PHP中的几种排序算法

# 开发环境

1、环境搭建：Windows 7+Apache 2.4.18+MySQL 5.7.11+PHP 7.1.0。

2、文本编辑器：Sublime 3。

# 二、主要技术

本案例主要使用PHP 7中的几种排序算法：快速排序、选择排序、插入排序、冒泡排序、归并排序来实现排序的功能。

# 三、效果图展示

效果图如图1所示：

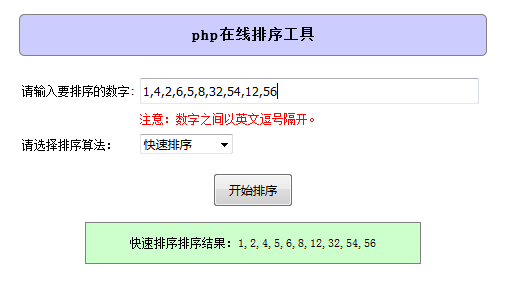


图1 效果图（快速排序结果）

# 四、具体步骤

## 4.1、界面设计

1. 表单设计

<!DOCTYPE html>

<html>

<head>

<title>php中的几种排序算法</title>

<meta charset="utf-8">

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<div id="divTitle"><b>php在线排序工具</b></div>

<form antion="index.php" method="post">

<table id="tbMain">

<tr>

<td>请输入要排序的数字: </td>

<td><input type="text" name="nums"

id="nums" class="txt"></td>

</tr>

<tr>

<td></td>

<td><span id="zy">注意：数字之间以英文

逗号隔开。</span></td>

</tr>

<tr>

<td>请选择排序算法：</td>

<td>

<select name="suanfa" id="suanfa" class="txt">

<option>快速排序</option>

<option>选择排序</option>

<option>插入排序</option>

<option>冒泡排序</option>

<option>归并排序</option>

</select>

</td>

</tr>

<tr>

<td colspan="2">

<center><input type="submit" name="sub"

value="开始排序" id="sub"></center>

</td>

</tr>

</table>

</form>

</body>

</html>

<?php

require("suanfa.php"); // 调用排序算法

?>

②样式设计（CSS样式）

/\*css样式\*/

body{

cellpadding:0;

cellspacing:0;

margin:0;

padding:0;

background-color:white;

font-size:18px;

font-family:'宋体';

}

#divTitle{

width:700px;

height:60px;

border:1px solid gray;

border-radius:8px;

background-color:#CCCCFF;

margin:30px auto;

text-align:center;

line-height:60px;

font-size:22px;

}

#tbMain{

width:700px;

margin:auto;

line-height:2em;

}

#nums{

width:500px;

}

#suanfa{

width:140px;

}

#sub{

width:120px;

height:50px;

font-size:18px;

margin-top:20px;

}

.txt{

font-size:18px;

height:30px;

}

#zy{

color:red;

}

#res{

margin: 20px auto;

width:500px;

height:60px;

border:1px solid gray;

text-align:center;

line-height:60px;

background-color:#ccffcc;

}

## 4.2、排序算法设计

### (1) 快速排序

排序思想：通过一趟排序将待排的记录分为两个独立的部分，其中一部分的记录的关键字均不大于另一部分记录的关键字，然后再分别对这两部分记录继续进行快速排序，以达到整个序列有序，具体做法需要每趟排序设置一个标准关键字和分别指向头一个记录的关键字和最后一个记录的关键字的指针。

function quickSort($arr) {

//判断是否继续进行

$length = count($arr);

if($length <= 1) {

return $arr;

}

//选择第一个数字作为基准

$base\_num = $arr[0];

//遍历除了基准外的所有数字，按照大小关系放入两个数组内，

之后初始化两个数组

$left\_array = array(); //小于基准

$right\_array = array(); //大于基准

for($i=1; $i<$length; $i++) {

if($base\_num > $arr[$i]) {

//放入左边数组

$left\_array[] = $arr[$i];

} else {

//放入右边数组

$right\_array[] = $arr[$i];

}

}

//分别对两数组进行相同的排序处理方式递归

$left\_array = quickSort($left\_array);

$right\_array = quickSort($right\_array);

//合并数组

return array\_merge($left\_array, array($base\_num),

$right\_array);

}

### (2) 选择排序

排序思想：选出最小的一个数字与第一个位置数字交换，之后再剩余的数当中再次找到最小的数字与第二个位置交换，依此循环到倒数第二个数字和最后一个数字比较结束为止。

function selectSort($arr) {

$len=count($arr);

for($i=0; $i<$len-1; $i++) {

// 假设最小值的位置

$p = $i;

for($j=$i+1; $j<$len; $j++) {

// $arr[$p] 已知的最小值

if($arr[$p] > $arr[$j]) {

//比较发现更小的记录下最小值的位置，并且在下次比较时

采用已知的最小值进行比较。

$p = $j;

}

}

//确定当前最小值的位置，保存到$p中。如果发现最小值的

位置与当前假设的位置$i不同，则位置互换即可。

if($p != $i) {

$tmp = $arr[$p];

$arr[$p] = $arr[$i];

$arr[$i] = $tmp;

}

}

return $arr;

}

### (3) 插入排序

排序思想：当前插入位置之前的元素有序，若插入当前位置的元素比有序元素最后一个元素大，则什么也不做，否则在有序序列中找到插入的位置，并插入。

function insertSort($arr) {

$len = count($arr);

for ($i = 1; $i < $len; $i++) {

// 当前值

$key = $arr[$i];

// 当前位置

$pos = $i;

// 如是当前位置 >0 && 当前值的前一个值 > 当前值 选出最值

while ($pos > 0 && $arr[$pos - 1] > $key) {

// 当前值 = 前一个值

$arr[$pos] = $arr[$pos - 1];

// 当前位置后移

$pos = $pos - 1;

}

// 找到当前值的位置

$arr[$pos] = $key;

}

return $arr;

}

### (4) 冒泡排序

排序思想：从前往后对相邻的两个数字依次进行比较调整，让较大的数字往下沉，让较小的数字往上升，即每相邻的数字进行对比排序，顺序不符合时将其调换位置。

function maopao($arr)

{

$len = count($arr);

for($i=1; $i<$len; $i++) //最多做n-1趟排序

{

$flag = false; //本趟排序开始前，交换标志应为假

for($j=$len-1;$j>=$i;$j--)

{

if($arr[$j]<$arr[$j-1]) //交换记录

{ //如果是从大到小的话，只要在这里的判断改

成if($arr[$j]>$arr[$j-1])就可以了

$x=$arr[$j];

$arr[$j]=$arr[$j-1];

$arr[$j-1]=$x;

$flag = true; //发生了交换，故将交换标志置为真

}

}

if(! $flag) //本趟排序未发生交换，提前终止算法

return $arr;

}

}

### (5) 归并排序

排序思想：采用“分而治之”的思想，把待排序的 n 个元素的序列分解成两个子序列, 每个子序列包括 n/2 个元素，然后对每个子序列分别调用归并排序al\_merge()函数, 进行递归操作，最后合并两个排好序的子序列,生成排序结果。

//al\_merge函数将指定的两个有序数组(arr1arr2,)合并并且排序。

function al\_merge($arrA,$arrB)

{

$arrC = array();

while(count($arrA) && count($arrB)){

//这里不断的判断哪个值小,就将小的值给到arrC,但是到

最后肯定要剩下几个值,不是剩下arrA里面的就是剩下

arrB里面的而且这几个有序的值,肯定比arrC里面所有

的值都大所以使用

$arrC[] = $arrA['0'] < $arrB['0'] ?

array\_shift($arrA) : array\_shift($arrB);

}

return array\_merge($arrC, $arrA, $arrB);

}

//归并排序主程序

function al\_merge\_sort($arr){

$len=count($arr);

if($len <= 1)

//递归结束条件,到达这步的时候,数组就只剩下一个元素了,

也就是分离了数组

return $arr;

$mid = intval($len/2);//取数组中间

//拆分数组0-mid这部分给左边left\_arr

$left\_arr = array\_slice($arr, 0, $mid);

$right\_arr = array\_slice($arr, $mid);//拆分数组

mid-末尾这部分给右边right\_arr

$left\_arr = al\_merge\_sort($left\_arr);//左边拆分完后

开始递归合并往上走

$right\_arr = al\_merge\_sort($right\_arr);//右边拆分

完毕开始递归往上走

$arr=al\_merge($left\_arr, $right\_arr);//合并两个数组,

继续递归

return $arr;

}

## 4.3、输出排序结果

if(isset($\_POST['sub'])){

// 获取用户输入

$nums=$\_POST['nums']; // 要排序的数字

$suanfa=$\_POST['suanfa']; // 选择的排序算法

// 判断输入是否为空

if(empty($nums)){

echo "<script>alert('请输入要进行排序的数字！

');</script>";

}else{

// 将字符串分割为字符串数组

$arr\_nums=explode(",",$nums);

switch($suanfa){

case '快速排序':

$result=quickSort($arr\_nums);

break;

case '选择排序':

$result=selectSort($arr\_nums);

break;

case '插入排序':

$result=insertSort($arr\_nums);

break;

case '冒泡排序':

$result=maopao($arr\_nums);

break;

case '归并排序':

$result=al\_merge\_sort($arr\_nums);

break;

}

// 输出排序结果

$res="";

for($i=0;$i<count($result);$i++){

$res.=$result[$i].",";

}

$str = substr($res,0,strlen($res)-1); // 去掉最后

一个字符（去掉最后的逗号）

echo "<div id='res'>".$suanfa."排序结

果：".$str."</div>";

}

}

# 五、网页测试

开启Apache服务器后，打开浏览器，在浏览器地址栏中输入文件地址进行测试。例如输入： “localhost:8080/cmfile/index.php”，可以看到程序运行结果如图2所示。

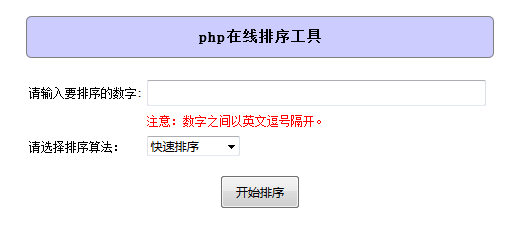


图2 案例结果图（排序操作界面）

在这里我们输入“1,4,2,6,5,8,32,54,12,56”来进行各种排序算法的测试，可以看到如下列所示的排序结果：

### (1) 快速排序测试结果

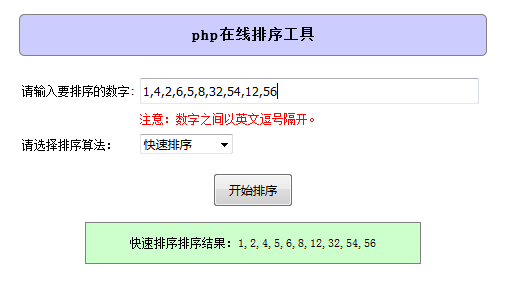


图3 案例结果图（快速排序）

### (2) 选择排序测试结果

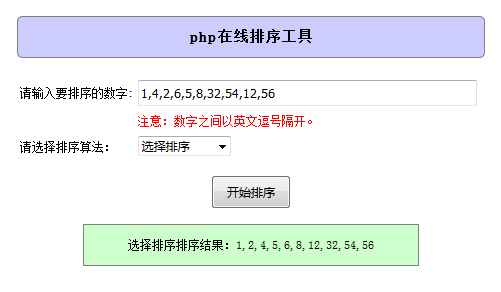


图4 案例结果图（选择排序）

### (3) 插入排序测试结果



图5 案例目结果图（插入排序）

### (4) 冒泡排序测试结果

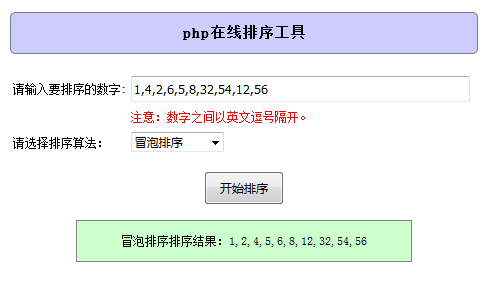


图6 案例结果图（冒泡排序）

### (5) 归并排序测试结果

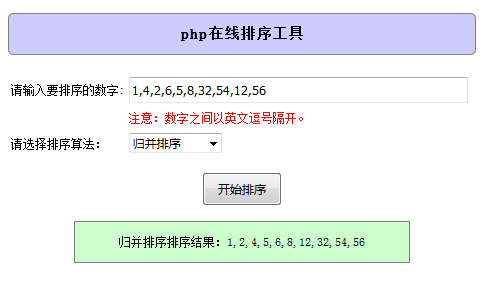


图7 案例结果图（归并排序）