A Rigorous Derivation of the Bubble Sort Curve

Michael DiFranco

March 2024

Abstract

Sorting algorithms are often visualized by videos of bar plots or scatter plots whose elements are gradually sorted by height. When bubble sort is displayed in this way, the plots appear to approximate a curve. We formalize this observation by developing a precise definition of what it means to say that a sorting algorithm's scatter plots approach a shape. We then prove that bubble sort approaches a specific shape comprising a diagonal line and the region under a hyperbola. Our definitions and proofs use tools from elementary analysis and probability theory.

1 Introduction

It is common to visualize sorting algorithms with videos of two-dimensional diagrams where the items in the list are represented by objects which are sorted according to their height. The most common of these are bar plots and scatter plots. (See Figure 1) Regardless of the specific representation, these videos reveal that certain algorithms "shape" the data in certain ways as they run. For most algorithms, the shapes of the diagrams are obvious to anyone with a working understanding of the algorithm. But that is not the case for bubble sort.

Bubble sort is one of the simplest sorting algorithms. It involves repeatedly scanning across the list and swapping adjacent items whenever they are out of order. When bubble sort runs on a large, uniformly shuffled list, its diagram forms a distinct curve. (See Figure 2) However, it is not so clear why this particular algorithm should appear as a curve, and it is even less clear what

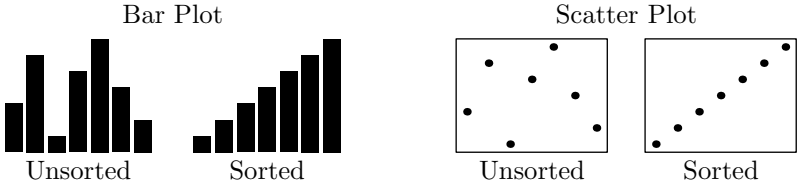

Figure 1: The bar plot and scatter plot representations of a list of 7 items

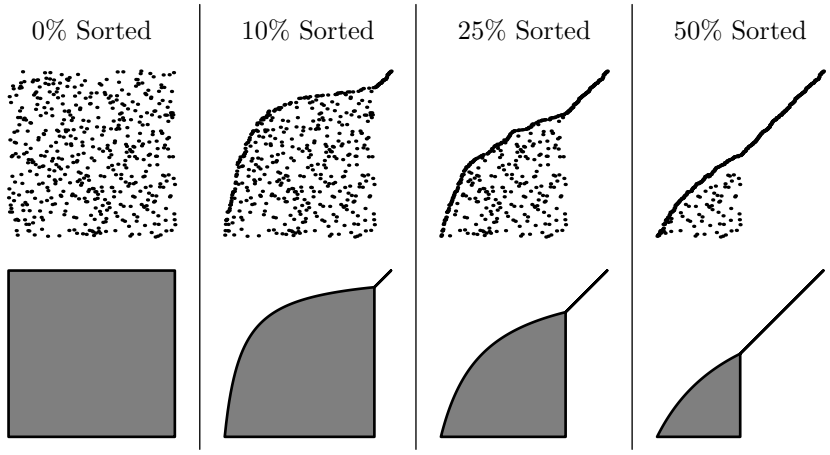

Figure 2: A list of 500 items at different moments in the execution of bubble sort, along with the shapes the the diagrams approximate.

exactly this curve is. It is possible to intuitively derive a formula for the curve without too much hassle, but we would need to lean on a handful of assumptions, as well as an overly vague notion of what it means to say that the diagrams approach a shape. This paper is not devoted to such a casual approach, but rather to a rigorous formulation and proof of the problem.

We will interpret the scatter plot representation of bubble sort as a finite set of points in three dimensions (Two dimensions x and y for space, one dimension t for time). We will formulate a precise definition of what it means to say that a sorting algorithm's diagram approaches a shape (Definition 8). We will then prove that bubble sort does indeed approach a shape as per our definition. In particular, we will prove that the diagrams approach the set

$$\left\{(x,y,t) \in [0,1]^3 : y \leq \frac{x}{x+t},\ x \leq 1-t\right\} \cup \left\{(x,x,t) \in [0,1]^3 : x > 1-t\right\}.$$

The shapes in the bottom of Figure 2 are t-cross sections of this set with t values of $0, 0.1, 0.25$, and 0.5.

1.1 A Brief History

The problem of efficiently sorting lists is an important and extensively researched topic in computer science. Since the mid 20th century, a multitude of different sorting algorithms have been invented. Bubble sort entered the scene in 1956 when it was described by Edward Friend, who noted that other simpler algorithms, such as insertion sort, outperformed it [3]. Despite some subsequent refinements, it was never able to overcome this first impression. Notably, Knuth gave a detailed analysis of its performance in 1973, with the conclusion that, "bubble sort seems to have nothing to recommend it, except a catchy name and

the fact that it leads to some interesting theoretical problems" [5]. (The shape of the algorithm's visualizations likely was not one of the interesting theoretical problems on his mind.) Therefore, from a practical point of view, bubble sort has been fully understood for decades. But despite its absence from real-world applications, it is a classic, almost always making an appearance in sorting algorithm visualizations.

Sorting algorithm visualizations have been around at least since 1981, when Ronald M. Baecker produced *Sorting out Sorting* [1]. This film showcases the bar plot and scatter plot representations of several sorting algorithms, including bubble sort. Since then, as computers became orders of magnitude more powerful, the number of programs devoted to algorithm visualization rapidly increased. Today, the largest platform for these videos is likely YouTube, where users have been uploading visualizations since late 2006. Some of these videos have gone viral, accumulating millions of views, and thereby exposing sorting algorithm visualizations to many who otherwise have no connection to computer science. Additionally, interactive real-time visualizations are readily available on the web, and there are even mobile apps purely dedicated to visualizing sorting algorithms.

But despite the popularity of these visualizations and the research that has gone into the algorithms they represent, there seems to have been no serious mathematical analysis of the these visualizations themselves. Discussion of the shapes of they produce is confined to casual video comments and internet forums, where the question of the bubble sort curve has been proposed, but seemingly not given enough thought for a proper answer.

1.2 Bubble Sort

Bubble sort works by scanning across the list from left to right, comparing two adjacent items at each step. If at any step the left item is greater than the right item, the two items are swapped. This process is repeated until the entire list is sorted. The algorithm can be represented in pseudo-code as follows:

```
int N = list.length();
for (int i = 0; i < N - 1; i++) {
    for (int j = 0; j < N - 1; j++) {
        if (list[j] > list[j + 1]) {
            list.swapItems(j, j + 1);
        }
    }
}
```

This is an interpretation of the original algorithm [3], though it is the least efficient interpretation possible. Implementations of bubble sort typically include some common-sense optimizations which reduce redundant comparisons. But since we are only concerned with the shape that the diagrams make, we do not care about performance, and this simple version is the most sensible for our purposes.

Note that we use the convention of 0-indexes lists (The first item is in index 0). We will call each execution of the outer loop an *iteration*. We will often refer to a list *at I iterations*. When we do so, we are referring to the state of the list immediately after the Ith execution of the outer loop. In the code above, we run $N-1$ iterations. We will show that this is the minimum number of iterations required to guarantee that the list is sorted (Corollary 2). But rather than taking the most direct route to this fact, we will first introduce the notion of a *Greatest So Far (GSF)* item.

Definition 1. Consider a list at I iterations. If the item in index n is greater than or equal to the item in index m for all $m < n$, then we call the item in index n a *Greatest So Far (GSF)* item, and we all n a GSF index.

GSF indexes are important because the behavior of the algorithm during an iteration is completely determined by which items are GSF.

Theorem 1. *Given a list of N items, let $g_0, g_1, \cdots, g_k$ be all the GSF indexes, ordered from least to greatest. Over the next iteration,*

- *For all $m < k$, the item in index g_m will move to index $g_{m+1} - 1$.*
- *The item in index g_k will move to index $N - 1$.*
- *For every non-GSF index n, the item in index n will move to index $n-1$.*

Proof. We will track the variable `j` in the pseudo-code above. Whenever the item in index `j` is greater than its successor, a swap will occur and it will move to index `j+1`. But then the value of `j` will increment, so the item will again be in index `j` for the next execution of the inner loop. Therefore, by induction, this item will repeatedly move to the right until it directly precedes an item which is not less than it. It will not be swapped with this other item, so there it will rest.

Now, consider a GSF index g_m with $m < k$. Since the item in index g_m is greater than or equal to all the items before it, it will not be swapped with its predecessor when `j` $= g_m - 1$, so `j` will increment to g_m and our item will find itself in index `j`. As described above, it will then move to the right until it directly precedes the first item which is not less than it, which is the next GSF item by definition. Thus the item in index g_m moves to index $g_{m+1} - 1$.

For g_k, the same process occurs, but there is no next GSF item to stop the item from moving all the way to the end of the list and resting in the final index, $N - 1$.

Finally, every non-GSF item is passed by a GSF item on its way to its new location. When the two items are swapped, the GSF item moves to the right and the non-GSF item moves to the left, where it stays. □

Corollary 1. *After I iterations, the largest I items are sorted in their final positions, which are the largest I indexes of the list.*

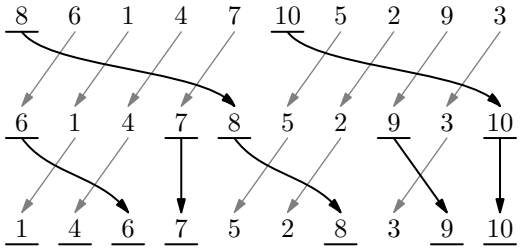

Figure 3: The movement of items during the first two iterations. At each iteration, the GSF items are underlined. This illustrates the behavior described in Theorem 1.

Proof. Note that, for any M, the largest item of the first M items is GSF, and it is the last GSF item of these first M items. We can now prove the corollary with a simple induction argument.

Assume that at I iterations, the largest I items are sorted in indexes $N - I, N-I+1, \ldots, N-1$. Clearly each of these indexes is GSF. Then, the $(I+1)$st largest item is the largest of the first $N-I$ items, which is in the last GSF index of these items. Therefore, by Theorem 1, over the next iteration the $(I+1)$st largest item moves to index $N-I-1$, and all the larger items do not move. Thus at $I+1$ iterations the, largest $I+1$ items are sorted in the $I+1$ largest indexes of the list.

For the base case, note that the largest item in the list is the last GSF item, so after the first iteration, it will be moved to index $N-1$ by Theorem 1. □

Corollary 2. *It takes $N-1$ iterations to guarantee that a list of N items is sorted.*

Proof. By Corollary 1, the largest $N-1$ items will be sorted at $N-1$ iterations, leaving only the smallest item. But then there is only one place left for the smallest item, so it cannot be in the wrong place, and the whole list must be sorted.

To prove that it takes no less than $N-1$ iterations, recall that an item can move no more than one space to the left per iteration. Therefore, if the smallest item starts in index $N-1$, it will take $N-1$ iterations to move it to its correct index of 0. □

Because of Theorem 1, we are able to analyze the behavior of algorithm purely mathematically, without ever again having to step through the code. Corollary 1 shows that at every point in time, the diagram is split into two distinct sections: the fully-sorted section on the right, and the unsorted section on the left. We will end up analyzing the shape of each of these sections separately.

1.3 Mathematically Describing Diagrams

The width, height, and duration of the videos are completely arbitrary, so we will normalize them. Our diagrams will be one unit wide, one unit tall, and will run for one unit of time. We will also stipulate that each iteration takes the same amount of time. To be specific, our diagrams will fit in the unit cube $[0,1]^3$. We will represent each point in the diagram with the coordinates x, y, and t, where t represents time.

Our specification that each iteration takes the same amount of time is only loosely accurate for the simplest and least optimized version of the algorithm. And even in that case, the number of time-consuming swaps is not the same for every iteration. However, this slight departure from reality is well worth the simplicity, and one can simply transform the time dimension of the final shape to account for more optimized versions of the algorithm. If you are still uncomfortable, you may interpret the t dimension as "progress", rather than "time", thereby sidestepping any issues of how long each iteration takes.

Although sorting algorithms can act on any type of data that can be ordered, we will restrict ourselves to lists of real numbers in the range $[0,1]$ so that each item in the list is a valid y-coordinate in the diagram. For the other two coordinates, we can spread the points evenly. The point corresponding to the item in index n at I iterations will have an x-coordinate of n/N and a t-coordinate of I/N. For simplicity, we will not plot the points at each step within an iteration, but little would change if we did.

It will be useful to define a function which gives the number in a certain index of the list at a certain iteration:

Definition 2. Given a list of N items, the height function of the list is

$$H(n,I) = \text{the number in index } n \text{ of the list at } I \text{ iterations},$$

where $n, I \in \{0,1,2,\ldots,N-1\}$.

We call this the height function because it gives the height of a point in the list's diagram. To be clear, this is not one single function. It only exists in the context of a list, and each list will have its own height function. The same is true for the diagram, which we will now define.

Definition 3. Given a list of N items, the diagram of the list is the set

$$D = \left\{ \left(\frac{n}{N}, H(n,I), \frac{I}{N} \right) : \ n, I \in \{0,1,2,\ldots,N-1\} \right\}.$$

Our goal is then to prove that if the lists are generated in a "uniform" way, then the set D will tend to "converge" to a certain connected set as the length of the list increases without bound.

1.4 Uniform Methods of Generating Lists

We have to specify how the lists are generated, because the method of generating the list could conceivably alter the shape that the diagram forms. In this section,

we will define a what it means to generate a list *uniformly*. We will then assume for the rest of the proof that the lists are generated in such a way.

To be specific, we define a "method of generating lists" as an infinite sequence of random processes – one for each natural number N – such that the N'th random process generates a list of N items. For example, one of the simplest methods of generating lists goes like this:

> To generate a list of N items, randomly choose N independent real numbers uniformly from $[0, 1]$.

We will call this the *real number method*. However, the most common way of generating lists for these visualizations is as follows:

> To generate a list of N items, list the natural numbers 1 through N. Shuffle the list randomly, then map each number n to n/N.

We will call this the *natural number method*.

There are infinitely other methods that we might concoct, and not all methods will yield the same shape. For example, the trivial method of setting every item to 0 will just form a horizontal line at $y = 0$, no matter what algorithm is run. To exclude such cases, we will only consider lists which are generated uniformly. Loosely speaking, a method of generating lists is uniform if the points are uniformly spread over the unit square. To phrase this more precisely, we will use the standard notion of convergence in probability.

Definition 4. A sequence of random variables $Z_0, Z_1, Z_2, \cdots$ converges in probability to a random variable Z if, for all $\varepsilon > 0$,

$$\lim_{n\to\infty} P(|Z - Z_n| < \varepsilon) = 1.$$

We express this by saying $Z_n \xrightarrow{p} Z$ as $n \to \infty$.

Definition 5. Consider a method of generating lists. Let R be an arbitrary rectangle contained in the unit square. Call the area of this rectangle A. Given a natural number N, let the random variable Z_N represent the number of points in the initial state of the list which are inside R. The method of generating lists is *uniform* if

$$\frac{Z_N}{N} \xrightarrow{p} A \quad \text{as} \quad N \to \infty$$

for every possible choice of R.

Convergence in probability is well-suited for the definition of uniformity. But for the arguments ahead, it will be more convenient to use a different shorthand to capture probabilities.

Definition 6. Let ϕ be a statement that makes sense in the context of a list. Given a method of generating lists, we define $P_N(\phi)$ to be the probability that ϕ is true of a list of N items generated with the method.

We will be particularly concerned with cases in which $\lim_{N\to\infty} P_N(\phi) = 1$ and in which $\lim_{N\to\infty} P_N(\phi) = 0$. We will say that such statements ϕ become arbitrarily likely or arbitrarily unlikely, respectively. If finitely many statements are arbitrarily likely, then so is their conjunction. Likewise, the disjunction of finitely many arbitrarily unlikely statements is arbitrarily unlikely.

We will rephrase our definition of uniformity a bit more precisely in terms of arbitrarily likely statements.

Definition 7 (Equivalent to Definition 5)**.** Given a list and four numbers x_0, x_1, y_0, y_1 with $0 \le x_0 < x_1 \le 1$ and $0 \le y_0 < y_1 \le 1$, let

$$Z = \left| \left\{ n \in \mathbb{N} : x_0 \le \frac{n}{N} \le x_1,\ y_0 \le H(n,0) \le y_1 \right\} \right|.$$

A method of generating lists is uniform if, for any such x_0, x_1, y_0, and y_1, and for any $\varepsilon > 0$,

$$\lim_{N\to\infty} P_N\left(\left| \frac{Z}{N} - (x_1 - x_0)(y_1 - y_0) \right| < \varepsilon \right) = 1.$$

The number Z in this definition is one particular instance of the random variable Z_N in Definition 5. Note that the boundary of the rectangle, having no area, does not factor into this definition. Therefore, our use of closed rectangles was arbitrary. This definition would have been equivalent had we chosen open or partially open rectangles, as we will in Section 3.

Indeed, the real number method and the natural number method are both uniform. We will prove this in section 4. Until then, we will simply assume we are dealing with uniform methods, and the specifics of the methods will not be relevant.

1.5 Mathematically Defining the Problem

The statement that bubble sort diagrams converge to some curved shape is vague, so we must translate it into a precise mathematical statement. In this section we develop a definition that captures this phenomenon, while assuming as little as possible about the nature of the specific sorting algorithm in question.

When we see the diagrams approaching the shape of some set S, we are loosely observing that the points in the diagram tend to "fill up" all of S, and that no points lie far outside of S. This can be captured more precisely with the following criteria: The diagrams approach the shape of S if, as $N \to \infty$,

- It becomes arbitrarily likely that every point in S becomes arbitrarily close to a point in the diagram, and
- It becomes arbitrarily *un*likely that there is a point *not* in S which is arbitrarily close to a point in the diagram.

This is essentially the definition we will use in this paper. However, a more precise definition will be useful:

Definition 8. Consider a uniform method of generating lists. A point $\mathbf{X}_0$ is *included* if for all $\varepsilon > 0$,

$$\lim_{N\to\infty} P_N(\text{There is a point } \mathbf{X} \in D \text{ such that } |\mathbf{X}_0 - \mathbf{X}| < \varepsilon) = 1.$$

A point $\mathbf{X}_0$ is *excluded* if there is an $\varepsilon > 0$ such that

$$\lim_{N\to\infty} P_N(\text{There is a point } \mathbf{X} \in D \text{ such that } |\mathbf{X}_0 - \mathbf{X}| < \varepsilon) = 0.$$

If every point in $[0,1]^3$ is either included or excluded, then we say that the bubble sort diagrams approach the set of included points.

At this moment it is not guaranteed that the bubble sort diagrams approach any shape at all, since the limits may not converge, or they may converge to values other than 0 or 1. Indeed, there are other sorting algorithms where this is the case. For example, if we were to instead analyze quicksort [4] we would find that most points in $[0,1]^3$ are neither included nor excluded. This is due to the fact that quicksort operates by selecting one of the list's random items as a pivot about which it splits the list, causing the diagram to have different proportions each time it is run. Fortunately bubble sort has no such issue, and since this paper is concerned with bubble sort, we will leave the topic of quicksort here.

There are a couple of simple theorems about included and excluded points which will be useful for our analysis, so we will prove them here.

Theorem 2. *The set of included points is closed.*

Proof. Let $\mathbf{X}_0$ be a limit point of the set of included points. For every $\varepsilon > 0$, there is an included point $\mathbf{X}_1$ with

$$|\mathbf{X}_1 - \mathbf{X}_0| < \varepsilon.$$

Since $\mathbf{X}_1$ is included, we can let $\varepsilon' = \varepsilon - |\mathbf{X}_1 - \mathbf{X}_0|$, and then

$$\lim_{N\to\infty} P_N(\text{There is a point } \mathbf{X} \in D \text{ such that } |\mathbf{X}_1 - \mathbf{X}| < \varepsilon') = 1.$$

But if $|\mathbf{X}_1 - \mathbf{X}| < \varepsilon'$, then

$$\begin{aligned} |\mathbf{X}_0 - \mathbf{X}| &\le |\mathbf{X}_0 - \mathbf{X_1}| + |\mathbf{X}_1 - \mathbf{X}| \\ &< |\mathbf{X}_0 - \mathbf{X_1}| + \varepsilon' \\ &= \varepsilon, \end{aligned}$$

So

$$\lim_{N\to\infty} P_N(\text{There is a point } \mathbf{X} \in D \text{ such that } |\mathbf{X}_0 - \mathbf{X}| < \varepsilon) = 1.$$

Therefore, every limit point of the set of included points is itself included. □

Theorem 3. *If S is an open set such that*

$$\lim_{N\to\infty} P_N(\mathbf{X}_0 \in S \textit{ for some } \mathbf{X}_0 \in D) = 0,$$

then every point in S is excluded.

Proof. Since S is open, for every $\mathbf{X}_0 \in S$ there is $\varepsilon > 0$ such that $\mathbf{X} \in S$ if $|\mathbf{X} - \mathbf{X}_0| < \varepsilon$. Therefore it becomes arbitrarily unlikely that there is a point $\mathbf{X} \in D$ such that $|\mathbf{X} - \mathbf{X}_0| < \varepsilon$. □

2 The Sorted Section: $x > 1 - t$

We will first prove that if $x > 1 - t$, then the point (x, y, t) is included if $x = y$, and it is excluded otherwise. We will also prove that if $x = 1 - t$ and $x > y$, then the point is excluded. This is not the main attraction – the shape of this section of the diagram is rather obvious. It is clear from Corollary 1 that this section of the diagram is sorted, and for a sorted list to approach anything other than a diagonal line would suggest that the list was not generated uniformly. However, since we have constructed a specific definition of what it means for the diagrams to approach a shape, we are bound to prove that our definition is satisfied.

To start, we will define a function which represents the sorted list.

Definition 9. Given a list of N items, the F function of the list is is given by

$$F(n) = \text{the item in index } n \text{ after the list is fully sorted,}$$

where $n \in \{0, 1, 2, \ldots, N-1\}$.

It is useful to restate Corollary 1 in terms of this function:

Theorem 4. *Consider a list of N items at I iterations. Let $n \geq N - I$. Then*

$$H(n, I) = F(n),$$

and if $m < n$, then

$$H(m, I) \leq F(n).$$

As stated above, it is obvious that the sorted list will look like the diagonal line $y = x$. The next lemma puts this precisely by proving that if the x-coordinate of a point on the sorted list is close to some value, then the y-coordinate will be close to that same value.

Lemma 1. *Let $0 \leq x \leq 1$ and consider a uniform method of generating lists. For all $\varepsilon > 0$,*

$$\lim_{N\to\infty} P_N\left(\left|\frac{n}{N} - x\right| < \frac{\varepsilon}{2} \implies |F(n) - x| \leq \varepsilon\right) = 1.$$

Proof. First, we will show that

$$\lim_{N\to\infty} P_N\left(\frac{n}{N} - x < \frac{\varepsilon}{2} \implies F(n) - x \leq \varepsilon\right) = 1. \tag{1}$$

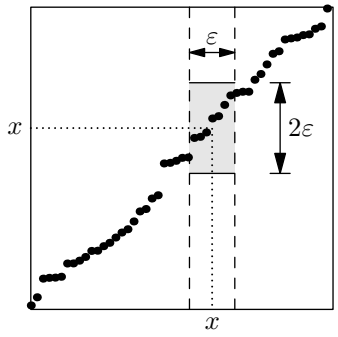

Figure 4: An illustration of Lemma 1. All the points $\left(\frac{n}{N}, F(N)\right)$ which are between the dashed lines are in the shaded region.

If $x + \varepsilon \geq 1$, then $F(N) \leq 1 \leq x + \varepsilon$, so (1) is trivially true. Therefore we will assume that $x + \varepsilon < 1$. Given a list of N items, let k be the number of items which are less than or equal to $x + \varepsilon$. Then

$$F(n) \leq x + \varepsilon \text{ for all } n < k. \tag{2}$$

But clearly, k is the number of points which are in the rectangle $[0, 1] \times [0, x + \varepsilon]$ in the initial state of the list. Therefore, by Definition 7,

$$\lim_{N \to \infty} P_N \left(\left| \frac{k}{N} - (x + \varepsilon) \right| < \frac{\varepsilon}{2} \right) = 1. \tag{3}$$

But $|k/N - (x + \varepsilon)| < \varepsilon/2$ implies that $k > N(x + \varepsilon/2)$, in which case $n < k$ for all $n < N(x + \varepsilon/2)$. Therefore, (2) and (3) imply that

$$\lim_{N \to \infty} P_N \left(n < N \left(x + \frac{\varepsilon}{2} \right) \implies F(n) \leq x + \varepsilon \right) = 1,$$

which is equivalent to (1).

Now we will use a symmetrical argument to show that

$$\lim_{N \to \infty} P_N \left(\frac{n}{N} - x > -\frac{\varepsilon}{2} \implies F(n) - x \geq -\varepsilon \right) = 1. \tag{4}$$

If $x - \varepsilon \leq 0$, then $F(N) \geq 0 \geq x - \varepsilon$, so (4) is trivially true. Therefore we will assume that $x - \varepsilon > 0$. Given a list of N items, let k be the number of items which are less $x - \varepsilon$. Then

$$f(n) \geq x - \varepsilon \text{ for all } n > k. \tag{5}$$

But clearly, k is the number of points which are in the rectangle $[0, 1] \times [0, x - \varepsilon]$ in the initial state of the list. Therefore, by Definition 7,

$$\lim_{N \to \infty} P_N \left(\left| \frac{k}{N} - (x - \varepsilon) \right| < \frac{\varepsilon}{2} \right) = 1. \tag{6}$$

But $|k/N - (x-\varepsilon)| < \varepsilon/2$ implies that $k < N(x-\varepsilon/2)$, in which case $n > k$ for all $n > N(x-\varepsilon/2)$. Therefore (5) and (6) imply that

$$\lim_{N\to\infty} P_N\left(n > N\left(x - \frac{\varepsilon}{2}\right) \implies F(n) \geq x - \varepsilon\right) = 1,$$

which is equivalent to (4).

Finally, (1) and (4) together imply that

$$\lim_{N\to\infty} P_N\left(-\frac{\varepsilon}{2} < \frac{n}{N} - x < \frac{\varepsilon}{2} \implies -\varepsilon \leq F(n) - x \leq \varepsilon\right) = 1,$$

which is equivalent to the claim of the lemma, so we are done. □

Theorem 5. *Consider a point $\mathbf{X}_0 = (x_0, y_0, t_0) \in [0,1]^3$ where $x_0 > 1 - t_0$. Then $\mathbf{X}_0$ is included if $x_0 = y_0$, and it is excluded if $x_0 \neq y_0$.*

Proof. Since the set of points (x,t) with $x > 1-t$ is open, we can choose $\varepsilon > 0$ to be small enough that every point in the set

$$S = \left\{(x,y,t) \in \mathbb{R}^3 : |x - x_0| < \frac{\varepsilon}{2},\ |t - t_0| < \frac{\varepsilon}{2}\right\}.$$

satisfies $x > 1 - t$. Given a uniformly generated list of N items, for every point $(x,y,t) \in D \cap S$, we have $x = n/N$, $t = I/N$, and $y = H(n,I)$, for some natural numbers n and I. Because $x > 1-t$, it must be that $n > N - I$, so Theorem 4 implies that $y = F(n)$. And since $|n/N - x_0| < \varepsilon/2$, we have by Lemma 1 that

$$\lim_{N\to\infty} P_N\left(|y - x_0| \leq \varepsilon \text{ for all } (x,y,t) \in D \cap S\right) = 1. \tag{7}$$

Therefore, if $x_0 \neq y_0$, we can let $\varepsilon < |y_0 - x_0|$, so that $\{(x,y,t) \in S : |y - x_0| > \varepsilon\}$ is an open set containing $\mathbf{X}_0$ that becomes arbitrarily unlikely to contain a point from D as $N \to \infty$. Therefore $\mathbf{X}_0$ is excluded by Theorem 3.

On the other hand, if $x_0 = y_0$, then (7) implies that it becomes arbitrarily likely that each point $\mathbf{X} \in D \cap S$ satisfies

$$|\mathbf{X} - \mathbf{X}_0| = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (t-t_0)^2} < \sqrt{\left[\frac{\varepsilon}{2}\right]^2 + \varepsilon^2 + \left[\frac{\varepsilon}{2}\right]^2} = \frac{\sqrt{6}}{2}\varepsilon.$$

Since for any ε, the set $D \cap S$ is not empty when N is large enough, and since ε can be made arbitrarily small, $\mathbf{X}_0$ is included. □

We have just proved the inclusion and exclusion of points with $x > 1-t$, and in Section 3 we will prove the inclusion and exclusion of points with $x < 1-t$. This leaves us with $x = 1-t$. Of these points, the ones with $y \leq x$ will be will also be covered in Section 3, so we still need to consider those points with $y > x$.

Theorem 6. *If $x_0 = 1 - t_0$ and $y_0 > x_0$, then the point $\mathbf{X}_0 = (x_0, y_0, t_0)$ is excluded.*

Proof. Since the set of points (x, y) satisfying $y > x$ is open, there is a positive ε small enough that the point

$$(x_1, y_1) = (x_0 + \varepsilon, y_0 - \varepsilon)$$

satisfies $y_1 > x_1$. Then $\mathbf{X}_0$ is contained in the set

$$S = \{(x, y, t) : x < x_1, y > y_1, t > 1 - x_1\}.$$

Consider a list of N items. A point from D will be in S if and only if there is a pair of natural numbers n, I such that

$$n < Nx_1, \quad I > N(1 - x_1), \quad \text{and} \quad H(n, I) > y_1. \tag{8}$$

We will show that this becomes arbitrarily unlikely as $N \to \infty$.

Let k be the number of items that are greater than or equal to y_1. Assume, for now, that $k < N(1 - x_1)$. If $I > N(1 - x_1)$, then $I > k$, so by Corollary 1, all of the k items which are greater than or equal to y_1 are in indexes $N - k$ through $N - 1$. Thus, $H(n, I) < y_1$ for all $n < N - k$. This means that, since $Nx_1 < N - k$, the first two conditions of (8) imply that $H(n, I) < y_1$, so it is impossible for all three conditions of (8) to be satisfied. Therefore, since we started with the assumption that $k < N(1 - x_1)$, we have shown that

$$k < N(1 - x_1) \implies \mathbf{X} \notin S \text{ for all } \mathbf{X} \in D. \tag{9}$$

Now, k is clearly the number of points which start in the rectangle $[0, 1] \times [y_1, 1]$. Therefore, by Definition 7,

$$\lim_{N\to\infty} P_N\left(\left|\frac{k}{N} - (1 - y_1)\right| < y_1 - x_1\right) = 1,$$

which implies that

$$\lim_{N\to\infty} P_N(k < N(1 - x_1)) = 1.$$

This combined with (9) implies that

$$\lim_{N\to\infty} P_N(\mathbf{X} \notin S \text{ for all } \mathbf{X} \in D) = 1,$$

or equivalently,

$$\lim_{N\to\infty} P_N(\mathbf{X} \in S \text{ for some } \mathbf{X} \in D) = 0.$$

Therefore, S is an open set containing $\mathbf{X}_0$ which becomes arbitrarily unlikely to include a point from D, so $\mathbf{X}_0$ is excluded. □

3 The Unsorted Section: $x < 1 - t$

We now move on to the main problem: proving that the bubble sort diagrams approach the curved region when $x < 1 - t$. To do this, we will modify the

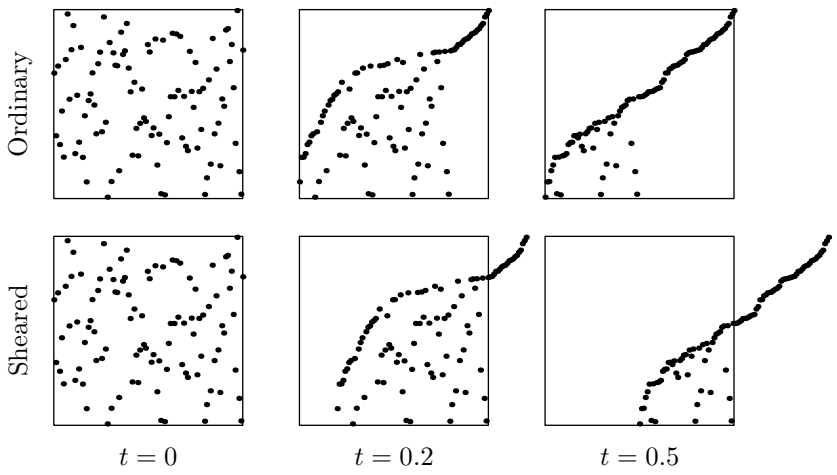

Figure 5: t-cross sections of the ordinary and sheared diagrams of a list, plotted on the unit square.

diagram slightly by shearing it parallel to the t-axis. Then we will superimpose rectangles onto this sheared diagram, and we will find that the number of points which lie inside a rectangle can easily be tracked as the algorithm runs. By choosing specific rectangles and noting at which moments they have points in them, we will be able to determine which points are included and which are excluded.

3.1 The Sheared Diagram

Theorem 1 gives a convenient description of how the list changes over the course of an iteration. However, there is one inconvenient aspect about this behavior: it has a leftward bias. Each GSF item moves one space to the left of the next GSF item, and every non-GSF item moves one space to the left of itself. It would be simpler if each GSF item moved directly to the position of the next GSF item, and if each non-GSF item stayed put. To make this happen, we just need to shift the whole list one space to the right after every iteration.

When we shift the entire list in this way, the effect on the three-dimensional diagram is a shear transformation parallel to the t-axis. We will therefore describe these shifted lists with the word *shear*.

Definition 10. A *sheared list* of N items is a copy of a list of N items which is shifted forward by the number of iterations that have been performed. That is, at I iterations, the sheared list consists of the indexes I through $I + N - 1$, and the item in index n of the original list is in index $n + I$ of the sheared list.

Definition 11. Given a sheared list at I iterations, the item in index n of the sheared list is GSF if the item in index $n - I$ of the original list is GSF. In this case, we will call n a GSF index of the sheared list at I iterations.

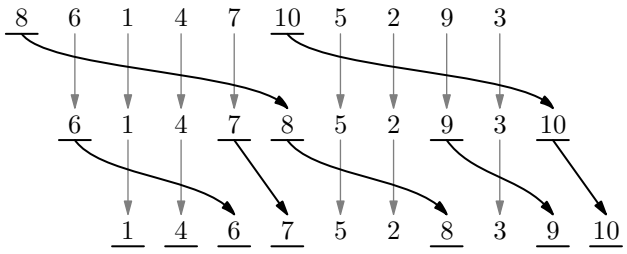

Figure 6: A repeat of Figure 3, but with a sheared list rather than an ordinary list. Note that non-GSF items do not move, as stated in Theorem 7.

Though we use the same acronym for GSF items/indexes in the original list and the sheared list, our meaning will be clear from context. If we begin with, "given a sheared list," then any mention of GSF items/indexes will be referring to the GSF items of the sheared list, rather than the original list. We will now restate state Theorem 1 in terms of sheared lists.

Theorem 7. *Given a sheared list of N items at I iterations, let $g_0, g_1, \cdots, g_k$ be all the GSF indexes, ordered from least to greatest. After the next iteration,*

- *For all $m < k$, the item in index g_m will move to index g_{m+1}.*
- *The item in index g_k will move to index $N + I$.*
- *Every non-GSF item will not move.*

Definition 12. The *sheared diagram* of a list is the set

$$D_s = \{(x + t, y, t) : (x, y, t) \in D\},$$

or, equivalently,

$$D_s = \left\{ \left(\frac{n}{N}, H(n - I, I), \frac{I}{N} \right) : I, (n - I) \in \{0, 1, 2, \ldots, N - 1\} \right\}.$$

Figure 5 hints at why the sheared diagram is better to work with than the ordinary diagram: The points in the bottom left corner of the unit square remain completely stationary as the algorithm runs. And though the right side of the sheared diagram escapes from the unit square, there is no need to worry. The points which lie outside the square are those for which $x > 1 - t$ in the original diagram, and we already dealt with them in Section 2.

Definition 13. Consider a uniform method of generating lists. A point $\mathbf{X}_0$ is *shear-included* if for all $\varepsilon > 0$,

$$\lim_{N \to \infty} P_N(\text{There is a point } \mathbf{X} \in D_s \text{ such that } |\mathbf{X}_0 - \mathbf{X}| < \varepsilon) = 1.$$

A point $\mathbf{X}_0$ is *shear-excluded* if there is an $\varepsilon > 0$ such that

$$\lim_{N\to\infty} P_N(\text{There is a point } \mathbf{X} \in D_s \text{ such that } |\mathbf{X}_0 - \mathbf{X}| < \varepsilon) = 0.$$

Since the mapping from the ordinary diagram to the sheared diagram is continuous and invertible, small neighborhoods are mapped to small neighborhoods. Therefore it is easy to see that the inclusion and exclusion of points is preserved by this transformation.

Theorem 8. *A point (x, y, t) is included if and only if $(x + t, y, t)$ is shear-included. Likewise, a point (x, y, t) is excluded if and only if $(x + t, y, t)$ is shear-excluded.*

3.2 Counting Points in Boxes

We will analyze the sheared diagram by drawing rectangles over it and counting the number of items in the rectangles at each iteration. It will be convenient to only consider rectangles which are closed on the top and left, but open on the bottom and right. For conciseness, we will call such a rectangle a box. We will refer to the number of points inside a box as the box's *value*.

Definition 14. A *box* is a set of the form

$$B = [x_0, x_1) \times (y_0, y_1],$$

where $0 \le x_0 < x_1 \le 1$ and $0 \le y_0 < y_1 \le 1$.

Definition 15. Given a sheared list of N items, the *value* of a box $B = [x_0, x_1) \times (y_0, y_1]$ at I iterations is

$$V(B, I) = \left| \left\{ (x, y, t) \in D_s : (x, y) \in B,\ t = \frac{I}{N} \right\} \right|,$$

or, equivalently,

$$V(B, I) = \left| \left\{ n \in \mathbb{N} : x_0 \le \frac{n}{N} < x_1,\ y_0 < H(n - I, I) \le y_1 \right\} \right|.$$

If $V(R, I) = 0$, then we say that B is *empty* at I iterations.

It will be useful to describe the position of points relative to boxes. To do this, we will use the cardinal directions north, west, and northwest.

Definition 16. Consider a box $B = [x_0, x_1) \times (y_0, y_1]$. Let n be the index of an item in a sheared list of N items at I iterations.

- The item is north of B if $x_0 \le \frac{n}{N} < x_1$ and $H(n - I, I) > y_1$.
- The item is west of B if $\frac{n}{N} < x_0$ and $y_0 < H(n - I, I) \le y_1$.
- The item is northwest of B if $\frac{n}{N} < x_0$ and $H(n - I, I) > y_1$.

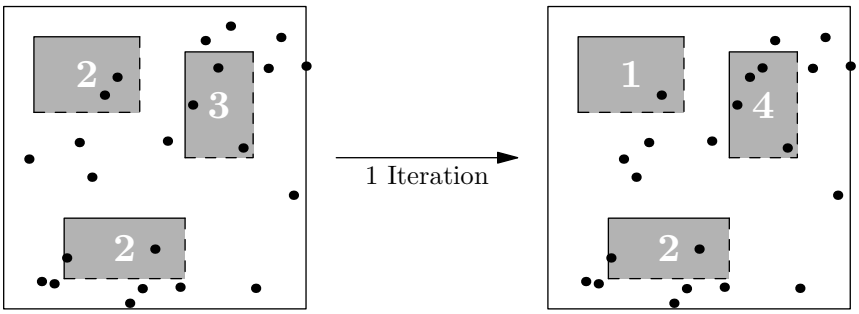

Figure 7: Three boxes on a sheared diagram before and after an iteration. The changes in the values of boxes demonstrate the three cases of Theorem 9.

We neglect east, south, and every other combination of cardinal directions because we do not need them. This next theorem states that the change in a box's value depends only on whether or not there are items north, west, or northwest of the box. This will be the key to proving that the diagrams approach a shape because it will allow us to forget the exact state of the list, and to instead only worry about the values of certain boxes.

Theorem 9. *Let B be a box, and consider a sheared list at some number of iterations. Over the course of the next iteration,*

- *the value of B will decrease by 1 if B is not empty and there are no items north, west, or northwest of B,*
- *the value of B will increase by 1 if there are items both north and west, but not northwest, of B.*
- *Otherwise, the value of B will not change.*

Proof. Let the list be of N items, and let it be at I iterations. We will say that any items whose index n satisfies $x_0 \leq \frac{n}{N} < x_1$ is *in the column of B*, and we will say that any item whose index n satisfies $y_0 < H(n-I, I) \leq y_1$ is *in the row of B*. An item is in B if and only if it is in both the row and the column of B.

Since only GSF items move, and they all move to the position of the next GSF item, in a single iteration it is only possible for one item to leave the column of B, and it is only possible for one item to enter the column of B. In particular,

- The only item which leaves the column of B is the largest GSF item in the column of B. We will call this the *leaving item*. If this item exists, it will always leave the column of B, because it moves to the position of the next GSF item (if there is one) which is outside the column of B by

definition, or it moves to index $N + I$, which puts it outside of the column of every possible box.

- The only item which can enter the column B is the largest GSF item whose index is less than Nx_0. We will call this the *entering item*. It will enter the column of B if and only if the leaving item exists, because otherwise there would be no GSF items in the column of B. This would imply that the entering item passes through B and arrives at the position of the next GSF item, or the end of the list if there is no such item, which puts it outside the column of B.

Since items only move horizontally, the entering item only enters B if it is in the row of B, and the leaving item only leaves B if it is in the row of B.

If there is an item northwest of B, then there must be at least one GSF item northwest of B. This means that both the entering item and the leaving item item – if it exists – are outside the row of B, as they are too high. Therefore, no items enter nor leave B, and the value of B does not change.

If there is an item north of B, but there are no items northwest of B, then there must be a GSF item north of B. This means that the leaving item exists, but is too high to be in the row of B. Therefore, no item leaves B.

- In this case, if there is no item west of B, then the entering item cannot be in the row of B, if it exists at all. Therefore, no item enters B, so the value of B does not change.

- On the other hand, if there is an item west of B, then the entering item must exist, and it must be in the row of B. Since the leaving item exists, the entering item enters B. Thus the value of B increases by 1.

If there are no items north or northwest of B, but there is an item west of B, then the entering item is in the row of B. If there are no GSF items in B, then nothing enters or leaves B, so the value of B does not change. If there is at least one GSF item in B, then the leaving item exists, and it is in the row of B, since it cannot be north of B. Thus, an item enters B and an item leaves B, so the value of B does not change.

Finally, if there are no items north, west, or northwest of B, then the entering item, if it exists, is too low to be in the row of B, so no items enter B. If B is not empty, then there is a GSF item in B, so the leaving item exists, so the value of B decreases by 1.

We have now covered every possible case and shown that the behavior is as described in the statement of the theorem, so the proof is complete. □

This theorem has an intriguing consequence. We can partition the unit square into a grid of boxes, and write down the value of each box in the initial state of the list. Then we can forget the actual state of the list, and we will still be able to perfectly track the values of the boxes as the algorithm progresses! All we have to do is look at each box and note whether there are nonempty

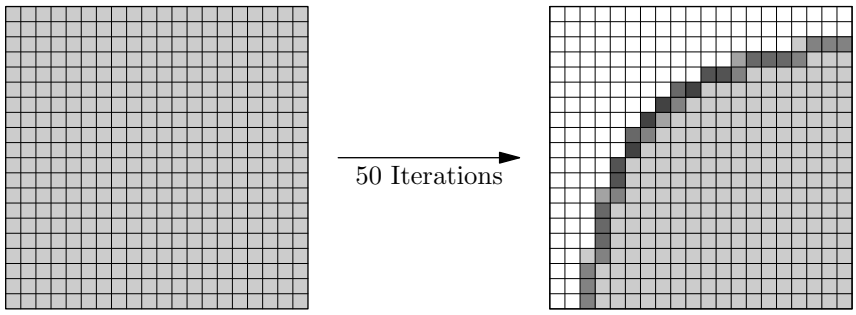

Figure 8: A 20×20 grid of boxes, each with an initial value of 1, before and after 50 iterations are applied. The value of the box is represented by shading.

boxes north, west, and northwest of it. Applying the rules of Theorem 9 to each box gives the state of the grid in the next iteration.

If we "pixelate" the diagram with a grid of equally sized boxes and assume that each box starts with the same value, then a very clear shape emerges. This is shown in Figure 8. Unlike the raw bubble sort diagram which is random and uneven, this pixelated representation is perfectly symmetrical and ordered. The hope of finding such simple rules which yield such a clear shape was what motivated the idea of counting points in boxes in the first place. However, we do not need the entire grid to prove which points are included and excluded. We just need a couple results about boxes, which we will now prove.

Corollary 3. *Let B be the box $[0, x_1) \times (y_0, 1]$. Given a sheared list,*

$$V(B, I) = \begin{cases} V(B, 0) - I & I < V(B, 0) \\ 0 & I \geq V(B, 0) \end{cases}.$$

Proof. Since there cannot be any points north, west, or northwest of B, the value of B will decrease by 1 for each iteration, until it is empty. □

Corollary 4. *Consider a sheared list. Let $B = [x_0, x_1) \times (y_0, y_1]$ and let $B' = [0, x_1) \times (y_0, 1]$. If $V(B, 0) > 0$, then*

$$V(B, I) > 0 \quad \textit{for all} \quad I < V(B', 0).$$

Proof. Corollary 3 applies to the boxes $[0, x_0) \times (y_0, 1]$ and $[0, x_1) \times (y_1, 1]$, whose union is $B' \backslash B$. Thus, once $B' \backslash B$ is empty, it will remain empty forever. Note that $B' \backslash B$ consists of all the points north, west, and northwest of B. So Theorem 9 implies that the value of B cannot decrease until $B' \backslash B$ is empty. It is thus impossible for B to be empty unless $B' \backslash B$, and therefore all of B', is empty. But by Corollary 3, B' is not empty for any $I < V(B', 0)$, so neither is B. □

3.3 Finding Shear-Included and Shear-Excluded points

Now we are ready to find which points are shear-included and shear-excluded. We will start with excluded points.

Theorem 10. *Let* $\mathbf{X}_0 = (x_0, y_0, t_0) \in [0,1) \times (0,1] \times (0,1]$. *If* $x_0(1-y_0) < t_0$, *then* $\mathbf{X}_0$ *is shear-excluded.*

Proof. Since the set of points (x, y, t) satisfying $x(1-y) < t$ is open, there is an $\varepsilon > 0$ which is small enough that

$$(x_1, y_1, t_1) = (x_0 + \varepsilon, y_0 - \varepsilon, t_0 - \varepsilon)$$

satisfies $x_1(1-y_1) < t_1$. Furthermore, we can let ε be small enough that $x_1 < 1$, $y_1 > 0$, and $t_1 > 0$. Let

$$S = \{(x, y, t) : x < x_1,\ y > y_1,\ t > t_1\}$$

and note that $\mathbf{X}_0 \in S$. Given a uniformly generated sheared list of N items, a point from the sheared diagram is in S only if there is a pair of natural numbers n, I with

$$\frac{n}{N} < x_1, \quad H(n-I, I) > y_1, \quad \text{and} \quad \frac{I}{N} > t_1. \tag{10}$$

We will show that this becomes arbitrarily unlikely as $N \to \infty$.

Consider the box $B = [0, x_1) \times (y_1, 1]$. Since the sheared list is generated uniformly,

$$\lim_{N\to\infty} P_N\left(\left|\frac{V(B,0)}{N} - x_1(1-y_1)\right| < t_1 - x_1(1-y_1)\right) = 1,$$

which implies that

$$\lim_{N\to\infty} P_N\left(V(B,0) < Nt_1\right) = 1. \tag{11}$$

But if we assume that $V(B,0) < Nt_1$, then Corollary 3 implies that B is empty for all $I > Nt_1$. This means that $H(n-I, I) \le y_1$ for all $n < Nx_1$. Therefore it is impossible to satisfy all three conditions of (10). Thus (11) implies that

$$\lim_{N\to\infty} P_N\left(\mathbf{X} \notin S \text{ for all } \mathbf{X} \in D_s\right) = 1.$$

Therefore, S is an open set containing $\mathbf{X}_0$ which becomes arbitrarily unlikely to contain any points from the diagram as $N \to \infty$, so $\mathbf{X}_0$ is shear-excluded by Theorem 3. □

Theorem 11. *Let* $\mathbf{X}_0 = (x_0, y_0, t_0) \in [0,1) \times (0,1] \times (0,1]$. *If* $x_0(1-y_0) > t_0$, *then* $\mathbf{X}_0$ *is shear-included.*

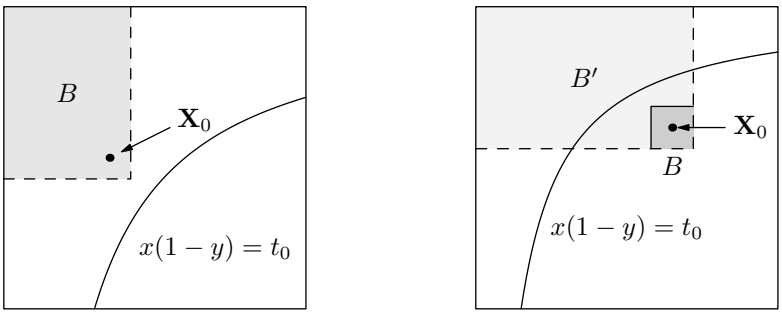

Figure 9: A 2D representation of the box used in the proof of Theorem 10 (left), and the boxes used in the proof of Theorem 11 (right).

Proof. Since the set of points (x, y, t) satisfying $x(1-y) > t$ is open, there is an $\varepsilon > 0$ small enough that

$$(x_1, y_1, t_1) = (x_0 - \varepsilon, y_0 + \varepsilon, t_0 + \varepsilon)$$

satisfies $x_1(1-y_1) > t_1$. We can also let

$$(x_2, y_2, t_2) = (x_0 + \varepsilon, y_0 - \varepsilon, t_0 - \varepsilon),$$

and we can choose ε to be small enough that $x_2 < 1$, $y_1 > 0$, and $t_0 > 0$. Let

$$S = [x_1, x_2) \times (y_2, y_1] \times (t_2, t_1)$$

and let

$$B = [x_1, x_2) \times (y_2, y_1].$$

A point from the sheared diagram is in S if $V(B, I) > 0$ for some I satisfying $t_1 < \frac{I}{N} < t_2$. We will show that this becomes arbitrarily likely as $N \to \infty$.

Let $B' = [0, x_2) \times (y_2, 1]$. If the sheared list is generated uniformly, then

$$\lim_{N\to\infty} P_N\left(\left|\frac{V(B', 0)}{N} - x_2(1-y_2)\right| < x_2(1-y_2) - t_2\right) = 1,$$

which implies that

$$\lim_{N\to\infty} P_N\left(V(B', 0) > Nt_2\right) = 1. \tag{12}$$

If we assume that $V(B', 0) > Nt_2$ and $V(B, 0) > 0$, then Corollary 4 implies that $V(B, I) > 0$ for all $I < Nt_2$. But since B has nonzero area, it becomes arbitrarily likely that $V(B, 0) > 0$ as $N \to \infty$. This and (12), combined with Corollary 4, imply that

$$\lim_{N\to\infty} P_N\left(V(B, I) > 0 \text{ for all } I < Nt_0\right) = 1,$$

which in turn implies that

$$\lim_{N\to\infty} P_N\left(\mathbf{X} \in S \text{ for some } \mathbf{X} \in D_s\right) = 1.$$

Since for any $X \in S$, we have

$$|\mathbf{X} - \mathbf{X}_0| < \sqrt{\varepsilon^2 + \varepsilon^2 + \varepsilon^2} = \varepsilon\sqrt{3},$$

and since ε can be made arbitrarily small, it follows that $\mathbf{X}_0$ is shear-included. □

3.4 Bringing it All Together

Theorem 12. *The shape of bubble sort is the set*

$$\left\{(x, y, t) \in [0,1]^3 : y \leq \frac{x}{x+t},\ x \leq 1 - t\right\} \cup \left\{(x, x, t) \in [0,1]^3 : x > 1 - t\right\}.$$

Proof. Recall that the shape of bubble sort exists if every point in $[0,1]^3$ is either included or excluded, in which case it is the set of included points. Theorem 5 shows that every point $(x, y, t) \in [0,1]^3$ with $x > 1 - t$ is either included or excluded, and that the included points are those in the second set in the statement of the theorem. This leaves only the points where $x \leq 1 - t$.

Theorem 11 states that a point is shear-included if $x(1 - y) > t$, $0 \leq x < 1$, $0 < y \leq 1$, and $0 < t \leq 1$. This, along with Theorem 8, implies that a point is included if

$$(x + t)(1 - y) > t, \quad 0 \leq x + t < 1, \quad 0 < y \leq 1, \quad 0 < t \leq 1.$$

But from Theorem 2, the set of included points is closed. Therefore, a point is included if

$$(x + t)(1 - y) \geq t, \quad 0 \leq x + t \leq 1, \quad 0 \leq y \leq 1, \quad 0 \leq t \leq 1,$$

which is satisfied if

$$y \leq \frac{x}{x+t}, \quad x \leq 1 - t, \quad (x, y, t) \in [0,1]^3.$$

These are precisely the points of the first set in the statement of the theorem.

Similarly, Theorem 10 states that a point is shear-excluded if $x(1 - y) < t$, $0 \leq x < 1$, $0 < y \leq 1$, and $0 < t \leq 1$. This, along with Theorem 8, implies that a point is excluded if

$$(x + t)(1 - y) < t, \quad 0 \leq x + t < 1, \quad 0 < y \leq 1, \quad 0 < t \leq 1.$$

But if $y = 0$ or $t = 0$, then it is impossible for $(x + t)(1 - y)$ to be less than t. Therefore, a point is excluded if

$$(x + t)(1 - y) < t, \quad 0 \leq x + t < 1, \quad 0 \leq y \leq 1, \quad 0 \leq t \leq 1,$$

which is satisfied if

$$y > \frac{x}{x+t}, \quad x < 1 - t, \quad (x, y, t) \in [0,1]^3.$$

We have shown that every point in the unit cube with $x > 1-t$ or $x < 1-t$ is either included or excluded. The only points remaining are those for which $x = 1-t$ and $y > x/(x+t)$. But these are precisely the points which Theorem 6 proves are excluded. Therefore, we have shown that every point in the unit cube is either included or excluded, and that the included points are those described in the statement of the theorem, so we are done. □

4 Proving the Uniformity of Common Methods

We have finally proven that, if the method of generating the lists is uniform, then the diagrams will indeed approach a certain shape. What remains is to show that the common methods of generating lists are uniform. To do this, we will use some standard results in probability theory. First, we will use a direct result of the weak law of large numbers.

Theorem 13 (The Weak Law of Large Numbers)**.** *Let $\overline{Z}_n$ be the sample mean of n independent and identically distributed random variables with expected value μ. Then*

$$\overline{Z}_n \xrightarrow{p} \mu \quad as \quad n \to \infty.$$

Corollary 5. *Consider an experiment which succeeds with probability p. Let Z_n be the number of successes of n independent trials. Then*

$$\frac{Z_n}{n} \xrightarrow{p} p \quad as \quad p \to \infty.$$

Second, we will use the mean and variance of the hypergeometric distribution. A derivation of these values can be found in [2].

Theorem 14. *Consider a collection of N items, K of which have some desired property. Randomly select n of these items (without replacement), and let Z denote the number of the selected items which have the desired property. The random variable Z has the* hypergeometric distribution*, whose mean and variance are given by*

$$E[Z] = \frac{nK}{N} \quad and \quad \mathrm{Var}(Z) = \frac{nK(N-n)(N-K)}{N^2(N-1)}.$$

Now, we can use the weak law of large numbers to prove that the real number method is uniform.

Theorem 15. *The real number method is uniform.*

Proof. Consider a rectangle $X \times Y$, where X and Y are nonempty intervals within $[0, 1]$. Let w and h be the lengths of X and Y, respectively. Given a natural number N, define W_N to be the number of nonnegative integers n for which $n/N \in X$. Clearly

$$\frac{W_N}{N} \to w \quad \text{as} \quad N \to \infty. \tag{13}$$

For each one of these W_N integers, say n, a real number is chosen uniformly from $[0, 1]$ to be in index n of the list. The resulting point is within the rectangle if and only if the chosen real number is in Y, which occurs with probability h. We can label such a case as a "success".

Therefore, the number of points in the rectangle, which we will write as Z_N, is the number of successes of W_N independent trials. Thus, by Corollay 5,

$$\frac{Z_N}{W_N} \xrightarrow{p} h \quad \text{as} \quad W_N \to \infty.$$

Since $W_N \to \infty$ as $N \to \infty$, this and (13) imply that

$$\frac{Z_N}{N} = \frac{W_N}{N} \cdot \frac{Z_N}{W_N} \xrightarrow{p} wh \quad \text{as} \quad N \to \infty,$$

so Definition 5 is satisfied. This proves that the real number method is uniform. □

Theorem 16. *The natural number method is uniform.*

Proof. Recall that the natural number method is the process of choosing a random permutation of the numbers 1 through N, and for each number plotting the point $\left(\frac{n}{N}, \frac{m}{N}\right)$, where m is the number and n is the index of the number. Clearly there are N possible x-coordinates and N possible y-coordinates for points, both of which are evenly spaced across the unit square.

Consider a nondegenerate rectangle R of width w and height h contained in the unit square. We will say an item is in the column of R if its point is in, above, or below R, and we will say an item is in the row of R if it is in, to the left of, or to the right of R. Given a number N, let W and H be the numbers of items which are in the column or the row of R, respectively. Clearly

$$\lim_{N\to\infty} \frac{W}{N} = w \quad \text{and} \quad \lim_{N\to\infty} \frac{H}{N} = h. \tag{14}$$

Define Z to be the random variable representing the number of points in R. When we shuffle the list and count the points in R, we can think of ourselves as randomly choosing W items to be in the column of R and counting how many of them have the property of being in row of R. This is the process of sampling without replacement. Therefore, by Theorem 14, the mean and variance of Z are given by

$$E[Z] = \frac{WH}{N} \quad \text{and} \quad \text{Var}(Z) = \frac{WH(N-W)(N-H)}{N^2(N-1)}.$$

By applying (14) to the mean we see that

$$\lim_{N\to\infty} E\left[\frac{Z}{N}\right] = \lim_{N\to\infty} \frac{WH}{N^2} = wh, \tag{15}$$

and by applying (14) to the variance we see that

$$\lim_{N\to\infty} \operatorname{Var}\left(\frac{Z}{N}\right) = \lim_{N\to\infty} \frac{WH(N-W)(N-H)}{N^4(N-1)} = 0.$$

That the variance converges to 0 is sufficient for the variable to converge in probability to the limit of the mean. Therefore

$$\frac{Z}{N} \xrightarrow{p} wh \quad \text{as} \quad N \to \infty.$$

Thus, since the choice of rectangle was arbitrary, the method is uniform. □

5 Future Considerations

Since the shapes of sorting algorithm visualizations have not been rigorously studied before, the entire world of shape-of-visualization proofs is open for exploration. Our definition of what it means for a diagram to approach a shape can easily be modified to work with many sorting algorithms. We chose bubble sort because its exact shape is not evident from an understanding of the algorithm. However, the shapes of most well-known sorting algorithms are glaringly obvious, and in those cases it is hard to blame one for refusing to trudge through such a technical proof of such a self-evident result.

But there is more to the appearance of sorting algorithm visualizations than the overall shape, and our method does not capture all the interesting details. In particular, we identified the shape of the diagram as a set that is "filled up" by the diagram's points, but our method has nothing to say about the density of the points. It is apparent from Figure 2 that the curved border of bubble sort's shape is far more densely-packed its body, a fact which our definition is blind to. For bubble sort we do not consider this to be important important since its allure lies in its overall shape, but for other sorting algorithms the density may be the more interesting aspect. For instance, at first glance heapsort's [6] scatter plot appears to form a gradient in density, so a more sophisticated method will be needed to understand it.

References

[1] Ronald M. Baecker. *Sorting Out Sorting*. 1980.

[2] J. R. Baxter. *Introduction to Probability*. John R. Baxter, 2023.

[3] Edward Friend. "Sorting on Electronic Computer Systems". In: *Journal of the ACM* 3.3 (1956), pp. 134–168. DOI: https://doi.org/10.1145/320831.320833.

[4] C. A. R. Hoare. "Quicksort". In: *The Computer Journal* 5.1 (Jan. 1962), pp. 10–16. DOI: https://doi.org/10.1093/comjnl/5.1.10.

[5] D. E. Knuth. “Sorting by Exchanging”. In: *The Art of Computer Programming Volume 3: Sorting and searching*. 2nd ed. Vol. 3. Addison-Wesley, 1998, pp. 105–110.

[6] J. W. J. Williams. “Algorithm 232: Heapsort”. In: *Communications of the ACM* 7.6 (1964), pp. 347–348.