Part III
Algorithm and Complexity of Folding

In Part III, we turn to a simple origami model away from the solid. When it comes to origami, it is a molding that folds two-dimensional squares with various folding lines. In recent complex origami, extremely fine and accurate folding such as pleat folding in 128 equal parts is required. When making such equally spaced creases, you notice that the number of folds varies depending on the way of folding. Specifically, if you fold paper in two or more sheets once, you can save the number of folds. On the other hand, if you fold it in a lot of sheets at the same time, it will be disturbed by the thickness of the paper, and you will not be able to fold with accuracy.

Useful frameworks for thinking about such "how to fold" are " *algorithm* " and " *computational complexity* ". They are the frameworks for discussing the skill of "method for computation" that comes out in computer science. The idea of applying these frameworks to origami is a recent trend in the field of computational geometry.

First, algorithm is the method of computation. Although it seems inseparable with computer programs, in fact, the history of the algorithm itself is much older than computers. "Euclidean algorithm" is a procedural method for finding the greatest common divisor of two natural numbers.[1] This is said to be the oldest algorithm which is known from about 300 BC. Since the real computer was invented in the 1930s, it can be said that the idea of algorithm itself is over than 2000 years older than computers.

On the other hand, computational complexity is the cost necessary for computation. It is used when quantifying the cost needed to compute and discussing goodness or badness of algorithms. In the case of computers, there are *time complexity* to evaluate computation time and *space complexity* to evaluate the amount of memory required for computation. Generally, it is known that time and space are in a trade-off relationship with algorithms on computers. In other words, generally, fast algorithms tend to consume a lot of memory and save computation speed when saving memory. Of course, developing a clever algorithm sometimes makes it possible to create a high-speed program with reduced memory. When introducing this approach into

[1]Euclidean algorithm is a computation procedure (.=.algorithm) for finding the greatest common divisor for two given natural numbers p and q (for the sake of simplicity, we assume $p > q$). Let r be the remainder obtained by dividing p by q. If r is 0, then q is the greatest common divisor. If r is not 0, repeat the same operation regarding as q and r are new p and q. If we repeat this until r becomes 0, then the last q is the greatest common divisor of the original p and q.

origami, we can discuss the computational complexity of origami. For example, even with the same origami model, if they had to fold 500 times or 50 times, most people would like the latter. On the other hand, folding a lot of paper will be difficult if the number of sheets increases, and precision will also deteriorate, so you would want to avoid this situation. In order to discuss such an intuitive feeling more quantitatively and theoretically, research on origami algorithm and computational complexity has started.

In Part III, we will learn the latest results of "computational complexity" and "algorithm" of computational origami, which is a new academic field.

In any framework, in order to consider algorithms and computational complexity, it is necessary to properly define the "model" as the basis of the discussion and the "basic operations" allowed on the model. In other words, to discuss the quality of algorithms, it is meaningless unless considering the same set of basic operations on a common model. Such a model and the basic operations on the model must be valid and persuasive in the origami field. In the origami society, basic operations called "Huzita's six axioms" and "Hatori's seventh axiom" have already well known, and this framework is quite stable. (Since these basic operations are not handled directly in this book, concrete axioms are omitted here. Interested readers should refer to the reference [DO07].) Therefore, in considering general origami algorithms and computational complexity, it would be reasonable to premise these basic operations on a two-dimensional plane. Given such a background, "origami" and "algorithm" are compatible.

In computer science, the efficiency of an algorithm is measured by the time complexity as the number of basic operations to be performed and the space complexity as the memory area to be used. What is equivalent to such "time complexity" or "space complexity" in "origami algorithm"? Regarding the time complexity, "number of folds" based on the basic axioms is considered as a natural correspondence. This has the name *folding complexity*. On the other hand, what about space complexity? In recent years, the notion of crease width has been proposed, and research has begun. Nonetheless, these notions are still not absolute and research has just begun.

Here, we introduce models of "origami algorithm" and "computational complexity of origami" and the latest results. First, I clarify "model": "origami" handled in this section is a one-dimensional line segment. In other words, for example, imagine that the origami is an elongated paper tape with vertical creases. Moreover, most of the current results are the results of model with equally spaced creases. Yes, it is a very simple model that cannot be further simplified. From the viewpoint of algorithm and computational complexity, although it is such a simple model, there are many problems that have not been solved yet, and it is a surprisingly deep theme. Of course, the following extensions will be easy to consider:

- Extension to non-equidistant creases.
- Extension to 2D plane.
- Extension to diagonal creases.

At the time of writing this book, it is the actual situation that we cannot reach such extensions. Conversely, it is an unexplored area where there is still room for

cultivation. We will introduce these undeveloped areas in the latter part of Part III and Part IV. In the field of computational origami, it is not uncommon that discovery of college students, and in some cases, high school students, tends research to make progress. While experiencing the interest of computational origami as applications of computer science, you might want to think about what to do about various studies.

Computational complexity in origami
Both of the notions "folding complexity" and "crease width" introduced in this book are invented by me and named by my collaborator Erik D. Demaine, who is a professor at MIT, and one of his favorite research topics is computational origami. (In fact, he is the godfather of the research area "computational origami"!) At a time when both notions were presented and discussed, the results gradually came out, and they were aptly named. Since they have good compatibility with the philosophy of computer science in various viewpoints, research of these notions is expected to make progress in the future. In the past, several attempts at research of "cost of folding" had been made. For example, there was some research to estimate the cost of origami design by the summation of the cost which corresponds to each folding. However, it seems that none of them has been established. As far as I know, it is the first time that we investigate the complexity of origami from the viewpoint of computer science in this book.