数独游戏设计文档

班级： \*\*\*\*\*\*\*\*\*\* 姓名： \*\*\*\*\*\* 学号： \*\*\*\*\*\*\*\*\*\*

1. 问题描述

设计并实现一个基本的数独游戏。功能包括根据用户选择的级别给出初始盘面数字（级别越低，数字越多），并且能够实时监测并提示冲突；当用户完成后，保存最近十次的成绩（完成一次所需时间）以及最好成绩。

1. 问题分析

数独是源自18世纪瑞士的一种数学游戏。是一种运用纸、笔进行演算的[逻辑游戏](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E6%B8%B8%E6%88%8F/12814189)。玩家需要根据9×9盘面上的已知数字，推理出所有剩余空格的数字，并满足每一行、每一列、每一个粗线宫（3\*3）内的数字均含1-9，不重复。

因此，核心问题就是我们需要得到一个含有一定数目空格的九宫格，并且一定存在一种填写方式使得九宫格满足数独的要求。

我们可以先得到一个完整的，满足每一行、每一列、每一个粗线宫（3\*3）内的数字均含1-9，不重复的数独表。然后随机得到数独表中空格的位置，在通过可视化生成游戏界面，实现交互功能。

在用户与界面交互过程中，实时监测冲突是否产生，检验提交的答案是否正确，保存最近十次成绩和最好的一次成绩。

1. 算法设计

本游戏为使用JavaScript开发的网页数独游戏，因此以下算法实现为JavaScript实现。

经问题分析可得我们需要解决的几个核心子问题，并设计算法如下：

1. 传统回溯法生成数独表

网上提供的数独算法多种多样，但绝大部分的核心思想依然是传统回溯法：

通过一个9\*9的二维数组sudu表示一个数独九宫格。

首先初始化该数组，值都为0，然后将该九宫格的第一行依次随机填入1-9中的一个数字。接下来，从第二行开始，遍历该数组剩下的元素。

每遍历到一个位置，随机填入1-9中的一个数字，然后判断这个九宫格目前的所有数字是否满足每一行、每一列、每一个粗线宫（3\*3）内的数字均含1-9，不重复的要求。如果满足，继续向下一个位置遍历；如果不满足，则重新随机填写该位置。当重复填写了十次之后依然不满足要求，则回溯到上一个位置，重新填写该位置。

直到遍历完所有数组后，我们也就得到了一个完整的数独表。用sudu表示。

Math.floor(Math.random() \* 9 + 1)可得到1-9范围内的随机整数。

1. 随机生成难度级别不同的数独

由（一）得，sudu数组保存着完整的数独表，那么游戏中的有空格的数独表可以用9\*9的二维数组curSudu表示，其中空格位置的数字设为0。如果填入了数字，则实时修改curSudu中该位置的值。

通过空格个数的不同来区分数独难度级别，空格个数越多，难度级别越高。设计该游戏难度级别有三个，分别为，初级：15个空格；中级：30个空格；高级：45个空格。

欲实现该需求，则先根据难度随机生成对应数目的空格的位置坐标（i,j），用数组表示为：blanks=[{row: i, column: j},……]。其中0 <= i <= 8, 0 <= j <= 8。通过Math.floor(Math.random() \* 9)即可生成需要的i，j。curSudu一开始等于sudu，得到blanks后将curSudu对应空格的值设为0。

1. 用户填写数独时实时判断是否产生冲突

用户每填写完一个数字，都会触发判断冲突事件。由于之前填过的数字都被判断过，所以只需要判断刚刚填入的数字所在的行，列，粗线宫内是否满足条件。

首先保存刚刚填入的数字num，然后对所在的行，列，（3\*3）区域，分别进行遍历，假如遍历到的数字等于刚刚填入的数字并且所在的位置不同，则该填入数字在九宫格内产生了冲突；否则没有冲突。

1. 保存最近十次成绩和最好成绩

设计成绩的数据结构为：

let user = {

username: username,

time: time

}

时间表示成绩高低，时间越短，成绩越好。

每当游戏开始时，将当前时刻距 1970 年 1 月 1 日之间的毫秒数保存在全局变量time中。当提交答案，游戏结束时，time = ((new Date().getTime() - time) / 1000 / 60).toFixed(2)，得到从开始到结束的时间，以分钟为单位，保留两位小数。

每次提交成绩时，将用户名和成绩保存在user中。然后从数据中通过getRank()得到最近的十次成绩（用长度为10的一维数组ranks表示），通过getMaxGrade()得到最好的一次成绩maxGrade。

将user.time与maxGrade.time进行比较，如果user.time比maxGrade.time更小，则maxGrade=user；然后将user插入ranks的最前面，去掉ranks最后一个元素，将ranks的长度保持在十。然后存入数据中

1. 软件架构

该软件是由JavaScript，HTML，CSS编写的的网页游戏，其中主要逻辑算法由JavaScript完成。而其软件架构为MVC模式

1. 控制层（Controller）

JavaScript是一门事件驱动型的语言，通过在网页的特定DOM中绑定相应的事件监听，当用户在界面上特定的DOM元素上进行特定操作时就会触发对应的事件监听函数时，对逻辑进行处理。

1. 视图（View）

该软件的界面由HTML和CSS决定，文件夹中的index.html和style.css会在浏览器的运行下生成对应的DOM树与样式，并在浏览器上渲染出界面。Js可以修改DOM树和样式，从而改变视图。

1. 模型（Model）

sudu.js和index.js文件中保存着实现处理事件的对应代码。而数据保存在浏览器的localstorage中。因为由于规定对浏览器的限制，浏览器只能读取本地文件，却不能修改本地文件，所以在不使用服务器的情况下，本软件通过浏览器的localstorage来保存修改成绩。

1. 实现与测试
2. 软件实现

A．逻辑处理

1）生成数独表和判断是否冲突的函数

在sudu.js中定义Sudu函数，返回函数：

回溯法生成数独表：gennerateShudu(),

判断填入元素所在的行，列，（3\*3）是否冲突：checkRow(), checkColumn(), checkNine()。

以下为代码：

let Sudu = (function() {

//初始化二维数组

let gennerateArr = function () {

let arr = new Array(9);

for (let i = 0; i < arr.length; i++) {

arr[i] = new Array(9);

arr[i].fill(0, 0, 9);

}

return arr

}

//对二维数组的第一行随机填入不重复的1-9数字

let init = function(firstRow) {

for (let i = 0; i < firstRow.length; i++) {

while (true) {

let rand = Math.floor(Math.random() \* 9 + 1);

if (firstRow.indexOf(rand) === -1) {

firstRow[i] = rand;

break;

}

}

}

}

//在生成数独表的过程中，判断填入的数字是否会冲突

let judge = function(row, column, num, sudu) {

//judge row

for (let i = 0; i < column; i++) {

if (sudu[row][i] === num) {

return false;

}

}

//judge column

for (let i = 0; i < row; i++) {

if (sudu[i][column] === num) {

return false;

}

}

//judge local

let count = column % 3 + row % 3 \* 3;

while (count--) {

//console.log(row - row % 3 + Math.floor(count / 3), column - column % 3 + count % 3);

if (sudu[row - row % 3 + Math.floor(count / 3)][column - column % 3 + count % 3] === num) {

return false;

}

}

return true;

}

//回溯法生成完整数独表

let gennerateShudu = function() {

let sudu = gennerateArr();

init(sudu[0]);

let filltime = 0;

for (let i = 1; i < 9; i++) {

for (let j = 0; j < 9; j++) {

filltime = 0;

while(filltime < 10) {

let num = Math.floor(Math.random() \* 9 + 1);

if (judge(i, j, num, sudu)) {

sudu[i][j] = num;

break;

} else {

filltime++;

}

}

if (filltime >= 10) {

if (j === 0) {

i--;

j = 8;

} else {

j--;

j--;

}

}

}

}

return sudu;

}

let checkRow = function(row, column, num, curSudu) {

for (let i = 0; i < 9; i++) {

if (curSudu[row][i] == 0) {

continue;

}

if (curSudu[row][i] == num && i != column) {

return false;

}

}

return true;

}

let checkColumn = function(row, column, num, curSudu) {

for (let i = 0; i < 9; i++) {

if (curSudu[i][column] == 0) {

continue;

}

if (curSudu[i][column] == num && i != row) {

return false;

}

}

return true;

}

let checkNine = function (row, column, num, curSudu) {

let j = Math.floor(row / 3) \* 3;

let k = Math.floor(column / 3) \* 3;

// 循环比较

for (let i = 0; i < 8; i++) {

if (curSudu[j + Math.floor(i / 3)][k + i % 3] == 0) {

continue;

}

if (curSudu[j + Math.floor(i / 3)][k + Math.round(i % 3)] == num && row != j + Math.floor(i / 3) && column != k + Math.round(i % 3)) {

return false;

}

}

return true;

}

return {

gennerateShudu: gennerateShudu,

checkRow: checkRow,

checkColumn: checkColumn,

checkNine: checkNine

};

})()

B．游戏的逻辑和界面的处理

let levelBlank = { //难度对应的空格数目

"初级": 15,

"中级": 30,

"高级": 45

}

let sudu; //保存完整数独表的二维数组

let curSudu; //保存当前游戏的数独表的二维数组

let status = 0; //0：准备 1：游戏中

let time; //保存游戏开始的时间或者游戏所用时间

//randomBlank

//params: level 难度级别

//return blanks 所有空格的坐标

let randomBlank = function(level) {

let blankNum = levelBlank[level];

let blanks = [];

for (let i = 0; i < blankNum; i++) {

let blank = {

row: Math.floor(Math.random() \* 9),

column: Math.floor(Math.random() \* 9)

}

let isIn = false;

for (let j = 0; j < blanks.length; j++) {

if (blank.row === blanks[j].row && blank.column === blanks[j].column) {

isIn = true;

break;

}

}

if (!isIn) {

blanks.push(blank);

} else {

i--;

}

}

return blanks;

}

//showSudu

//params:

//return

//开始游戏，生成完整数独表的二维数组，保存在sudu中，初始化curSudu，渲染数独游戏界面

let showSudu = function () {

let level = document.getElementsByClassName("select")[0].value;

let blankIndexs = randomBlank(level);

try {

sudu = Sudu.gennerateShudu();

status = 1;

curSudu = new Array(9);

for (let i = 0; i < sudu.length; i++) {

curSudu[i] = [];

for (let j = 0; j < sudu[i].length; j++) {

curSudu[i].push(sudu[i][j]);

}

}

let board = document.getElementsByClassName("game")[0];

let inputs = board.children;

for (let i = 0; i < blankIndexs.length; i++) {

let row = blankIndexs[i].row;

let column = blankIndexs[i].column;

curSudu[row][column] = 0;

}

for (let i = 0; i < curSudu.length; i++) {

for (let j = 0; j < curSudu[i].length; j++) {

inputs[j\*9+i].id = i + "&" + j;

if (curSudu[i][j] != 0) {

inputs[j\*9+i].value = curSudu[i][j];

inputs[j\*9+i].readOnly = "readonly";

inputs[j\*9+i].style.backgroundColor = "";

} else {

inputs[j\*9+i].value = "";

inputs[j\*9+i].readOnly = "";

inputs[j\*9+i].style.backgroundColor = "white";

inputs[j\*9+i].addEventListener("change", handleChange);

}

}

}

time = new Date().getTime();

}

catch (e) {

console.log(e);

}

}

//handleChange

//params: e 空格的值发生改变的事件

//return

//当空格填入的值发生改变时，对值进行监测，检查是否冲突，进行提示

let handleChange = function(e) {

let element = e.currentTarget;

let indexs = element.id.split("&");

curSudu[indexs[0]][indexs[1]] = element.value == '' ? 0 : element.value;

let tip = document.getElementsByClassName("cur\_situation")[0].children[0];

if (!(Sudu.checkRow(indexs[0], indexs[1], element.value, curSudu) && Sudu.checkColumn(indexs[0], indexs[1], element.value, curSudu) && Sudu.checkNine(indexs[0], indexs[1], element.value, curSudu))) {

tip.innerHTML = "\n第" + indexs[0] + "行第" + indexs[1] + "列冲突！";

element.style.backgroundColor = "#f34949";

} else {

tip.innerHTML = "";

element.style.backgroundColor = "white";

}

}

//isAnswerRight

//params:

//return boolean 提交的答案是否正确

let isAnswerRight = function() {

for (let i = 0; i < sudu.length; i++) {

for (let j = 0; j < sudu[i].length; j++) {

if (curSudu[i][j] != sudu[i][j]) {

return false;

}

}

}

return true;

}

//isAnswerFull

//params:

//return boolean 提交的数独表是否已经填满

let isAnswerFull = function () {

for (let i = 0; i < curSudu.length; i++) {

for (let j = 0; j < curSudu[i].length; j++) {

if (curSudu[i][j] == 0) {

return false;

}

}

}

return true;

}

//submitAnswer

//params:

//return

//提交答案时的总处理函数

let submitAnswer = function() {

if (status != 1) {

alert("请选择关卡，开始游戏！");

} else if (!isAnswerFull()) {

alert("请填满空格！！");

} else if (isAnswerRight() && status == 1) {

time = ((new Date().getTime() - time) / 1000 / 60).toFixed(2);

let tip = document.getElementsByClassName("cur\_situation")[0].children[0];

tip.innerHTML = "请选择关卡，继续游戏！"

status = 0;

showElement("username-box");

let timeLable = document.getElementsByClassName("time")[0];

timeLable.innerHTML = time;

let inputs = document.getElementsByClassName("game")[0].children;

Array.from(inputs).forEach(element => {

element.value = "";

element.style.backgroundColor = "#8ca0ff";

element.readOnly = "readonly";

});

} else if (!isAnswerRight()) {

alert("游戏失败！");

}

}

//showElement

//params: className 需要展示的DOM元素的class样式选择器

//return

//在浏览器界面中展示某个Dom元素

let showElement = function(className) {

document.getElementsByClassName(className)[0].style.display = "block";

document.getElementsByClassName("game")[0].style.opacity = 0.5;

document.getElementsByClassName("info")[0].style.opacity = 0.5;

}

//hideElement

//params: className 需要隐藏的DOM元素的class样式选择器

//return

//在浏览器界面中隐藏某个Dom元素

let hideElement = function(className) {

document.getElementsByClassName(className)[0].style.display = "none";

document.getElementsByClassName("game")[0].style.opacity = 1;

document.getElementsByClassName("info")[0].style.opacity = 1;

}

//submitGrade

//params

//return

//提交成绩

let submitGrade = function() {

storeGrade();

hideElement("username-box");

}

//closeChart

//params

//return

//关闭成绩榜单

let closeChart = function(e) {

hideElement("rank-chart")

}

//openChart

//params

//return

//打开成绩榜单

let openChart = function() {

showElement("rank-chart");

let rankLis = document.getElementsByClassName("rank-li")[0].children;

let ranks = getRank();

for (let i = 0; i < ranks.length; i++) {

rankLis[i].getElementsByClassName("username")[0].innerHTML = ranks[i].username;

rankLis[i].getElementsByClassName("user-time")[0].innerHTML = ranks[i].time;

}

for (let i = ranks.length; i < 10; i++) {

rankLis[i].getElementsByClassName("username")[0].innerHTML = '';

rankLis[i].getElementsByClassName("user-time")[0].innerHTML = '';

}

let maxGrade = getMaxGrade();

document.getElementsByClassName('b-username')[0].innerHTML = maxGrade.username;

document.getElementsByClassName('b-time')[0].innerHTML = maxGrade.time;

}

//getRank

//params

//return ranks 最近的十次成绩

//得到最近的十次成绩

let getRank = function() {

let rankStr = '';

let ranks = [];

if (localStorage.getItem('rank')) {

rankStr = localStorage.getItem('rank');

let rankArr = rankStr.split('||');

for (let i = 0; i < rankArr.length; i++) {

let tmp = rankArr[i].split('&');

ranks.push({

username: tmp[0].split('=')[1],

time: tmp[1].split('=')[1]

});

}

}

return ranks;

}

//getMaxGrade

//params

//return maxGrade

//得到最好的一次成绩

let getMaxGrade = function() {

let maxGradeStr = '';

if (localStorage.getItem('maxGrade')) {

maxGradeStr = localStorage.getItem('maxGrade');

}

let maxGrade = {

username: '',

time: ''

};

if (maxGradeStr) {

let arr = maxGradeStr.split('&');

let user = arr[0].split('=');

let time = arr[1].split('=');

maxGrade.username = user[1];

maxGrade.time = time[1];

}

return maxGrade;

}

//storeRank

//params rank 新保存的成绩

//return

//保存刚刚提交的成绩，得到最好的一次成绩并保存

let storeRank = function(rank) {

let ranks = getRank();

ranks.unshift(rank);

for (let i = 0; i < ranks.length - 10; i++) {

ranks.pop();

}

let rankStr = "";

for (let i = 0; i < ranks.length; i++) {

if (rankStr) {

rankStr += `||username=${ranks[i].username}&time=${ranks[i].time}`

} else {

rankStr = `username=${ranks[i].username}&time=${ranks[i].time}`

}

}

let maxGrade = getMaxGrade();

for (let i = 0; i < ranks.length; i++) {

if (ranks[i].time < maxGrade.time) {

maxGrade = ranks[i];

}

}

let maxGradeStr = `username=${maxGrade.username}&time=${maxGrade.time}`;

localStorage.setItem('maxGrade', maxGradeStr);

localStorage.setItem('rank', rankStr);

}

//storeGrade

//params

//return

//处理得到最终成绩并保存

let storeGrade = function() {

let username = document.getElementsByClassName("username")[0].value;

let user = {

username: username,

time: time

}

storeRank(user);

}

1. 软件测试

1、游戏初始界面



2、选择难度为“初级”后点击‘开始游戏’按钮后游戏开始界面



3、在第1行第2列填入5，产生冲突后右边文本框中提示冲突，空格颜色变红



4、在修改数字为9之后，冲突消失界面



5、填满数字提交答案之前的界面



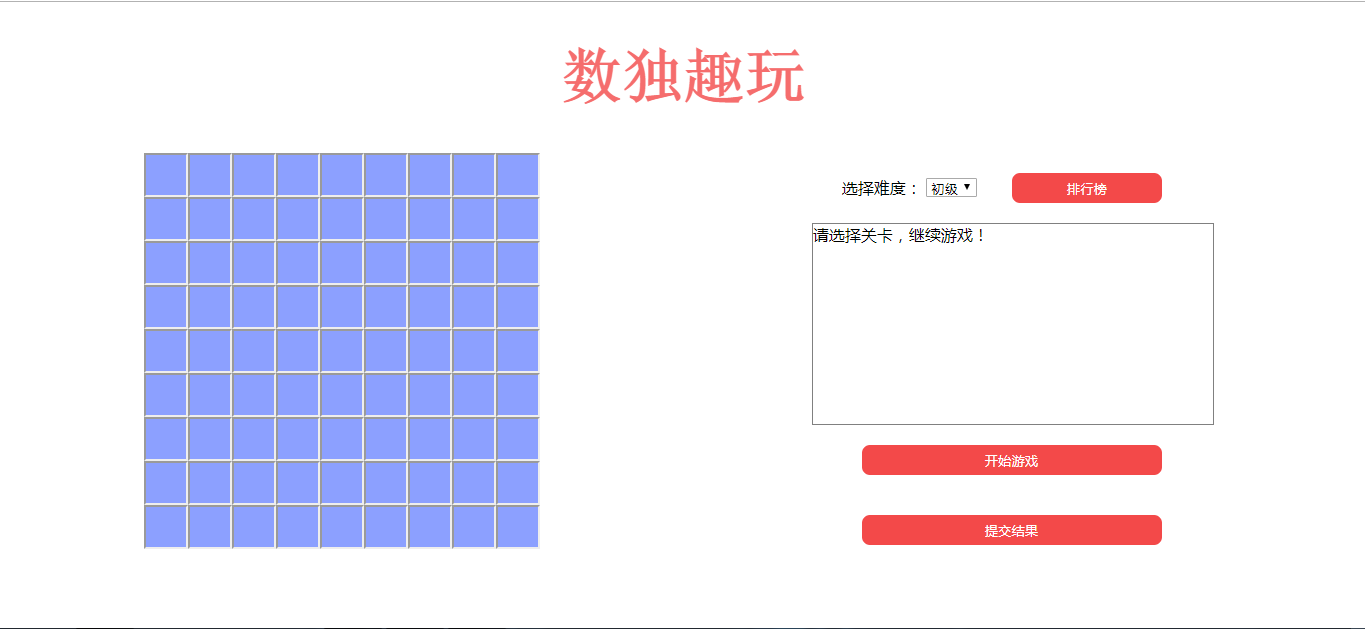
6、点击‘提交结果’按钮后游戏界面，显示用时3.71min，要求输入用户名



7、填入用户名



8、点击‘提交’按钮，成绩保存后的界面



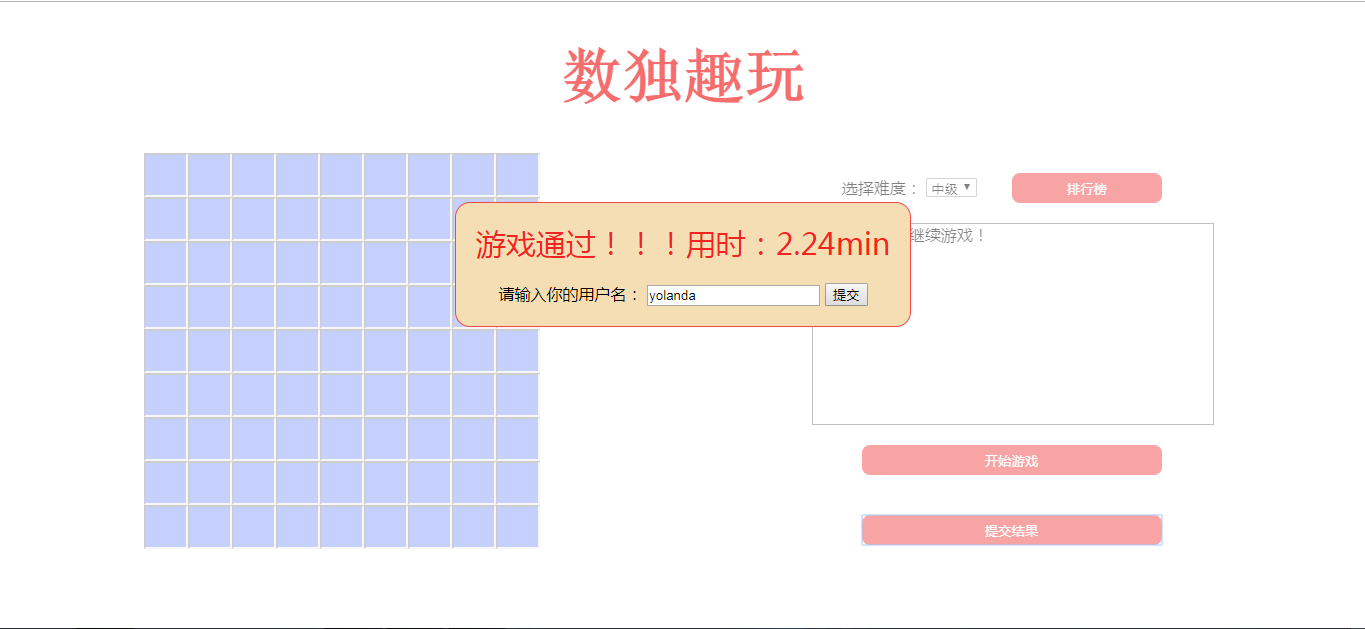
9、点击‘排行榜’按钮，打开成绩榜单，显示最好的一次成绩与最近十次成绩，其中最近一次成绩yolanda:3.71min保存在最前面



10、选择难度为‘中级’，点击‘开始游戏’



11、游戏开始到结束，保存成绩yolanda：2.24min



12、再次打开成绩榜单，可看到刚刚的成绩



1. 性能评价

传统的回溯法的性能较好，通过这种算法可以较快地得到满足数独要求的二维数组。但仍然存在可以优化的余地，不断的回溯的时间复杂度仍然较高，但在游戏中，并不碍事。

但是因为浏览器本身对于内存管理以及计算机的问题，在多次游戏之后，浏览器很有可能会崩溃，此时关掉浏览器，重新打开文件即可。

1. 总结

本次课程设计，我选择了编写数独游戏的课题，一开始时是准备用C语言编写，用easyx的图形库编写图形界面。但是发现easyx的图形库用起来仍然很麻烦。因此最后编写的语言选择了我最为熟悉的JavaScript，根据JavaScript的应用领域，可视化方面选择了网页版本。

本次课程设计较为简单，主要难点便在于如何设计出性能较好的生成完整数独表的算法以及如何在不用服务器的情况下，让浏览器能够对本地的文件进行修改。通过查询资料与自己琢磨，选择了传统的回溯法，并且用JavaScript实现。在上文也提到，由于产商，语言制定者对浏览器和语言的限制，JavaScript无法通过浏览器对本地的文件进行修改，仅能做读取操作，在多方权衡之后，选择了浏览器自带的localStorage，将最近十次成绩的字符串和最好一次成绩的字符串缓存于浏览器中。

通过本次课程设计，我对网页开发更加熟悉，对于浏览器的性能，特性，JavaScript语言的一些特性也更加了解。对于算法的设计也更加熟练，收益匪浅。