常见排序算法及 PHP 实现

> 全文代码使用 PHP7.2 语法编写

0. 五种基础排序算法对比

排序方法	时间复杂度（平均）	时间复杂度（最坏）	时间复杂度（最好）	空间复杂度
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$
快速排序	$O(nlog_2n)$	$O(n^2)$	$O(nlog_2n)$	$O(nlog_2n)$
归并排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(n)$

1. 冒泡排序（Bubble Sort）

冒泡排序 是一种交换排序，它的基本思想是：对**待排序**记录从后往前（逆序）进行多遍扫描，当发现相邻两条记录的次序与排序要求的规则不符时，就将这两个记录进行交换。这样，值较小的记录将逐渐从后面向前移动，就像气泡在水中向上浮一样。

算法描述

假设需要排序的记录有 n 个，其值保存在数组 A 中，使用冒泡排序法，需对数组 A 进行 n-1 次扫描，完成排序操作。具体过程如下：

1. 将 A[n-1] 与 A[n] 进行比较,若 A[n] < A[n-1] ，则交换两元系的位置。
2. 修改数组下标，使需要比较的两个元素为 A[n-1] 和 A[n-2] ，重复步骤(1)，对这两个元素进行比较。重复这个过程，直到对 A[1] 和 A[0] 进行比较完为止。完成第 1 遍扫描。
3. 经过第 1 遍扫描后，最小的元素已经像气泡一样“浮”到最上面，即位于元素 A[0] 中了。接下来重复前面的步骤，进行第 2 遍扫描，只是扫描结束位置到 A[2] 与 A[1] 进行比较完为止（因为 A[0]中已经是最小的数据，不用再进行比较）。
4. 通过 n-1 遍扫描，前 n-1 个数都已经排序完成，最后一个元素 A[n] 肯定就是最大的数了。至此，完成排序操作。

代码实现

```
/**
 * 冒泡排序
 * @param array $arr
 */function bubbleSort(array &$arr) : void{
    $length = count($arr);

    // 外层循环，从数组首部开始，每完成一次循环，可确定 $arr[$i] 位置的元素
    for ($i = 0; $i < $length; $i++){

        // 内层循环，$j 从后往前循环
        for ($j = $length - 1; $j > $i; $j--) {

            // 若前面的值大于后面的值，则互换位置
            if ($arr[$j] < $arr[$j - 1]) {
                // 互换数组两个位置的值
                [$arr[$j], $arr[$j - 1]] = [$arr[$j - 1], $arr[$j]];
            }
        }
    }
}
```

2. 选择排序（Selection Sort）

选择排序是通过 `n-i` 次关键字间的比较，从 `n-i+1` 个记录中选出关键字最小的记录，并和第 `i ( 1 <= i <= n )` 个记录交换。

算法描述

1. 维护数组中最小的前 n 个元素的已排序序列。
2. 每次从剩余未排序的元素中选取最小的元素，将其放在已排序序列的后面，作为序列的第 n+1 个记元素。
3. 以空序列作为排序工作的开始，直到未排序的序列里只剩一个元素时（它必然为最大），只需直接将其放在已排序的记录之后，整个排序就完成了。

代码实现

```
/**
 * 选择排序
```

```
 * @param array $arr
 */function selectionSort(array &$arr) : void{
    $length = count($arr);

    // 外层循环，从数组首部开始，每完成一次循环，可确定一个元素的位置
    for ($i = 0; $i < $length - 1; $i++) {
        // 选定的最小值的索引
        $minIdx = $i;
        // 从 $i + 1 位开始循环，判断当前选定的元素是否是当次循环的最小值
        for ($j = $i + 1; $j < $length; $j++) {
            // 若出现比选定的值还小的值，则替换最小值的索引
            if ($arr[$minIdx] > $arr[$j]) {
                $minIdx = $j;
            }
        }
        // 互换数组两个位置的值
        [$arr[$i], $arr[$minIdx]] = [$arr[$minIdx], $arr[$i]];
    }
}
/**
 * 选择排序 - 方法 2
 * @param array $arr
 */function selectionSort2(array &$arr) : void{
    $length = count($arr);

    // 外层循环，从数组首部开始，每完成一次循环，依次确定数组元素的位置
    for ($i = 0; $i < $length; $i++) {
        // 从 $i + 1 位开始循环，依次判定 $arr[$i] 与 $arr[$j] 的大小
        for ($j = $i + 1; $j < $length; $j++) {
            // 若 $arr[$i] 比 $arr[$j] 大，则互换两个元素的位置
            if ($arr[$i] > $arr[$j]) {
                // 互换数组两个位置的值
                [$arr[$j], $arr[$i]] = [$arr[$i], $arr[$j]];
            }
        }
    }
}
```

3. 插入排序（Insertion Sort）

插入排序 是通过构建有序序列，从未排序数据中选择一个元素，在已排序序列中从后向前扫描，找到相应位置并插入。插入排序在从后向前扫描过程中，需要把已排序元素逐个向后移动，为最新元素提供插入空间。

每天进步一点点，让优秀成为习惯！--六星教育

算法描述

1. 对于第 1 个元素，因为没有比较，将其作为已经有序的序列。
2. 从数组中获取下一个元素，在已经排序的元素序列中从后向前扫描，并进行判断。
3. 若排序序列的元素大于新元素，则将该元素向后移动一位。
4. 重复步骤(3)，直到在已排序的元素中找到小于或者等于新元素的元素，将新元素插入到该元素的后面。
5. 重复步骤(2)~(4)，直到完成排序。

代码实现

```php
/**
 * 插入排序
 * @param array $arr
 */function insertionSort(array &$arr) : void{
    $length = count($arr);

    // 从数组首部开始排序，每完成一次循环，可确定一个元素的位置
    for ($i = 0; $i < $length - 1; $i++) {
        // 内层循环从 $i + 1 个元素开始，一位一位向前比较
        // 若前面的值比自己大，则替换，直到前面的值比自己小了，停止循环
        for ($j = $i + 1; $j > 0; $j--) {
            if ($arr[$j] >= $arr[$j - 1]) {
                break;
            }
            [[$arr[$j], $arr[$j - 1]]] = [[$arr[$j - 1], $arr[$j]]];
        }

    }
}
/**
 * 插入排序 - 方法 2
 * @param array $arr
 */function insertionSort2(array &$arr) : void{
    $length = count($arr);

    // 从数组首部开始排序，每完成一次循环，可确定一个元素的位置
    for ($i = 0; $i < $length - 1; $i++) {
        // 从第二个元素开始，选择固定位置的值作为基准值
        $currentVal = $arr[$i + 1];
        // 初始键位于选定值的前一个位置
        $preIdx = $i;
```

```
        // 拿基准值一步一步向前比较，直到基准值比前面的值小，则两值互换位置
        while ($preIdx >= 0 && $currentVal < $arr[$preIdx]) {
            $arr[$preIdx + 1] = $arr[$preIdx];
            $arr[$preIdx] = $currentVal;
            $preIdx--;
        }

    }
}
```

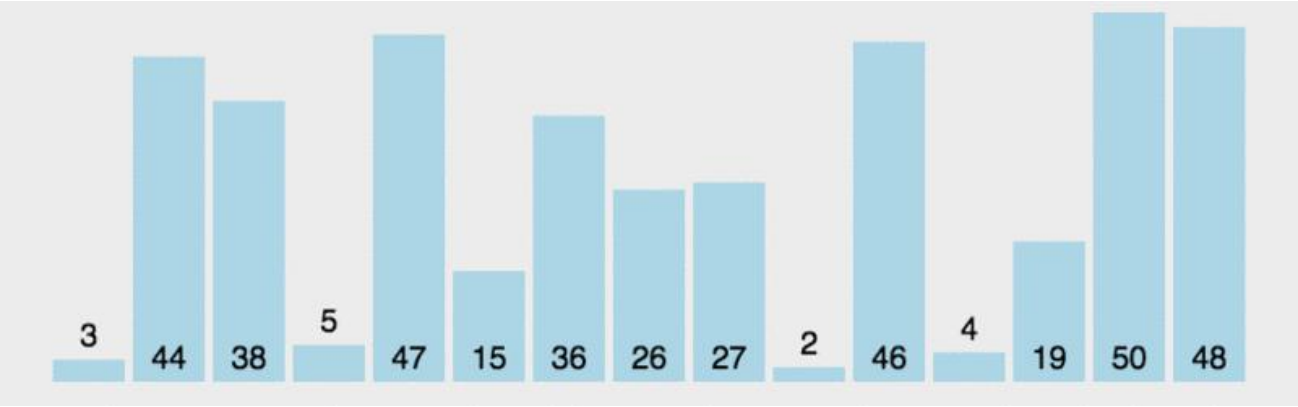
4. 快速排序（Quick Sort）

快速排序是通过一趟排序将待排记录分割成独立的两部分，其中一部分记录的关键字均比另一部分记录的关键字小，则可分别对这两部分记录继续进行排序，以达到整个序列有序的目的。

算法描述

快速排序使用分治策略来把待排序数据序列分为两个子序列，具体步骤如下：

1. 从数列中挑出一个元素，以该元素为“基准”。
2. 扫描一遍数列，将所有比“基准”小的元素排在基准前面，所有比“基准”大的元素排在基准后面。
3. 通过递归，将各子序列划分为更小的序列，直到把小于基准值元素的子数列和大于基谁值元素的子数列排序。



代码实现

```
/**
```

```php
 * 快速排序
 * @param $arr
 */function quickSort(& $arr) : void{
    $length = count($arr);

    // 若数组为空，则不需要运行
    if ($length <= 1) {
        return;
    }

    $middle = $arr[0]; // 选定一个中间值

    $left = []; // 接收小于中间值
    $right = [];// 接收大于中间值

    // 循环比较
    for ($i = 1; $i < $length; $i++) {

        if ($middle < $arr[$i]) {
            // 大于中间值
            $right[] = $arr[$i];
        } else {
            // 小于或等于中间值
            $left[] = $arr[$i];
        }
    }

    // 递归排序划分好的左右两边
    quickSort($left);
    quickSort($right);

    $arr = array_merge($left, [$middle], $right);
}
```

5. 归并排序（Merge Sort）

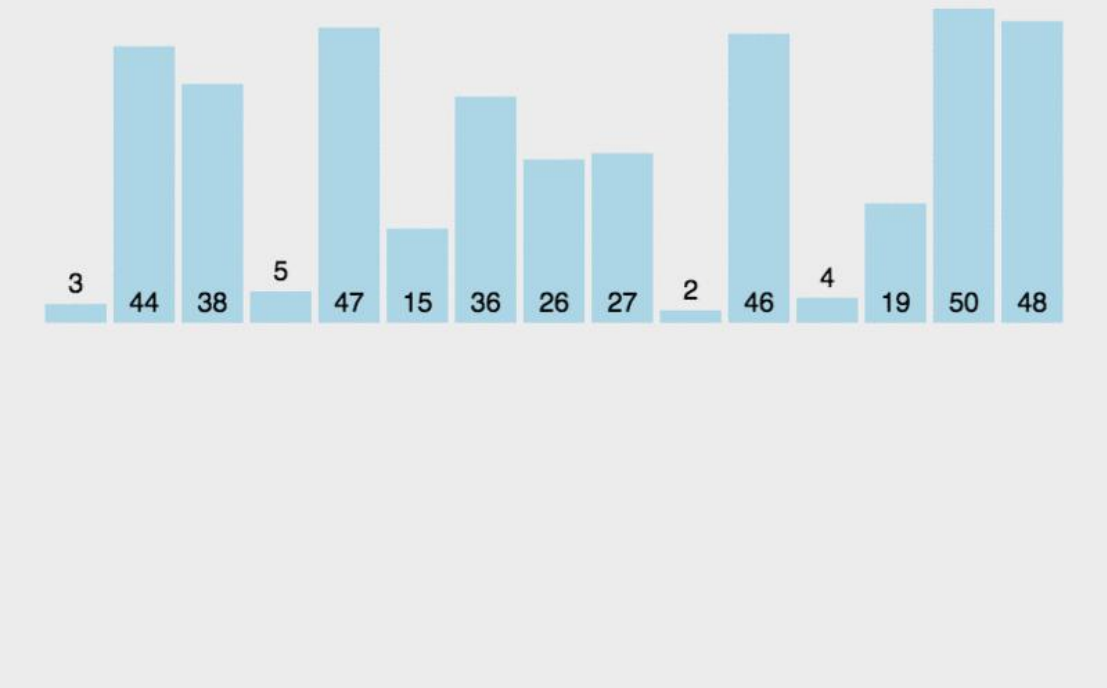
算法描述

归并是一种典型的序列操作，其工作是把两个或更多有序序列合并为一个有序序列。基于归并的思想也可以实现排序，称为归并排序。基本方法如下:

每天进步一点点，让优秀成为习惯！--六星教育

1. 初始时，把待排序序列中的 n 个元素看成 n 个有序子序列（因为只有 1 个元素的序列总是排好序的），每个子序列的长度均为 1。
2. 把序列组里的有序子序列两两归并，每完成一论归并，序列组里的序列个数减半，每个子序列的长度加倍。
3. 对加长的有序子序列重复上面的操作，最终得到一个长度为 n 的有序序列。

这种归并方法也称为简单的二路归并排序，其中每次操作都是把两个有序序列合并为一个有序序列。也可考虑三路归并或更多路的归并。



代码实现

```php
/**
 * 归并排序
 * @param array $arr
 * @return array
 */function mergeSort(array $arr){
    // 计算数组长度，若长度不大于 1，则不需要排序
    $length = count($arr);
    if ($length <= 1) {
        return $arr;
    }

    // 获取数组中间位置的索引
    $midIdx = floor($length / 2);
```

```
    // 把数组从中间拆分成左右两部分
    $left = mergeSort(array_slice($arr, 0, $midIdx));
    $right = mergeSort(array_slice($arr, $midIdx));

    // 合并两部分，同时进行排序
    return merge($left, $right);
}
/**
 * 合并数组，同时进行排序
 * @param array $left
 * @param array $right
 * @return array
 */function merge(array $left, array $right){
    // 分别计算左右两数组的长度
    $lLength = count($left);
    $rLength = count($right);

    // 左右两数组的索引
    $l = $r = 0;

    $lists = [];

    // 只有左右两数组都未遍历完成时，才有必要继续遍历
    // 当其中一个数组的元素遍历完成，说明另一个数组中未遍历过的值比遍历过的值都大
    while ($l < $lLength && $r < $rLength) {
        // 比较 $left[$l] 和 $right[$r]，取其中较小的值加入到 $lists 数组中
        if ($left[$l] < $right[$r]) {
            $lists[] = $left[$l];
            $l++;
        } else {
            $lists[] = $right[$r];
            $r++;
        }
    }

    // 合并 $lists 和 $left、$right 中剩余的元素
    return array_merge($lists, array_slice($left, $l), array_slice($right, $r));
}
/**
 * 合并数组，同时进行排序 - 方法 2
 * @param array $left
```

```
 * @param array $right
 * @return array
 */function merge2(array $left, array $right){
    // 分别计算左右两数组的长度
    $lLength = count($left);
    $rLength = count($right);

    // 左右两数组的索引
    $l = $r = 0;

    $lists = [];

    // 只有左右两数组都未遍历完成时，才有必要继续遍历
    // 当其中一个数组的元素遍历完成，说明另一个数组中未遍历过的值比遍历过的值都大
    while ($l < $lLength && $r < $rLength) {
        // 比较 $left[$l] 和 $right[$r]，取其中较小的值加入到 $lists 数组中
        if ($left[$l] < $right[$r]) {
            $lists[] = $left[$l];
            // PHP 中 unset 掉数组中的元素后，其他元素的键名不变
            unset($left[$l]);
            $l++;
        } else {
            $lists[] = $right[$r];
            unset($right[$r]);
            $r++;
        }
    }

    // 合并 $lists 和 $left、$right 中剩余的元素
    return array_merge($lists, $left, $right);
}
/**
 * 合并数组，同时进行排序 - 方法 3
 * @param array $left
 * @param array $right
 * @return array
 */function merge3(array $left, array $right){
    // 分别计算左右两数组的长度
    $lLength = count($left);
    $rLength = count($right);

    $lists = [];
```

```
    // 只有左右两数组都未遍历完成时，才有必要继续遍历
    // 当其中一个数组的元素遍历完成，说明另一个数组中未遍历过的值比遍历过的值都大
    while ($lLength > 0 && $rLength > 0) {
        // 比较 $left[$l] 和 $right[$r]，取其中较小的值加入到 $lists 数组中
        if ($left[0] < $right[0]) {
            $lists[] = $left[0];
            // PHP 中 unset 掉数组中的元素后，其他元素的键名不变
            unset($left[0]);
            // 重建数组索引，始终让比较的值在第一位
            $left = array_values($left);
            $lLength--;
        } else {
            $lists[] = $right[0];
            unset($right[0]);
            $right = array_values($right);
            $rLength--;
        }
    }

    // 合并 $lists 和 $left、$right 中剩余的元素
    return array_merge($lists, $left, $right);
}
```

每天进步一点点，让优秀成为习惯！--六星教育