**《数据结构》课程设计总结**

****

**学 号 ： 1952889**

**姓 名 ： 鲁灵伊**

**专业/年级： 信息安全**

**2021 年 7 月**

目录

[第一部分 算法实现设计说明 1](#_Toc81236295)

[1.1 题目 1](#_Toc81236296)

[1.2 软件功能 1](#_Toc81236297)

[1.2.1 控制面板部分 1](#_Toc81236298)

[1.2.2 排序分步实现部分 1](#_Toc81236299)

[1.3 设计思想 2](#_Toc81236300)

[1.4 逻辑结构与物理结构 3](#_Toc81236301)

[1.4.1 逻辑结构 3](#_Toc81236302)

[1.4.2 物理结构 3](#_Toc81236303)

[1.5 开发平台 3](#_Toc81236304)

[1.6 系统的运行结果与结果分析 3](#_Toc81236305)

[1.7 操作说明 3](#_Toc81236306)

[第二部分 综合应用设计说明 5](#_Toc81236307)

[2.1 题目 5](#_Toc81236308)

[2.2 软件功能 5](#_Toc81236309)

[2.2.1 控制面板部分 5](#_Toc81236310)

[2.2.2 模拟游戏部分 5](#_Toc81236311)

[2.3 设计思想 6](#_Toc81236312)

[2.4 逻辑结构与物理结构 6](#_Toc81236313)

[2.4.1 逻辑结构 6](#_Toc81236314)

[2.4.2 物理结构 7](#_Toc81236315)

[2.5 开发平台 7](#_Toc81236316)

[2.6 系统的运行结果分析说明 7](#_Toc81236317)

[2.7 操作说明 7](#_Toc81236318)

[第三部分 实践总结 8](#_Toc81236319)

[3.1 所做的工作 8](#_Toc81236320)

[3.2 总结与收获 9](#_Toc81236321)

[第四部分 参考文献 9](#_Toc81236322)

# 第一部分 算法实现设计说明

## 题目

几种排序：要求随机输入一组数据，随时给出某一趟排序的变化情况

1. 直接插入排序、折半插入排序、希尔排序；
2. 冒泡排序、快速排序；
3. 简单选择排序；

## 软件功能

该程序共有两个功能，分别为控制面板部分和排序分步实现部分。

### 控制面板部分

控制面板部分集合了排序过程需要的数组输入、程序运行、暂停和重启以及速度控制这三个功能，可以实现不同排序类型的切换、生成随机的数组序列、输入并判断手动输入的序列并调整动画演示的速度、启动排序过程。

* 1. 数组输入。数组输入包括手动输入和自动生成两种形式。手动输入形式可以支持任意长度的数组，针对输入错误，控制面板会自动判断非法字符和多余空格。而自动生成形式可以支持5至30个数组元素的随机生成，其中每个元素的数值范围为1-99。
  2. 程序运行、暂停和重启。程序运行、暂停和重启在第一步数组输入完成后方可实现。程序运行是指启动排序过程，显示分步排序结果；程序暂停是指在当前排序过程未结束前，完成当前的数组元素比较或元素移动后暂停，因此设置了暂停的延迟时间，保证动画演示的完整性；程序重启是指在程序运行或者暂停，或者结束后恢复最初的排列顺序。
  3. 速度控制。在排序分步实现的动画演示过程中，数组元素比较默认花费500毫秒，数组元素移动默认耗时1000毫秒，该演示速度对于理解排序过程较为合理，如果用户需要调整演示速度，可以通过速度控制功能实现。默认速度为3，数组元素比较时间为1500/速度毫秒，数组元素移动时间为3000/速度毫秒。

### 排序分步实现部分

根据题目要求，几种排序方式需要结合动画演示，给出分步运行的结果，其中动画演示包括元素的比较和移动。所有排序均为递增排序。

1. 元素比较。元素比较是排序过程中重要的步骤，每个排序过程的两元素比较均不相同。以冒泡排序为例，数组元素的比较针对的是相邻的两个元素，如果左边元素数值大于元素，则需要启动一次移动。再以希尔排序为例，数组元素的比较对象针对的是间隔元素，而非相邻元素。

元素比较默认花费0.5秒，演示中两个比较元素均为粉色。

1. 元素移动。元素移动是元素比较后关键的一步，当两个比较的元素左大右小，则需要将元素的位置互换。

元素移动默认花费1秒，演示中移动元素为黄色。

## 设计思想

本程序由React前端框架构建，在开发过程中，利用React的官方脚手架create-react-app搭建而成，其编程语言为jsx，即运用于React架构中的JavaScript的语法扩展；实现排序的核心代码利用JavaScript生成。

脚手架的安装：具体实现过程为下载Node >= 10.16 and npm >= 5.6，输入以下代码，

|  |
| --- |
| npx create-react-app my-app  cd my-app  npm start |

利用脚手架提供的组件以及主题定义，在src文件夹中扩展自己的文件。

在算法实现设计中，源文件的组成如下。

Components中包含了所有组件，包括导航栏、控制面板以及排序数组演示，其中排序数组演示是利用visualizer中SortManager和ArrayContainer实现的。Function中包含了六种排序的核心代码，采用异步函数能够使排序过程中演示元素比较与移动元素的过程。Common中包含了四个文件，分别为常量的确定、功能函数、状态管理以及样式设计。

以下是对Components中每一项组件和组件之间联系的介绍。

NavBar导航栏实现选择六种排序之一，主要实现工具为Material-UI中的Tab panels功能。根据Tab panel的选项来确定接下来调用的是何种排序的核心代码。

Controller控制面板实现数组输入、速度控制和演示的播放、暂停与重启。数组输入的结果会由状态管理器useData记录，同样的，速度控制选项会影响演示时间，而播放、暂停、重启的状态变化由状态管理器useControls记录。状态管理器useData, useControls由common/store管理。

AlgoDisplay排序数组演示由visualizer中的SortManager实现，其主要功能是确定当前的播放状态，该播放状态由状态管理器useControls记录；调用当前排序的核心代码，引入异步的演示函数，包括元素比较和元素移动。SortManager由ArrayContainer实现，目的是根据SortManager确定的待比较元素和待移动元素，分别给予动画演示的支持，动画演示由common/styles提供样式。

## 逻辑结构与物理结构

### 逻辑结构

逻辑结构为线性结构。线性结构中元素之间是一对一的关系。

### 物理结构

物理结构为顺序存储结构。顺序存储结构是由一段连续的内存结构组成，大小固定且访存方便，但插入删除效率低。

## 开发平台

集成开发环境（IDE）： Visual Studio Code v1.59.1

Web前端开发框架：React ^latest

Web前端React UI框架：Material-UI

@material-ui/core v4.11.1

@material-ui/icons v4.11.2

JavaScript运行环境：Node.js v14.17.3

第三方程序包或开源代码：

react-router-dom v5.2.0 基于react-router提供在浏览器运行环境下的路由组件。

styled-components v5.3.0 针对React的css-in-js框架，用于增强React组件中css的表现。

zustand v3.5.7 一个小型、快速、可扩展的React状态管理解决方案。

## 系统的运行结果与结果分析

本程序利用Visual Studio Code进行编程，在开发过程中，利用React的官方脚手架create-react-app搭建而成，其编程语言为jsx，即运用于React架构中的JavaScript的语法扩展。

第一，开发过程中要保证版本正确。

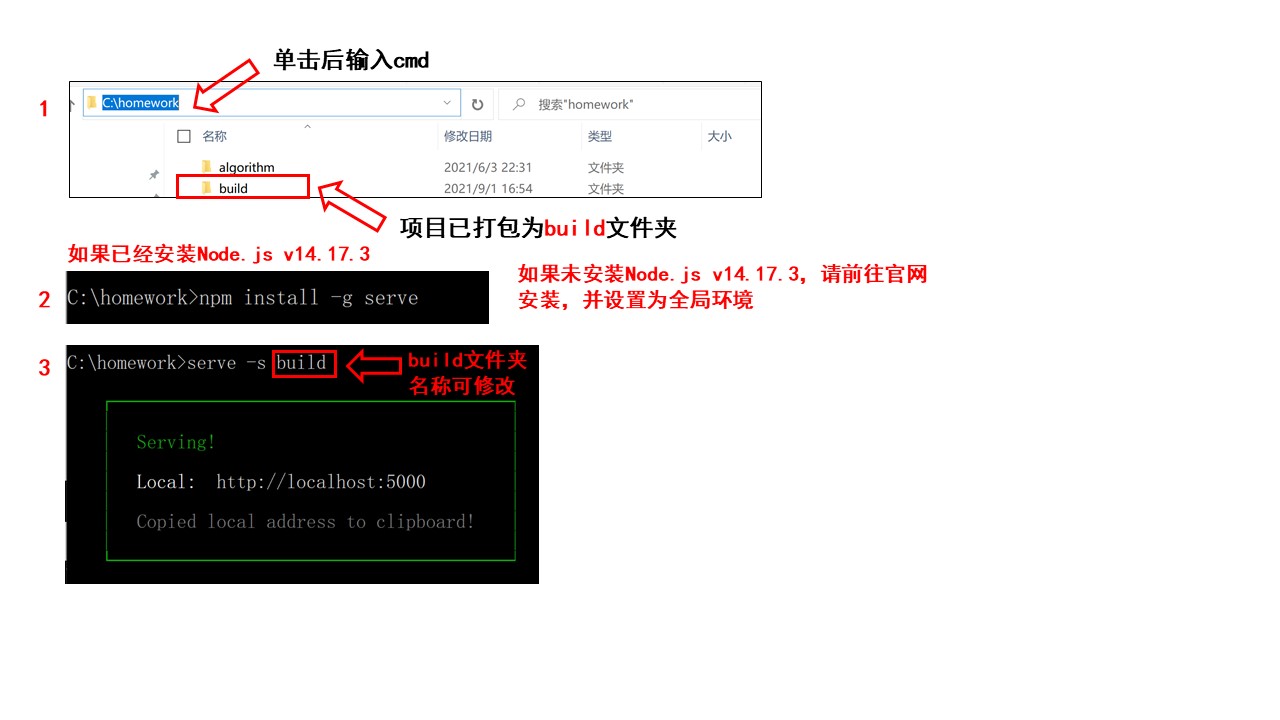
## 操作说明

运行说明：

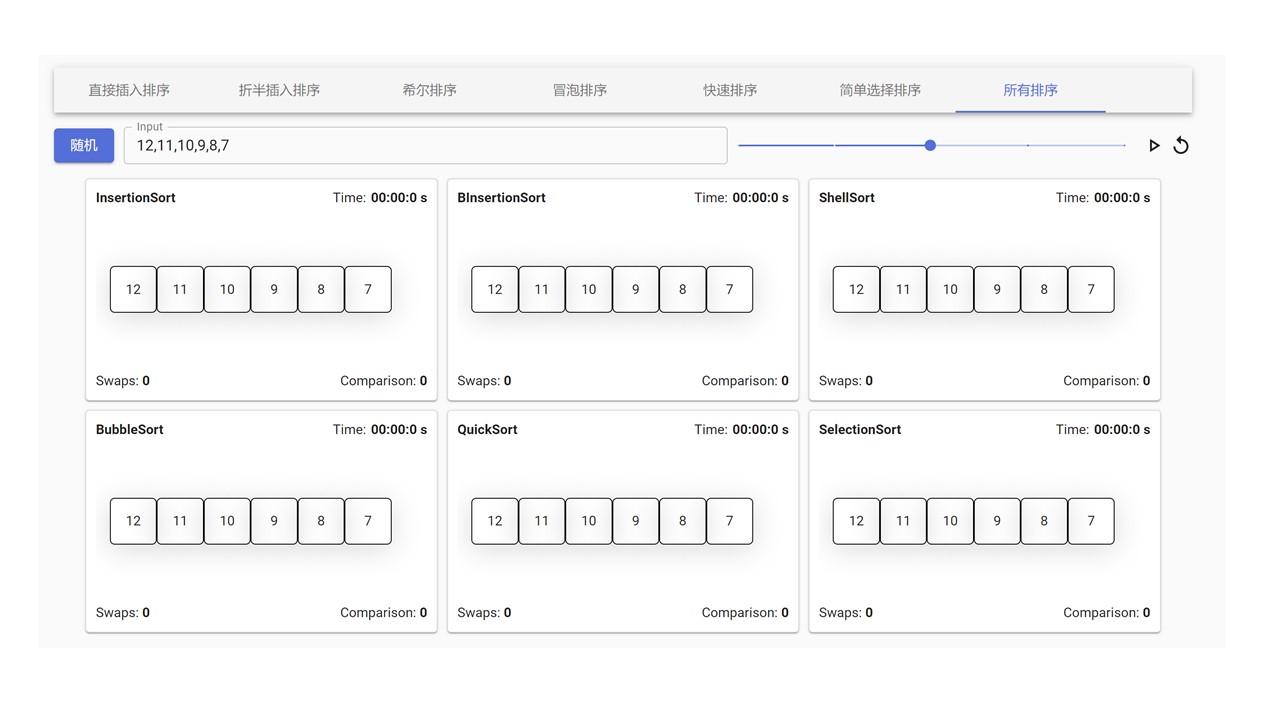
1. 安装Node.js v14.17.3，设置为全局环境。
2. 在文件夹build所在目录内打开cmd。
3. 输入以下代码。

|  |
| --- |
| npm install -g serve  serve -s build |

1. 复制cmd内的本地链接，项目已部署到本地localhost服务器，打开浏览器并粘贴链接即可运行。



页面展示：



控制面板操作：



排序演示说明：



# 第二部分 综合应用设计说明

## 题目

编号是1，2，……，n的n个人按照顺时针方向围坐一圈扔骰子（1-6），先选取一个人扔，按照扔的数字m，从扔骰子的人开始从1沿顺时针方向顺序报数，报到m时停止报数，报m的人出列，然后从他在顺时针方向的下一个人扔骰子，扔完后从1开始报数，如此，直到剩下一个人胜出。设计一个程序模拟这一过程。

1. 通过输入框输入1，2，……，n个人。
2. 模拟整个游戏过程。
3. 按照出列的顺序输出各个人的编号。

## 软件功能

该程序共有两个功能，分别为控制面板部分和模拟游戏部分。

### 控制面板部分

控制面板部分集合了数字输入、速度控制以及程序的运行、暂停与结束。

* 1. 数字输入。与算法实现设计类似，综合应用设计要求输入一个数字n，在程序中我们规定该数字的范围为1~99，同时可以实现对输入的检查，排除非法字符或空格。另外，数字输入可以由随机生成替代，其范围仍为1~99。
  2. 速度控制。在约瑟夫环的动画演示过程中，第m个人出列花费1000毫秒，计数耗时500毫秒，该速度对于观看动画演示过程较为合理，如果用户需要调整演示速度，可以通过速度控制功能实现。默认速度为3，计数时间为3000/速度毫秒，第m个人出列时间为1500/速度毫秒。
  3. 程序的运行、暂停与结束。程序运行、暂停和重启在第一步数字输入完成后方可实现。程序运行是指启动游戏过程，显示出列结果；程序暂停是指在当前游戏过程中随时暂停；程序重启是指在程序运行或者暂停，或者结束后恢复最初的人形圆圈。

### 模拟游戏部分

模拟游戏部分包括人形圆圈、1~m报数、第m人出列和显示出列顺序。

1. 人形圆圈。根据游戏规则，游戏开始前，1~n人顺时针方向一圈，此时在屏幕中央有一圆圈，该圆圈根据人数变化随时调整半径。
2. 1~m报数。根据游戏规则，在游戏过程中需要确定每一轮的随机数m，数字m由软件内部的随机数生成器生成，而非由玩家确定，在游戏过程中会随时显示每一轮的随机数m的结果，并根据该结果，顺时针方向计数。

当前报数人显示为粉色，计数时间为0.5秒。

1. 第m人出列。根据游戏规则，报到随机数m的人出列，离开人形圆圈，不再参与计数。

出列的人显示为黄色，出列时间耗时1秒。

1. 显示出列顺序。每一轮结束后，出列的人会单独出现在出列队伍中，此出列队伍只进不出，每轮动态更新。

## 设计思想

在综合应用中，源程序的组成如下。

在综合应用的界面中，依然保留了与算法实现题类似的导航栏、控制面板以及游戏动画演示，其中游戏动画演示是利用visualizer中的JosephContainer和JosephManager实现的。核心的游戏执行代码由Function中的Joseph实现，采用异步函数，将游戏的随机数生成、计数和出列过程一一展现。Common中包含了四个文件，分别为常量的确定、功能函数、状态管理以及样式设计。

以下是对Components中每一项组件和组件之间联系的介绍。

NavBar导航栏显示Joseph Circle选项，主要实现工具为Material-UI中的Tab panels功能。

Controller控制面板实现数字输入、速度控制和演示的播放、暂停与重启。数字输入的结果会由状态管理器useControl记录，同样的，速度控制选项会影响演示时间，而播放、暂停、重启的状态变化由状态管理器useControl记录。状态管理器useControl由common/store管理。

AlgoDisplay游戏演示由visualizer中的JosephManager实现，其主要功能是确定当前的播放状态，该播放状态由状态管理器useControl记录；调用约瑟夫环游戏的核心代码，引入异步的演示函数，包括游戏的报数过程和出列动画。JosephManager由JosephContainer实现，目的是根据JosephManager确定的当前报数人和出列人，分别给予动画演示的支持，动画演示由common/styles提供样式。为了给出出列顺序，在JosephContainer中还要另外展示出列元素，出列队列由JosephManager提供，由核心代码Joseph生成。

## 逻辑结构与物理结构

### 逻辑结构

逻辑结构为线性结构。线性结构中元素之间是一对一的关系。

### 物理结构

物理结构为顺序存储结构。顺序存储结构是由一段连续的内存结构组成，大小固定且访存方便，但插入删除效率低。

## 开发平台

集成开发环境（IDE）： Visual Studio Code v1.59.1

Web前端开发框架：React ^latest

Web前端React UI框架：Material-UI

@material-ui/core v4.11.1

@material-ui/icons v4.11.2

JavaScript运行环境：Node.js v14.17.3

第三方程序包或开源代码：

react-router-dom v5.2.0 基于react-router提供在浏览器运行环境下的路由组件。

styled-components v5.3.0 针对React的css-in-js框架，用于增强React组件中css的表现。

zustand v3.5.7 一个小型、快速、可扩展的React状态管理解决方案。

## 系统的运行结果分析说明

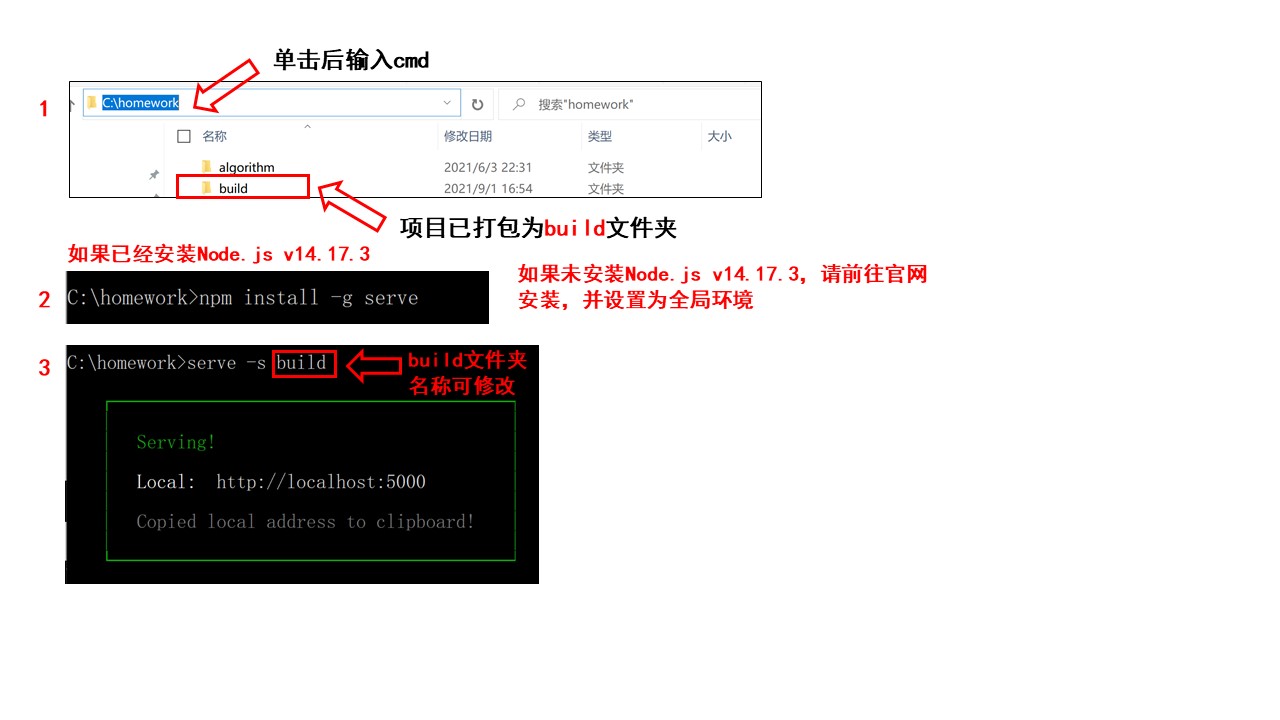
## 操作说明

运行说明：

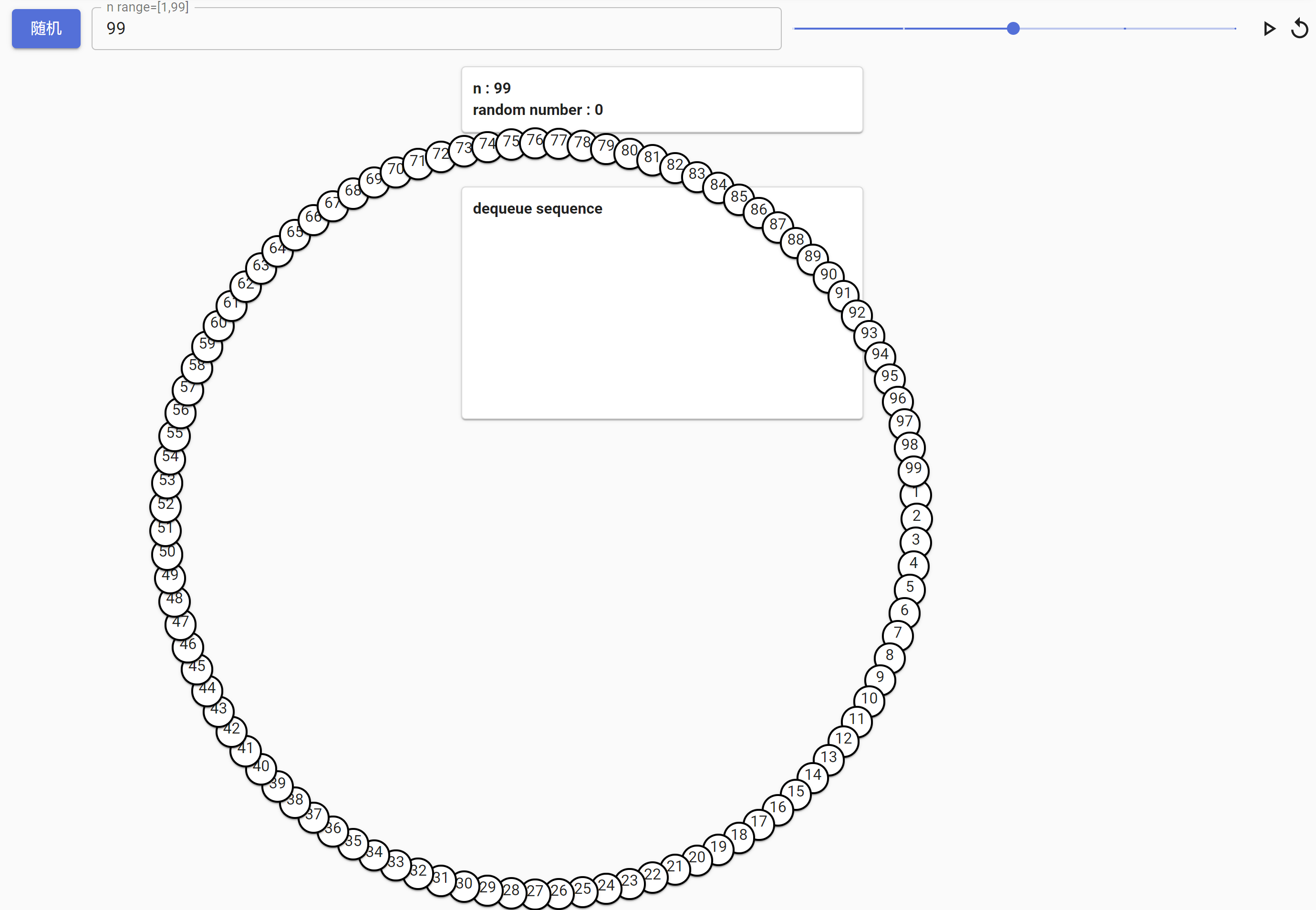
1. 安装Node.js v14.17.3，设置为全局环境。
2. 在文件夹build所在目录内打开cmd。
3. 输入以下代码。

|  |
| --- |
| npm install -g serve  serve -s build |

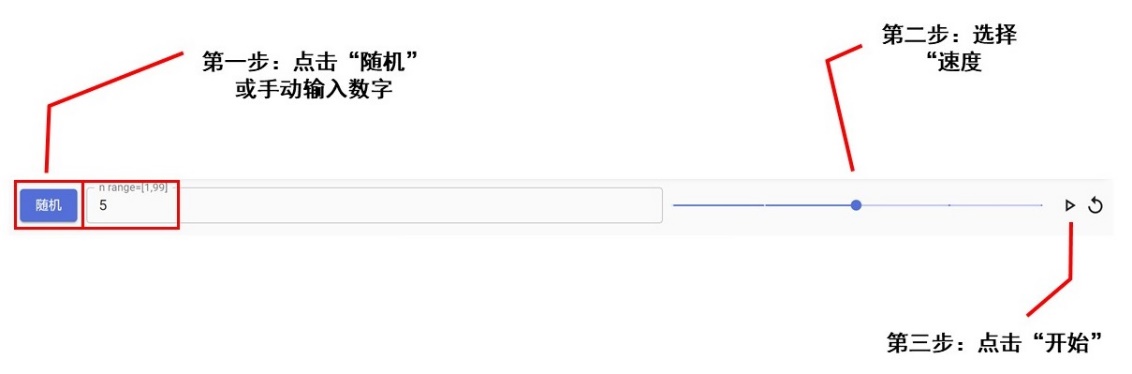
1. 复制cmd内的本地链接，项目已部署到本地localhost服务器，打开浏览器并粘贴链接即可运行。



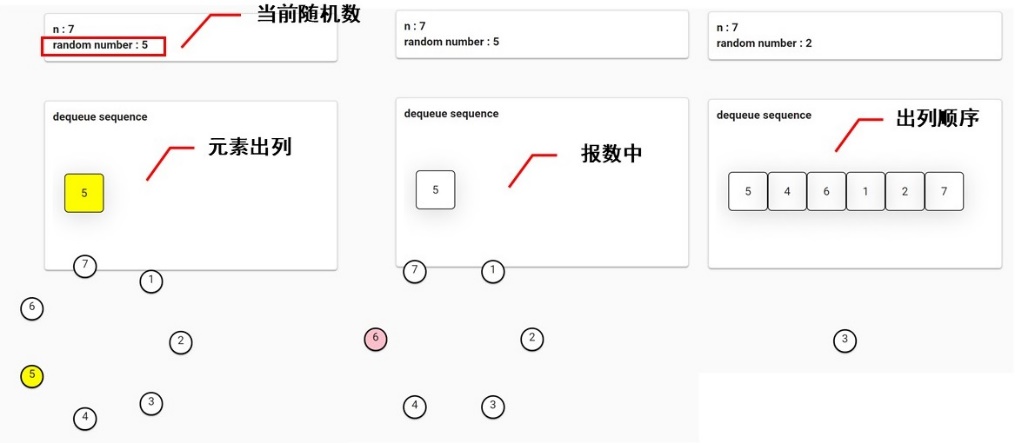
页面展示：



控制面板：



**动画演示：**



# 第三部分 实践总结

## 所做的工作

## 总结与收获

# 第四部分 参考文献