ColorSort

排序算法的可视化比较

##### 

**2017年1月5日**

目 录

[第1章 需求分析 1](#_Toc471431869)

[第2章 总体设计 1](#_Toc471431870)

[第3章 编程思想 2](#_Toc471431871)

[3.1 色块的划分 2](#_Toc471431872)

[3.2 颜色的比较 2](#_Toc471431873)

[3.3 像素操作 2](#_Toc471431874)

[第4章 关键环节实现 3](#_Toc471431875)

[4.1 颜色转换 3](#_Toc471431876)

[4.1.1 RGB颜色转HSL 3](#_Toc471431877)

[4.2.2 HSL颜色转RGB 3](#_Toc471431878)

[4.2 色块与对应像素转化 4](#_Toc471431879)

[4.3 色块颜色填充 4](#_Toc471431880)

[4.4 色块打乱 4](#_Toc471431881)

[第5章 运行测试 5](#_Toc471431882)

[第6章 总结 9](#_Toc471431883)

[附录：程序代码 10](#_Toc471431884)

1. 需求分析

比较插入排序、冒泡排序、希尔排序、快速排序四种常用的排序方法各自的特点和排序效率。以色相为比较依据，通过Canvas对像素进行操作把色彩的排序过程动态展示出来，给用户以直观的认识。

1. 总体设计

程序首先对预设的画布按照一定的规定划分出数量一定的色块，然后根据所分色块的数量以色相为衡量标准依次均匀地填充颜色。当画布初始化完以后，对所有色块进行随机打乱。用户可以选择插入排序、冒泡排序、希尔排序、快速排序中任意一种排序方法对乱序的色块进行排序，程序会实时展示排序的动态过程。用户可以直观的感受到不同排序方法的特点和效率。

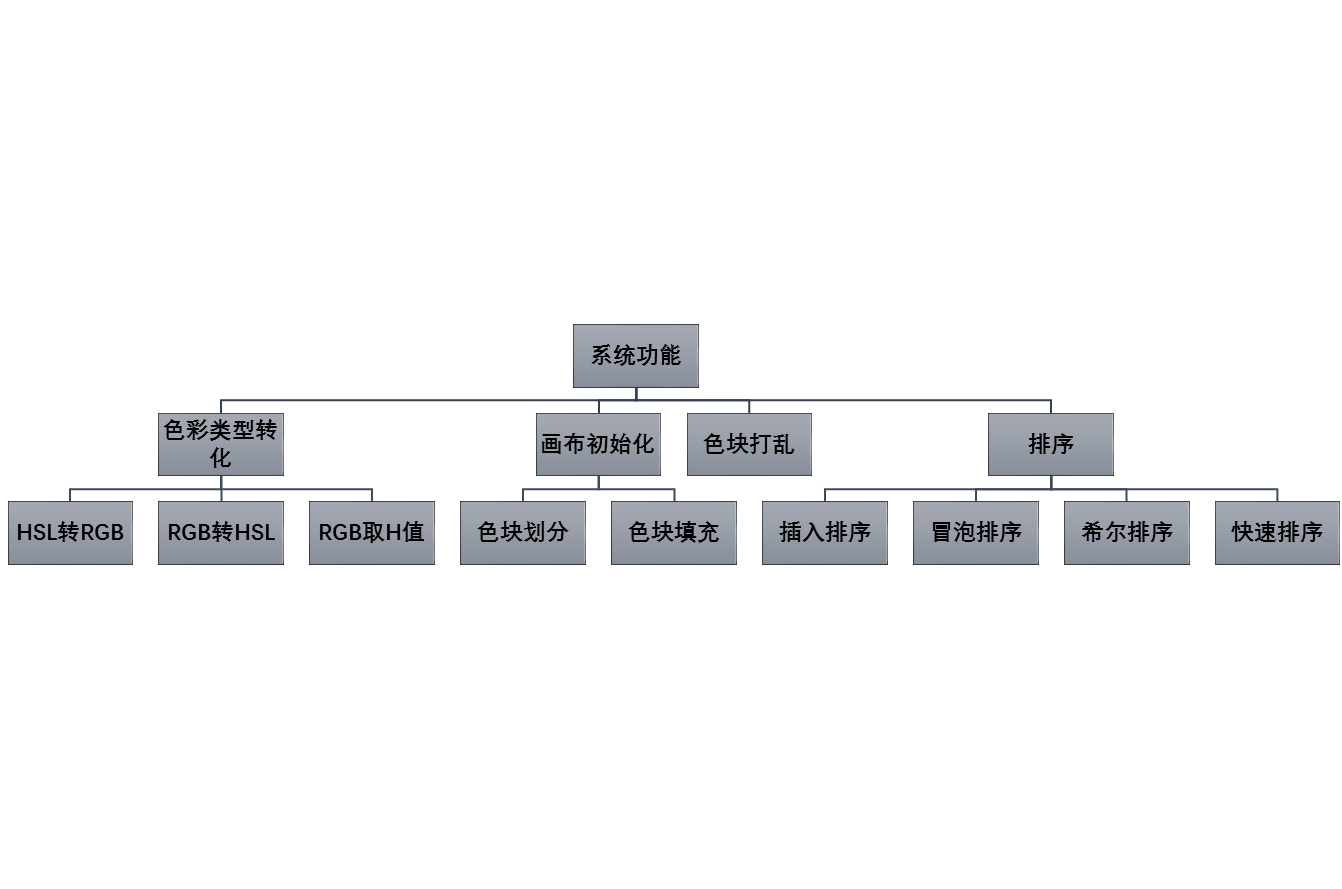


图2.1 功能关系图

1. 编程思想
   1. 色块的划分

因为不同排序方法的排序效率不同，所以处理的能够数据量是不同的，为了能够更好的展示并比较各种排序方法的排序方法的特点和排序效率，需要把画布划分为数量不同的色块。具体方式是把n\*n个像素点划分为一个色块。该n\*n个像素的颜色与“左上角”像素的颜色统一。根据这一思想我们就可以把长为width高为height的画布划分为(width \* height) / (n \* n)个色块，以方便根据排序方法排序效率的高低设置不同数量的色块进行排序展示。

* 1. 颜色的比较

我们常见的颜色标准为RGB色彩模式，并且在Canvas中ImageData的每个像素都储存着四个方面的信息，分别为R - 红色 (0-255)、G - 绿色 (0-255)、B - 蓝色 (0-255)、A - alpha 通道 (0-255; 0 是透明的，255 是完全可见的)。

因为每个像素有四个数值，显然不方便进行比较，所以我们应选择一个单一的衡量准则进行比较排序。在本程序中我们选择了HSL色彩模式这一工业界的一种颜色标准。HSL即是代表H: Hue 色相，S：Saturation 饱和度，L Lightness 明度三个通道的颜色。把其中的色相（H）来作为各颜色比较的衡量标准进行比较。

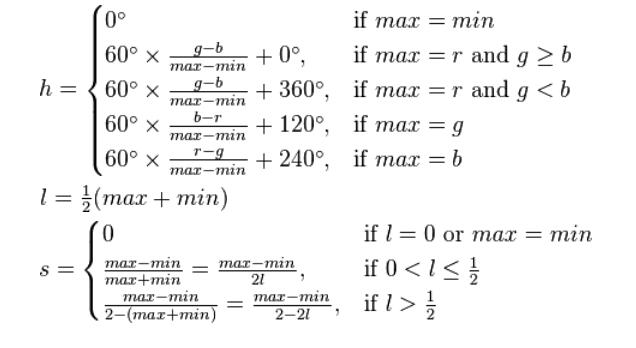
* 1. 像素操作

因为整张画布上的像素数量非常多，每一次都逐个转化后再操作无论是从代码编写还是程序运行方面来看都非常麻烦，所以在对颜色进行变化操作时编写了getColor和setColor这两个函数对每个色块中的n\*n个像素一同操作。简化排序中的编程难度，增强代码可读性的同时也可以在一定程度上提高程序的运行次效率。

1. 关键环节实现
   1. 颜色转换

4.1.1 RGB颜色转HSL

设 (r, g, b) 分别是一个颜色的红、绿和蓝坐标，它们的值是在 0 到 1 之间的实数。设 max 等价于 r, g 和 b 中的最大者。设 min 等于这些值中的最小者。要找到在 HSL 空间中的 (h, s, l) 值，这里的 h ∈ [0, 360）是角度的色相角，而 s, l ∈ [0,1] 是饱和度和亮度，计算为：

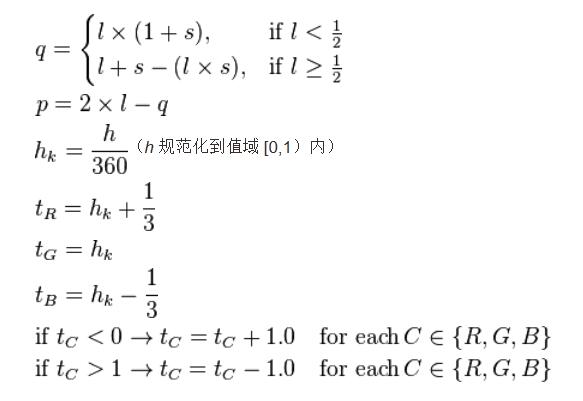


4.2.2 HSL颜色转RGB

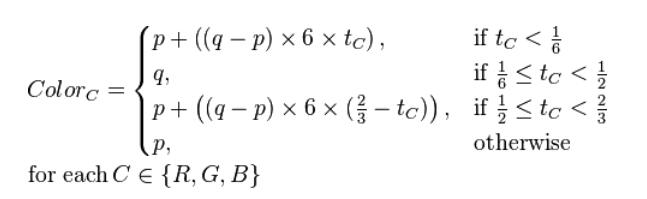
给定 HSL 空间中的 (h, s, l) 值定义的一个颜色，带有 h 在指示色相角度的值域 [0, 360）中，分别表示饱和度和亮度的s 和 l 在值域 [0, 1] 中，相应在 RGB 空间中的 (r, g, b) 三原色，带有分别对应于红色、绿色和蓝色的 r, g 和 b 也在值域 [0, 1] 中，它们可计算为：

首先，如果 s = 0，则结果的颜色是非彩色的、或灰色的。在这个特殊情况，r, g 和 b 都等于 l。注意 h 的值在这种情况下是未定义的。

当 s ≠ 0 的时候，可以使用下列过程：



对于每个颜色向量 Color = (ColorR, ColorG, ColorB) = (r, g, b),



* 1. 色块与对应像素转化

先把二维排列的色块“转化”为一维排列，通过每个色块边长所占像素的个数n 进一步转化为每个方块“左上角”像素的位置，最后根据每个像素储存四个方面的信息，找到目标像素的起始位置。

var row = Math.floor(index / (pixels.width / box));

var col = (index % (pixels.width / box));

row \*= box;

col \*= box;

var data = pixels.data;

var index = (row \* pixels.width + col) \* 4;

* 1. 色块颜色填充

先通过总的像素数目除以每一个色块所占像素数目求出当前画布划分出的色块数。

num=pixels.width \* pixels.height / Math.pow(box, 2)

然后把色相H对画块数目num经行平分，按照色块顺序依次递增按照色相H值进行填充。其中通道值A均默认为255 。

var rgb = hslToRgb(h, 1, 0.5);

h += step;

rgb.push(255);

setColor(pixels, i, rgb);

* 1. 色块打乱

依次对所有色块进行遍历，每当遍历到一个色块时用Math中的random函数生成一个大于0并且不大于色块数目num的一个随机整数作为另一个色块的位置，与当前色块进行颜色交换。这样遍历一遍后就可以确保整个画布上的色块是随机乱序的了。

randIndex = Math.floor(Math.random() \* num);

swapColor(pixels, i, randIndex);

1. 运行测试

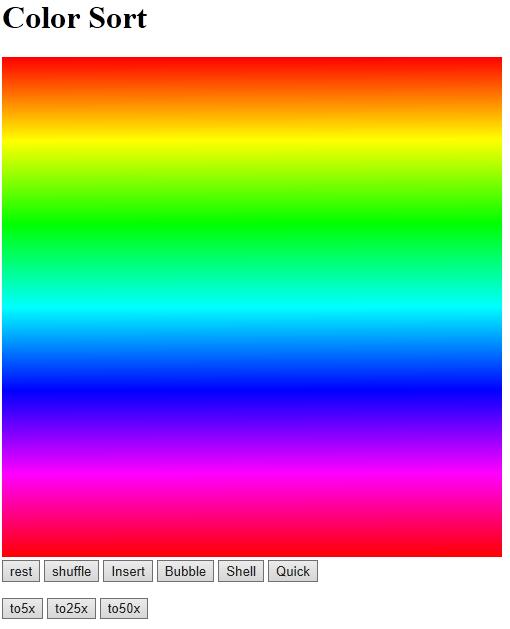


图5.1 程序初始化

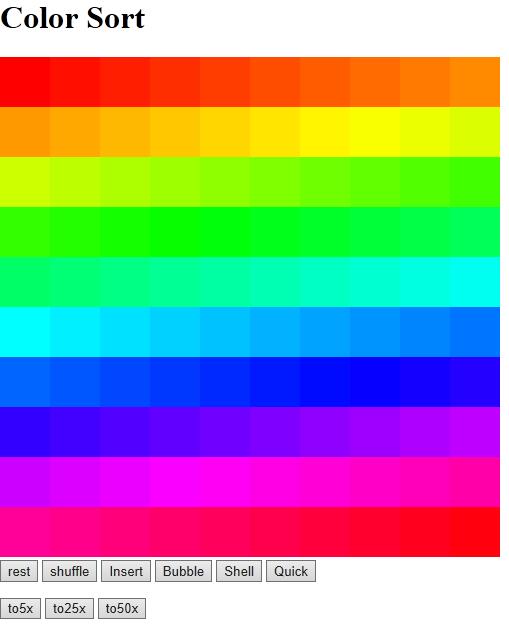


图5.2 50倍色块

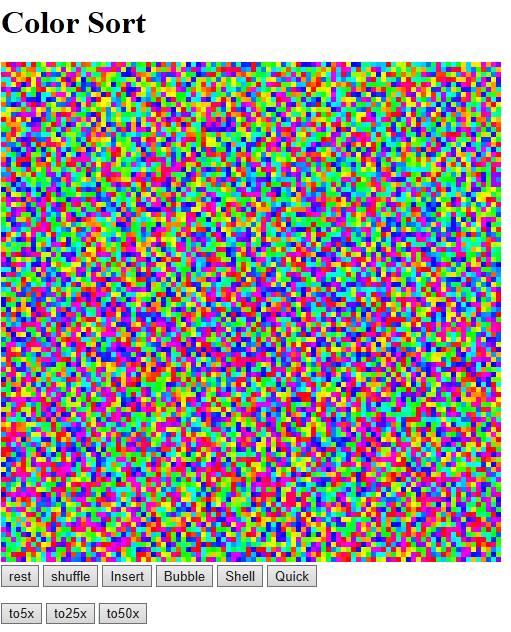


图5.3 5倍色块乱序

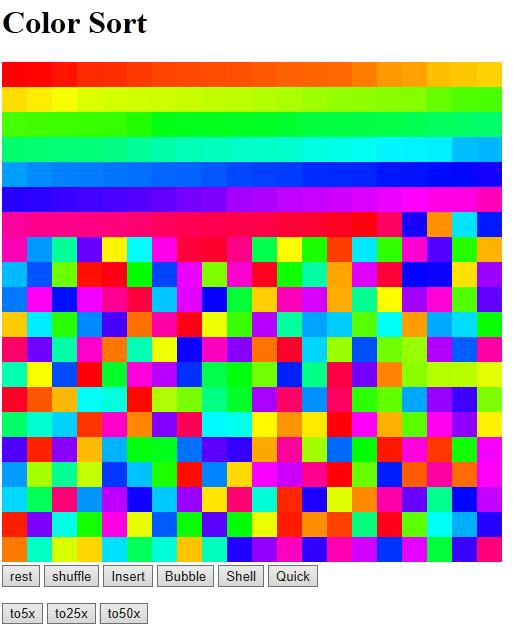


图5.4 25倍色块插入排序过程

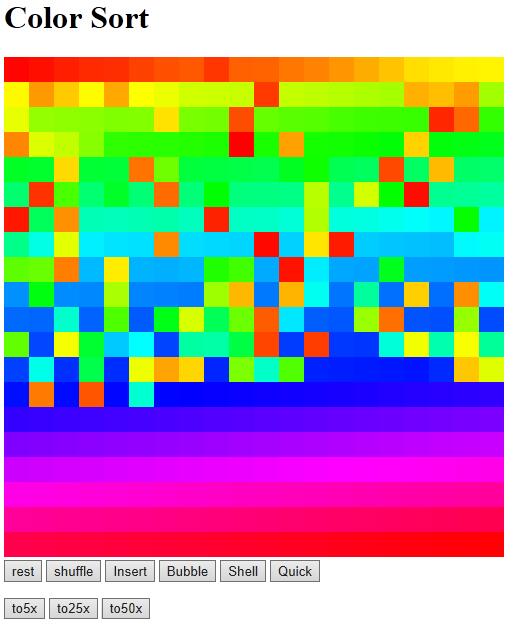


图5.5 25倍色块冒泡排序过程

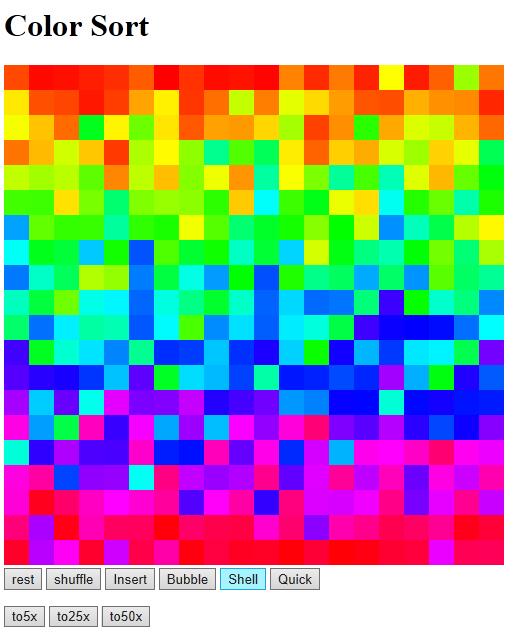


图5.5 25倍色块希尔排序过程

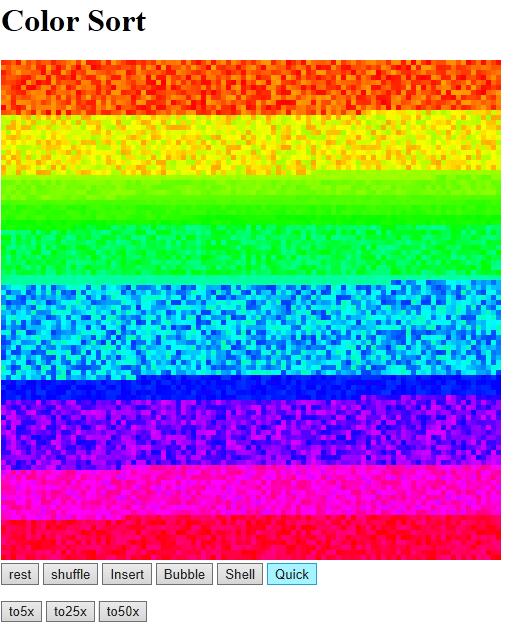


图5.6 5倍色块快速排序过程

1. 总结

插入排序：通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。但是在实现上，在从后向前扫描过程中，需要反复把已排序元素逐步向后挪位，为最新元素提供插入空间。

希尔排序：希尔排序，也称递减增量排序算法，是插入排序的一种高速而稳定的改进版本。在对几乎已经排好序的数据操作时，效率高。但插入排序一般来说是低效的，因为插入排序每次只能将数据移动一位。

冒泡排序：它重复地走访过要排序的元素，一次比较相邻两个元素，如果他们的顺序错误就把他们调换过来，直到没有元素再需要交换，排序完成。

快速排序：快速排序通常明显比其他Ο(n log n) 算法更快，在平均状况下，排序n个项目要Ο(n log n)次比较。在最坏状况下则需要Ο(n2)次比较，但这种状况并不常见。

**附录**：程序代码

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Cary\_ColorSort</title>

</head>

<body>

<h1>Color Sort</h1>

<canvas id="canvas" width="500" height="500"></canvas>

<div>

<form>

<div>

<input id="init" type="button" name="init" value="rest" />

<input id="shuffle" type="button" name="shuffle" value="shuffle" />

<input id="insert" type="button" name="insert" value="Insert" />

<input id="bubble" type="button" name="bubble" value="Bubble" />

<input id="shell" type="button" name="shell" value="Shell" />

<input id="quick" type="button" name="quick" value="Quick" />

</div>

<div><p></p></div>

<div>

<input id="to5x" type="button" name="to5x" value="to5x" />

<input id="to25x" type="button" name="to25x" value="to25x" />

<input id="to50x" type="button" name="to50x" value="to50x" />

</div>

</form>

</div>

<script>

var box = 1;

var cav = document.getElementById('canvas');

var context = cav.getContext('2d');

var pixels = context.getImageData(0, 0, cav.width, cav.height);

function hslToRgb(h, s, l){

var r, g, b;

if(s == 0){

r = g = b = l;

}else{

var hueToRgb = function hueToRgb(p, q, t){

if(t < 0) t += 1;

if(t > 1) t -= 1;

if(t < 1/6) return p + (q - p) \* 6 \* t;

if(t < 1/2) return q;

if(t < 2/3) return p + (q - p) \* (2/3 - t) \* 6;

return p;

}

var q = l < 0.5 ? l \* (1 + s) : l + s - l \* s;

var p = 2 \* l - q;

r = hueToRgb(p, q, h + 1/3);

g = hueToRgb(p, q, h);

b = hueToRgb(p, q, h - 1/3);

}

return [Math.round(r \* 255), Math.round(g \* 255), Math.round(b \* 255)];

}

function rgbToHsl(r, g, b){

r /= 255, g /= 255, b /= 255;

var max = Math.max(r, g, b), min = Math.min(r, g, b);

var h, s, l = (max + min) / 2;

if(max == min){

h = s = 0;

}else{

var d = max - min;

s = l > 0.5 ? d / (2 - max - min) : d / (max + min);

switch(max){

case r: h = (g - b) / d + (g < b ? 6 : 0); break;

case g: h = (b - r) / d + 2; break;

case b: h = (r - g) / d + 4; break;

}

h /= 6;

}

return [h, s, l];

}

function getH(rgb){

var hsl=rgbToHsl(rgb[0],rgb[1],rgb[2]);

return hsl[0];

}

function getColor(pixels, index){

if(typeof box != "number") {

box = 1;

}

// var row = (index % (pixels.height / box));

// var col = Math.floor(index / (pixels.height / box));

var row = Math.floor(index / (pixels.width / box));

var col = (index % (pixels.width / box));

row \*= box;

col \*= box;

var data = pixels.data;

var index = (row \* pixels.width + col) \* 4;

return [data[index], data[index + 1], data[index + 2], data[index + 3]];

}

function setColor(pixels, index, color){

if(typeof box != "number") {

box = 1;

}

// var row = (index % (pixels.height / box));

// var col = Math.floor(index / (pixels.height / box));

var row = Math.floor(index / (pixels.width / box));

var col = (index % (pixels.width / box));

row \*= box;

col \*= box;

var data = pixels.data;

var index = (row \* pixels.width + col)\*4;

var boxIndex;

for(var i = 0; i < box; ++ i)

{

for(var j = 0; j < box; ++ j)

{

boxIndex = index + (i \* pixels.width + j) \* 4;

data[boxIndex] = color[0];

data[boxIndex + 1] = color[1];

data[boxIndex + 2] = color[2];

data[boxIndex + 3] = color[3];

}

}

}

function swapColor(pixels, one, two){

var tempColor = getColor(pixels, one);

setColor(pixels, one, getColor(pixels, two));

setColor(pixels, two, tempColor);

}

function init(pixels){

var h = 0;

var step = 1 / pixels.width / pixels.height \* Math.pow(box, 2);

for(var i = 0, num = pixels.width \* pixels.height / Math.pow(box, 2); i < num; ++ i)

{

var rgb = hslToRgb(h, 1, 0.5);

h += step;

rgb.push(255);

setColor(pixels, i, rgb);

}

context.putImageData(pixels,0,0);

}

function shuffle(pixels){

var num = pixels.width \* pixels.height / Math.pow(box, 2);

var randIndex;

if(typeof box != "number"){

box = 1;

}

for(var i = 0; i < num; ++ i)

{

randIndex = Math.floor(Math.random() \* num);

swapColor(pixels, i, randIndex);

}

context.putImageData(pixels, 0, 0);

}

var insertSort=(function(pixels,beg,end){

var dt = new Date();

var cur = beg+1;

return function(){

if(cur==end){

alert("排序完成!");

context.putImageData(pixels,0,0);

for(var i=0;i!=this.elements.length;++i){

this.elements[i].style.backgroundColor="";

this.elements[i].disabled=false;

}

return;

}

var tempColor=getColor(pixels,cur);

var tempH=getH(tempColor);

var rgbJ=getColor(pixels,cur-1);

if(tempH<getH(rgbJ)){

for(var j=cur-1;j>=0;--j){

rgbJ=getColor(pixels,j);

if(tempH<getH(rgbJ)){

setColor(pixels,j+1,rgbJ);

}

else{

break;

}

}

setColor(pixels,j+1,tempColor);

}

++cur;

var loop=arguments.callee.bind(this);

var new\_date =new Date();

if(new\_date-dt>=0){

context.putImageData(pixels,0,0);

dt=new\_date;

}

requestAnimationFrame(loop);

}

});

var bubbleSort=(function(pixels,beg,end){

var dt = new Date();

var change=0;

var cur = end-1;

return function(){

if(cur<0||change!=0){

alert("排序完成!");

context.putImageData(pixels,0,0);

for(var i=0;i!=this.elements.length;++i){

this.elements[i].style.backgroundColor="";

this.elements[i].disabled=false;

}

return;

}

change=1;

var tempColor=getColor(pixels,cur);

var tempH=getH(tempColor);

for(var j=0;j<cur;++j){

if(getH(getColor(pixels,j))>getH(getColor(pixels,j+1))){

change=0;

swapColor(pixels,j,j+1);

}

}

--cur;

var loop=arguments.callee.bind(this);

var new\_date =new Date();

if(new\_date-dt>=40){

context.putImageData(pixels,0,0);

dt=new\_date;

}

requestAnimationFrame(loop);

}

});

var shellSort=(function(pixels,beg,end){

var dt = new Date();

var fraction = Math.floor(end/2);

return function(){

if(fraction<=0){

context.putImageData(pixels,0,0);

alert("排序完成!");

for(var i=0;i!=this.elements.length;++i){

this.elements[i].style.backgroundColor="";

this.elements[i].disabled=false;

}

return;

}

for(var j=fraction;j<end;++j){

for(var k=j-fraction;k>=0&&getH(getColor(pixels,k))>getH(getColor(pixels,fraction+k));k-=fraction){

swapColor(pixels,k,fraction+k);

}

}

fraction=Math.floor(fraction/2);

var loop=arguments.callee.bind(this);

var new\_date =new Date();

if(new\_date-dt>=40){

context.putImageData(pixels,0,0);

dt=new\_date;

}

requestAnimationFrame(loop);

}

});

var quickSort = (function(){

var dt = new Date();

var stack\_cnt=1;

function PARTITION(pixels,beg,end){

var i=beg-1;

var j=beg;

var m=Math.floor((beg+end)/2);

swapColor(pixels,m,end-1);

m=end-1;

var pivot=getH(getColor(pixels,m));

for(;j<m;++j){

if(getH(getColor(pixels,j))<pivot){

++i;

swapColor(pixels,i,j);

}

}

++i;

swapColor(pixels,i,m);

return i;

}

return function(pixels,beg,end){

if(end-beg<=1){

--stack\_cnt;

if(stack\_cnt==0)

{

alert("排序完成!");

context.putImageData(pixels,0,0);

for(var i=0;i!=this.elements.length;++i){

this.elements[i].style.backgroundColor="";

this.elements[i].disabled=false;

}

}

return;

}

var r=PARTITION(pixels,beg,end);

var new\_date=new Date();

if(new\_date -dt >= 40)

{

context.putImageData(pixels,0,0);

dt=new\_date;

}

var func = arguments.callee.bind(this);

var loop=function(){

func(pixels,beg,r);

func(pixels,r+1,end);

}

requestAnimationFrame(loop);

stack\_cnt+=2;

--stack\_cnt;

}

})();

init(pixels);

context.putImageData(pixels, 0, 0);

document.body.onload=function(){

var form = document.querySelector("form");

form.addEventListener("mouseup", function(event){

switch(event.target.tagName)

{

case "INPUT":

if(event.target.id == "init") {

init(pixels);

}

else if(event.target.id == "shuffle") {

shuffle(pixels);

}

else if(event.target.id == "insert"){

func = insertSort(pixels, 0, pixels.width \* pixels.height / Math.pow(box, 2));

func.call(this);

}

else if(event.target.id == "bubble"){

func = bubbleSort(pixels, 0, pixels.width \* pixels.height / Math.pow(box, 2));

func.call(this);

}

else if(event.target.id == "shell"){

func = shellSort(pixels, 0, pixels.width \* pixels.height / Math.pow(box, 2));

func.call(this);

}

else if(event.target.id == "quick"){

quickSort.call(this, pixels, 0, pixels.width \* pixels.height / Math.pow(box, 2));

}

else if(event.target.id == "to5x"){

box=5;

init(pixels);

}

else if(event.target.id == "to25x"){

box=25;

init(pixels);

}

else if(event.target.id == "to50x"){

box=50;

init(pixels);

}

break;

}

}, false)

}

</script>

</body>

</html>