The following shows a single iteration of a 3D Lebesgue curve.



Source: Screenshot taken from this interactive tool `https://www.geogebra.org/m/uvadkpfx`

8.9. **3D Hilbert curve.**



Source: `https://en.wikipedia.org/wiki/Hilbert_curve#/media/File:Hilbert3d-step3.png`

9. Specifics: Data Visualization

In the visualizations above, we see the curves in a general space. However, it seems as though we might like to map curves into a grid to represent discrete data points (or pixels). In this case, the space-filling curve provides a path that determines the correspondence between two dimensional grid coordinates and one dimensional data points.

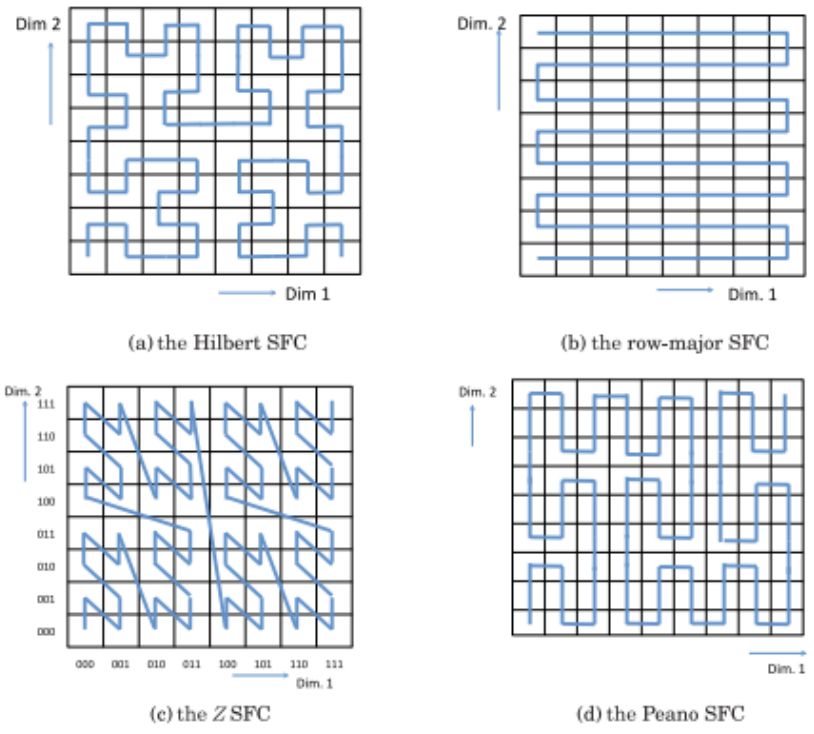


Fig. 1. Some popular SFCs in two dimensions.

Source: Optimality of Clustering Properties of Space-Filling Curves

Some of the curves we have seen so far seem to lend themselves more to a "grid filling" or "screen filling" application. As is illustrated above, the Hilbert (and Moore, as it is a version of the Hilbert), Z (aka Morton), and Peano curves lend themselves to this readily. It also seems like the Gosper curve traces a path through hexagons. Conversely, the dragon curve doesn't do this. Additionally, the final shape that it takes after many iterations is not obviously useful to any of our applications, though there may be cases for which it is useful.

10. Making Comparisons

There are numerous ways that we might make comparisons between different space-filling curves. One thing we might like to know is the degree to which different space-filling curves preserve the proximity of neighboring points in one and two dimensions. Even within such a metric, there are many ways we could go. For instance, do we care more that the neighboring points in one dimension (linear order) are close to each other in two dimensions or that the points nearby each other in two dimensions are relatively near each other when we consider the corresponding one dimensional points.

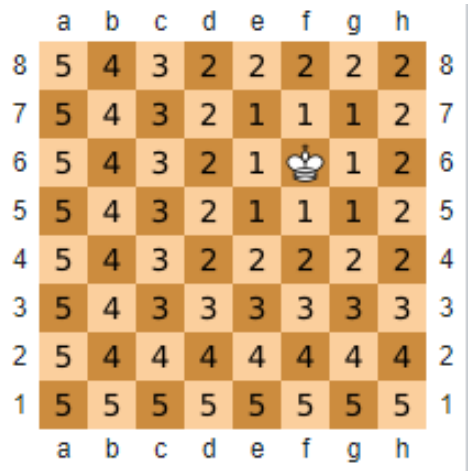
One possibility, suggested by Xu and Tirthapura in [Lower Bound on Proximity Preservation by Space-Filling Curves] is to consider the Nearest Neighbor Stretch. Xu and Tirthapura consider cells in a grid like the one above that are adjacent to one another (in the sense that they share an edge) and calculate the average increase in distance between nearest neighbor pairs in high dimensions (2 or 3 in our case) when the points are mapped into one dimension by a space-filling curve. That is, they calculate the average stretch for each of the "nearest neighbors" (distance ≤ 1) according to the Manhattan distance metric when the two-dimensional grid points are considered as a linear order. I don't think is the most useful metric for our purposes, and I will elaborate on why that is and provide a modification as an alternative in the following section.

From an implementation perspective, it also makes sense to consider the computational ease of converting between one and two-dimensional coordinates (and even three-dimensional, though I did not focus too much on this). Converting back and forth between 1D indices and 2D coordinates according to the Morton order, for instance, is relatively simple and can be accomplished with bit manipulations.

And of course, we have to consider whether or not all of these differences are preserved across different orders of curves. Is the Morton curve more locality preserving than Hilbert in some band of orders but not in others? That is, does the amount of discrete boxes we want to map through space change the curve that is optimal?

11. Refinements of Some Suggested Metrics: What Counts as a Nearest Neighbor

Strictly from a visualization perspective, the use of Manhattan distance does not make much sense. Instead, we might consider a metric like Chebyshev distance. What would this mean for our comparisons? Well, it would allow us to include those boxes diagonal from a box in its group of nearest neighbors. I think that this makes quite a bit of sense from the perspective of easily analyzing clusters of data visually. Using the Euclidian metric would also have this property, but we would lose some of the benefits of Manhattan distance, like its natural application to a grid of pixels. With Chebyshev distance, we can keep (to a degree) the visual intuition provided by using Euclidian distance and some of the computational benefits of the Manhattan distance.



Source: Chebyshev Distance Wikipedia

The illustration above shows the Chebyshev distance for different squares in relation to the King in f6. If we are considering the nearest neighbors of f6 using Chebyshev distance, all the squares labeled with "1" would be considered. If we were considering the nearest neighbors using the Manhattan distance, we would only count e6, f7, g6, and f5–or those squares that share an edge with f6. We would be leaving out those squares diagonal from the king, which I think is a mistake if we want to use a metric that appeals to our visual intuition.

This should help resolve an oddity in the literature in which a snake curve is characterized as just as locality preserving as a Hilbert curve or other truly space-filling curve when this is directly contrary to our intuition about what it means to be locality preserving.

12. Comparing Curves Computationally

As part of my background reading and research, I created a small JavaScript program to visualize some of these different curves. Beyond giving me a means to practice JavaScript before working on the larger JavaScript applications of my team, this also provided a way to computationally compare some of the different curves. Specifically, I sought to compare the Hilbert, Moore, and Morton curves, as those seemed to be the most promising for the applications in which we were interested.

Below are the results of a few tests that I ran:

Average Nearest Neighbor Stretch

x: order, y: curve	2	3	4	5	6	7	8	9	10
Hilbert	2.87	5.99	19.52	49.22	152.11	480.54	1587.69	5517.00	undefined
Morton	2.72	5.37	11.04	22.71	46.38	94.04	189.71	381.38	765.04
Moore	3.67	7.82	26.29	63.08	195.88	631.49	2091.15	7318.45	undefined

Median Nearest Neighbor Stretch

x: order, y: curve	2	3	4	5	6	7	8	9	10
Hilbert	2	2.5	3.5	8.5	19.5	32.5	77.5	179.5	351.5
Morton	2	3	3	4	4	4	4	4	4
Moore	2.5	2.5	3.5	8.5	19.5	32.5	78	179.5	undefined

I computed the "Average Nearest Neighbor Stretch" as follows: for each box in the curve, I computed the average "stretch" (or increase in distance) that occurred when each of its nearest neighbors (defined as those boxes Chebyshev distance 1 away) were transformed from their 2D coordinates back to their 1D indices. I then took the average across all boxes for each order to calculate the ANNS displayed in the chart above.

I followed a similar approach for the "Median Nearest Neighbor Stretch," but wherever I took the average before, I took the median.

I was somewhat surprised the the Morton curve outperformed the Hilbert and Moore curves so greatly at larger orders. The Morton curve has large jumps between some successive indices. However, this appears to be offset by the fact that most successive indices are quite close to each other. Further, it is relatively less common that adjacent boxes are very far away from each other (though it still happens) unlike the Hilbert curve. As one would expect, the Hilbert and Moore curves measured quite similarly, as the Moore curve is essentially a version of the Hilbert curve with different start and end points.

Both of these metrics are flawed, and there are many other metrics along which these and other curves could be compared. For the sake of moving on to implementations, I did not spend much more time considering other metrics. The results above lent me enough confidence that Morton encoding would work well for the IP use-case.

13. Suggestions

In this section I provide some suggestions for when different space-filling curves would be useful. In general it seems as though the Hilbert, Moore, and Morton curves are the most promising for data visualization uses, but I have not been exhaustive in my research. For now I will focus my suggestions to these curves, though.

13.1. **IP Addressing.** As stated in the previous section, the Morton curve seems like the best choice for IP address visualization. The Morton curve is encoded through bit interleaving, which has the bonus of creating clusters that match those formed by IP addressing. Throughout my background reading, I sometimes saw it stated casually that the Hilbert curve was the best space-filling curve for locality preservation. However, as my results above demonstrate, this is not the case across all metrics, although I can't comment on the general case. The fact that Morton order can be easily determined (for getting a two dimensional coordinate from a one dimensional coordinate and from getting a one dimensional coordinate from a two dimensional coordinate) is another bonus. Using some bit masking tricks, bits can be easily interleaved and de-interleaved quite quickly.

13.2. **Other use-cases.** For other use-cases, it seems to make sense to ask a few questions to determine which space-filling curve would best meet the needs of the project. If you already know you would like to use a space-filling curve, there are two main questions that differentiate the Hilbert/Moore and Morton curves.

First, is the speed of computation very important to you when determining the one dimensional index from the two dimensional index or vise-versa? There are algorithms that make calculating Hilbert/Moore coordinates relatively fast, but it seems unlikely that they will approach the speed of the Morton encoding. If this is the case, you might lean towards the Morton encoding.

Second, do you care that your mapping is strictly continuous in the sense that all adjacent points in the one dimensional ordering have to be adjacent in the two dimensional ordering? If this is the case, the Hilbert or Moore curves are likely the best bet (without making other considerations). IP addresses work particularly well because the jumps in the Morton encoding correspond to places where numerically adjacent IP addresses are unlikely to be on the same network. If your data has similar jumps, it might be worthwhile to consider if the Morton encoding works well.

Finally, if you are deciding between the Moore and Hilbert curves, you should consider the endpoints of your data. Does your data "wrap around"? That is, do you want your first and last indices to be close to one another? If that is the case, the Moore curve is likely the best choice. If not, or if it is not particularly important to you, the Hilbert curve is probably best.

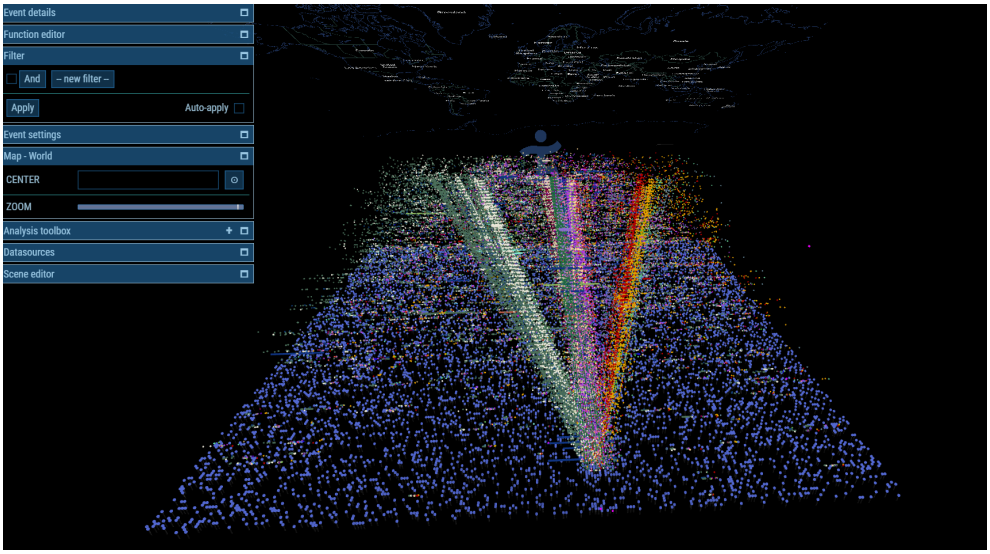
14. Looking Back: What We Went With

I wrote the majority of the above during the first few weeks of my internship when I was primarily doing background reading and familiarizing myself with the project at hand. Though I aimed to compile a report of

some of my work and a list of resources that might be useful to those who wish to continue it, my primary motive was to keep track of what I had learned to aid in implementation. I implemented a variety of the above curves in a web app and measured them computationally as described in section 12.

I wrote code in JavaScript for generating such curves at first using L-systems. While this was an easy way to start generating visualizations and gaining intuition for different curves, such an implementation would not really be practical in actual applications. So, I also wrote code (which can be found in the same repository) for non-L-system-based generations of different orders of Hilbert, Moore, and Z-curves. These implementations allow one to call functions to translate between curve and index for up to 32 bit indices. The Z-curve coordinates are generated independent of the order of the curve, but the Hilbert and Moore coordinates depend on the order of the curve. If you wish to have a curve that defines x indices, you will need an order of curve such that $4^{order} \geq x$. Note that the implementations I have written for the Hilbert and Moore curves are still recursive, but with a maximum number of iterations. That is, if you wish to know the coordinates of a point on a Hilbert (or Moore) curve of order 4, the function will only be called recursively up to 4 times. I believe that it is possible to rewrite the code to generate Hilbert curves non-recursively using similar ideas to the recursive code provided. I have not done this for readability, though it may be worth adding a non-recursive version. Doing the same for the Moore curve would be more difficult, and I am not sure if it could be done given that the Moore curve is generated from a modification of the previous order of Hilbert curve. There may be a way to generate Moore curves non-recursively, but not one that I am aware of.

14.1. **Visualizing IP Hits.** The particular implementation that I ended up working on was laying out the boxes representing different IP Addresses in our netflow application. In this application, we provide an overview of all network traffic incoming to IHMC servers. Using data from a database, we match IP addresses to latitude and longitude. The visualization is the flow of data from the geographic location associated with the source IP address to different IP addresses on the IHMC Network. Colors of glyphs (representing different events) correspond to their country of origin, and the width of each glyph corresponds to the number of bytes transmitted in a given connection. There are only about 60 servers on the network (at least that were already visualized and receiving public network traffic), so laying these out differently does not provide much benefit. To test some of the ideas I had worked on, I generated artificial data to visualize the flow to a greater number of IP addresses. I used the Z curve in the layout. The benefit of such a curve was that it clearly grouped the boxes corresponding to the IHMC network together. The following is a screenshot of the application mid flow. The boxes are laid out according to a Morton encoding as determined by the last three bytes of the IP addresses. Boxes are only generated for IPs that are "hit," so there are not boxes in every place that there could be.



In the artificial data that I generated, I duplicated an existing data set, changing only the the ids and destination IP addresses. Then, I visualized the flow of 800,000 rows of data by combining the original data set with

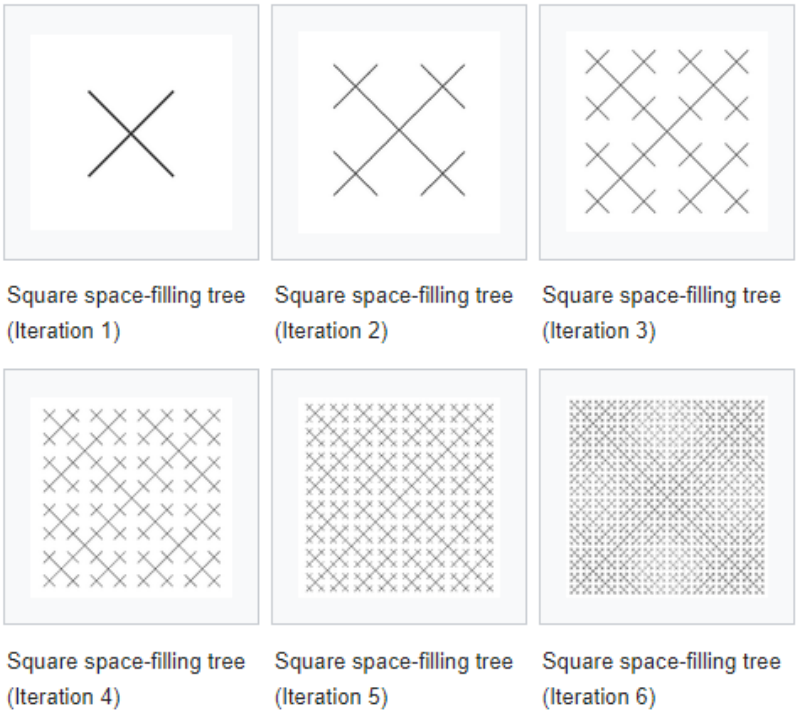
the artificial data. So, as we would expect, about half of the data in the above photo is flowing towards the IHMC network, while half is flowing to other areas of the address space.

14.2. **What Happened to IPv6?** As it turns out, IPv6 is incredibly sparse. Moreover, IPv6 addresses are quite long. Since the team is not primarily working on IPv6 addresses, we decided it would not be useful to spend too much time worrying about them during my internship. Additionally, it would probably be best to work with IPv6 addresses using tools other than those the team is currently using. Given that the original application was only designed to layout up to 60 addresses, it would take a redesign to be able to visualize the range of addresses used in IPv6 in any helpful manner. Still, the patterns above were chosen in part due to their scalability and could plausibly be extended to map out IPv6 addresses. For example, it would be easier to work in a language that supported larger numbers and their manipulation, though it would still be plausible to do so in JavaScript with some additional tricks and overhead. If we could work in a language where we had a 128-bit unsigned integer (or could more easily simulate the existence of a 128 bit unsigned integer), we would have a unique bit pattern for each of the 2^{128} addresses of IPv6 using a Morton encoding.

15. Other Options: Space-Filling Trees

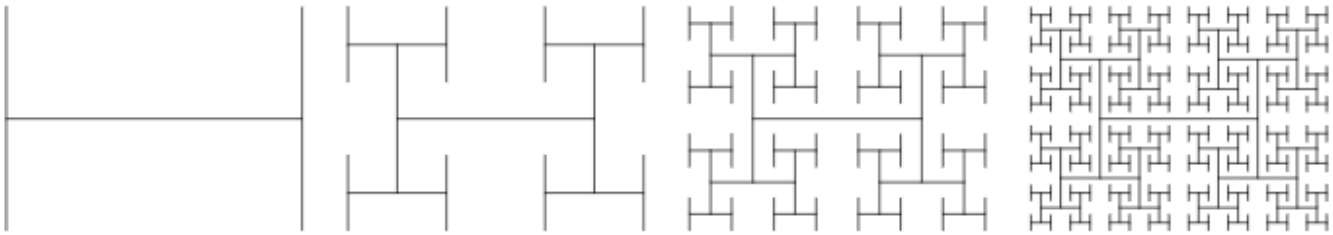
There is another type of algorithm we could consider. Instead of designing an algorithm to model a space-filling curve, we could model such an algorithm after a space-filing tree. This might be useful to do in cases where there is not a natural order to the data being considered but there is some kind of clustering or structure we might like to display. An example of such a space-filling tree is an H fractal or 3D H fractal, displayed below:

15.1. **Square space-filling tree.** The following graphic illustrates the most simple kind of space-filling tree: a square space-filling tree



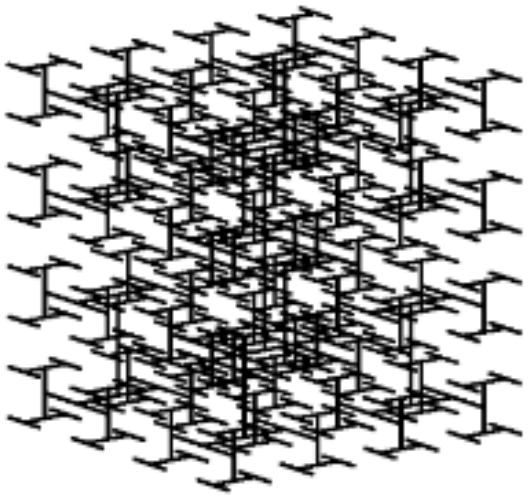
Source: `https://en.wikipedia.org/wiki/Space-filling_tree`

15.2. **H fractal (H tree).** The following graphic shows a few iterations of an H-fractal



Source: `https://mathworld.wolfram.com/H-Fractal.html`

15.3. **3D H fractal (3D H tree).** The following graphic shows one iteration of a 3D H fractal



Source: `https://upload.wikimedia.org/wikipedia/commons/c/cc/3D_H-fractal.png`

I did not explore this as much, though it is a possible avenue for further research.

16. References, Resources, and Further Reading

- 3Blue1Brown Video on the Hilbert Curve, provides a nice introduction to space-filling curves
- Space-Filling Curves, a fairly readable textbook and reference on space-filling curves and their applications in scientific computing
- Generating Hilbert Curves Using L-Systems, short paper
- Space-Filling Curves: Hans Sagan, oft-cited textbook for mathematical approach to space-filling curves
- Optimality of Clustering Properties of Space-Filling curves, paper from which I took some of the above figures. However, the authors seem to take a loose definition of space-filling curves, so it may be of limited use. I did read it early on in my background reading and felt it helped me get a grasp for the subject, though.
- A lower bound on proximity Preservation by Space-Filling Curves, similar to the previous reference, the authors take a loose definition of space-filling curves, but it may be helpful background reading.

There are, of course, many publications on the clustering properties and various other metrics associated with space-filling curves. Given that the goal of this summer internship was not to pursue mathematical research, I have omitted most of them here as I did not spend too much time examining their contents. I have, however, kept a list of papers I have run across that seem relevant (even if I cannot vouch for them directly), and I can provide the list if it is of interest. Even so, it is by no means complete and was compiled in a haphazard fashion, so I would not lend it too much weight.