Requirement and Specification

This section describes the requirements and specifications of the software. We hope specifying the requirements can better help to build a software for users to understand the principle of sorting algorithms and prove their correctness.. The first part is an overall description of the user requirements of both functional and non-functional. The second part outlines functional system specifications and related non-functional specifications of the software.

User Requirements

Functional Requirements

What a steakeholder needs to be able to do.

Users want to learn sorting algorithms through the software.

Users want to watch animations of the process of sorting algorithms.

Users want to understand and prove the correctness of sorting algorithms.

Users want the software to support multiple languages

Non-functional Requirements

Constraints on what a steakeholder needs to be able to do.

The software can only animate bubble sort, selection sort, insertion sort, merge sort, quick sort, heap sort.

System Specifications

Functional Specification

What the software must do to meet the user requirements.

The list is divided into two parts. The core feature list contains functions with high priority while the additional one consists of optional functions.

Core features

1. Hierarchical learning feature

1.1 A user shall be able to choose a difficulty level according to individual mastery level of algorithms.

2. Sorting animation related features

2.1.1 A user shall be able to select a type of sorting algorithm.

2.1.2 A user shall be able to view the animation, which shows the sorting algorithm process.

2.1.3 A user shall be able to view explanations of each step of sorting algorithms.

2.2.1 A user shall be able to start from different time frames of an animation.

2.2.2 A user shall be able to customize the array of numbers to be animated

2.2.3 A user shall be able to ask the software to randomly generate input.

3. Correctness related features

3.1.1 A user shall be able to test an algorithm through several sets of legal inputs to see if it can work correctly.

3.1.2 A user shall be able to see explanations of sorting algorithms’ correctness.

3.2.1 A user shall be able to choose from several sets of illegal inputs.

3.2.2 A user shall be able to test an algorithm which cannot work out the sorting job.

3.2.3 A user shall be able to view explanations of why a sorting job of an sorting algorithm fails.

4. Open source

4.1 A user shall be able to access the source code of this software.

Additional features

1.

2. A user shall be able to follow a guide of basic algorithms.

3. A user shall be able to download learning notes provided by the software.

4.

A user shall be able to view history of the latest learned sorting algorithms.

5. A user shall be able to compare different sorting algorithms’ time complexity.

6. A user shall be able to view individual learning progress.

7. A user shall be able to do after-class exercises.

8. A user shall be able to practice by sorting an array of numbers manually.

9. A user shall be able to create breakpoints to see outputs at that specific point.

10. A user shall be able to share the information of the software.

11. A user shall be able to view the pseudo-code of sorting algorithms.

12. A user shall be able to view the programming code of sorting algorithms.

13. A user shall be able to write feedbacks to developers.

14. A user shall be able to switch languages.

Non-functional Specifications

Constraints on what the software must do to meet the user requirements.

1. Interoperability

1.1 The software can only run on Windows system.

1. Usability:

2.1 The software can only support English and Chinese.

2.2 The software can only provide programming code in C, Python, and Java language.

1. Capability

3.1 The hardware requirements for basic performance is 2GB RAM and 200MB storage.

The software should be able to open in 1 second by users.