PHP算法

全球变暖

目　录

二分查找

二分查找又称折半查找，优点是比较次数少，查找速度快，平均性能好;其缺点是要求待查表为有序表，且插入删除困难。
因此，折半查找方法适用于不经常变动而查找频繁的有序列表。
首先，假设表中元素是按升序排列，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功;
否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。
重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

```
<?php
$arr = array(1,5,9,15,50,65);

/**
 * $arr 待查找的元素数组
 * $low 开始元素的下标
 * $hign 结束元素的下标
 * $k 待查找的元素
 * @return 查找元素的下标
 */
function bS($arr,$low,$hign,$k){
    if ($low <= $hign){
        $mid = intval(($low +$hign)/2);//计算中间的元素下标
        if ($arr[$mid] == $k){//如果相等
            return $mid;
        } else if ($k < $arr[$mid]){//元素下标在前面一部分
            return bS($arr, $low, $mid-1, $k);
        } else {//元素下标在后面一部分
            return bS($arr, $mid+1, $hign, $k);
        }
    } else {
        return -1;
    }
}

var_dump(bS($arr, 0, count($arr), '9'));
```

顺序查找

顺序查找是在一个已知无(或有序)序队列中找出与给定关键字相同的数的具体位置。原理是让关键字与队列中的数从最后一个开始逐个比较，直到找出与给定关键字相同的数为止，它的缺点是效率低下。

```
<?php
$arr = array(12,26,35,84,41,2);
/**
 * @param unknown $arr 待查找元素数组
 * @param unknown $k 待查找元素
 */
function iotd($arr,$k){
    foreach ($arr as $key=>$val){
        if ($k === $val){
            return $key;
        }
    }
    return -1;
}

var_dump(iotd($arr, 35));
```

线性表

线性表(linear list)是数据结构的一种，一个线性表是n个具有相同特性的数据元素的有限序列。数据元素是一个抽象的符号，其具体含义在不同的情况下一般不同。

在稍复杂的线性表中，一个数据元素可由多个数据项(item)组成，此种情况下常把数据元素称为记录(record)，含有大量记录的线性表又称文件(file)。

线性表中的个数n定义为线性表的长度，n=0时称为空表。在非空表中每个数据元素都有一个确定的位置，如用ai表示数据元素，则i称为数据元素ai在线性表中的位序。

线性表的相邻元素之间存在着序偶关系。如用(a1，…，ai-1，ai，ai+1，…，an)表示一个顺序表，则表中ai-1领先于ai，ai领先于ai+1，称ai-1是ai的直接前驱元素，ai+1是ai的直接后继元素。当i=1,2，…，n-1时，ai有且仅有一个直接后继，当i=2，3，…，n时，ai有且仅有一个直接前驱 。

```
<?php
$arr = array(1,5,9,8,2);
/**
 * 删除最后一个元素并返回
 * @param unknown $arr
 */
function pop_ll(&$arr){
    return array_pop($arr);
}

var_dump(pop_ll($arr));

var_dump($arr);
```

```
<?php
$arr = array(1,5,9,8,2);
/**
 * 删除最后一个元素并返回
 * @param unknown $arr
 */
function pop_ll(&$arr){
    return array_pop($arr);
}
/**
 * 添加一个元素至数组前面
 * @param unknown $arr
 * @return mixed
 */
```

```
function shift_ll(&$arr,$ele){
    return array_unshift($arr,$ele);
}

var_dump(pop_ll($arr));

var_dump(shift_ll($arr,100));

var_dump($arr);
```

冒泡排序

冒泡排序（Bubble Sort，台湾译为：泡沫排序或气泡排序）是一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

这个算法的名字由来是因为越大的元素会经由交换慢慢“浮”到数列的顶端，故名。

```
<?php
$arr = array(12,5,63,54,144);

function bubbleSort(&$arr){
    $len = count($arr);//计算长度
    for($i=0;$i<$len;$i++){
        for ($j=$i+1;$j<$len;$j++){
            if ($arr[$j] < $arr[$i]){//如果后面的元素小于前面的元素，则进行交换
                $tmp = $arr[$j];
                $arr[$j] = $arr[$i];
                $arr[$i] = $tmp;
            }
        }
    }
}

bubbleSort($arr);

var_dump($arr);
```

快速排序

快速排序(Quicksort)是对冒泡排序的一种改进。

快速排序由C. A. R. Hoare在1962年提出。它的基本思想是:通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

```
<?php
$arr = array(
    12,
    5,
    63,
    54,
    144
);

function quickSort($arr)
{
    if (count($arr) <= 1) {
        return $arr;
    }
    $key = $arr[0]; // 第一个元素
    $left_arr = array(); // 左边数组
    $right_arr = array(); // 右边数组
    for ($i = 1; $i < count($arr); $i ++) {//从第2个元素开始遍历
        if ($arr[$i] <= $key)//小于第一个元素的，放在左边数组
            $left_arr[] = $arr[$i];
        else
            $right_arr[] = $arr[$i];//大于第一个元素的，放在右边
    }
    $left_arr = quickSort($left_arr);//对左边的数组排序
    $right_arr = quickSort($right_arr);//对右边的数组排序
    return array_merge($left_arr, array(
        $key
    ), $right_arr);//合并数组
}

var_dump(quickSort($arr));
```

约瑟夫环问题

约瑟夫环(约瑟夫问题)是一个数学的应用问题:已知n个人(以编号1，2，3...n分别表示)围坐在一张圆桌周围。从编号为k的人开始报数，数到m的那个人出列;他的下一个人又从1开始报数，数到m的那个人又出列;依此规律重复下去，直到圆桌周围的人全部出列。通常解决这类问题时我们把编号从0~n-1，最后 结果+1即为原问题的解。

```
<?php

/**
 * 返回最后一个人编号
 * @param unknown $n 已知人数
 * @param unknown $m 间隔数
 * @return mixed
 */
function king($n, $m)
{
    // 生成数组
    $monkey = range(1, $n);
    $i = 0;
    while (count($monkey) > 1) {
        $i += 1; // 从第一个开始
                 // 将第一个人出环
        $head = array_shift($monkey);
        if ($i % $m != 0) {
            // 将第一个人放到尾部
            array_push($monkey, $head);
        } else {
            // 这个人出环了
            //echo $head . '<br>';
        }
    }
    return $monkey[0];
}
echo king(10, 7);
```