个人资料

PlusPlus1

访问： 176334次

积分： 2410

等级： BLOG 5

排名： 第9452名

原创： 55篇 转载： 2篇

译文： 1篇 评论： 65条

算法——寻找两个有序数组的中值

2014-04-12 12:48 1806人阅读 评论(3) 收藏 举报

分类： Algorithm（13）

1. 算法描述

有两个数组 A 和 B，均为有序排列，A的长度为m，B的长度为n，求 A 和 B 合在一起后的中值.

2. 问题分析

- 这里要注意一下：要充分利用 **A**和**B**均为有序的特性
- 该问题进一步可转化为求**A**和**B**的任意**K**值，如三分位、四分位.

思路一：将**A**和**B**合并成新的数组

[java] view plain copy print ?

```
/**
     * 合并有序数组，然后寻找k值
     *
     * @param a
     *            有序数组a
     * @param b
     *            有序数组b
     * @param k
     *            k值位置，0<=k<=a.length+b.length-1
     * @return k值
     */
    public static int findKthByMerge(int[] a, int[] b, int k) {
        System.out.println("Find kth by merge array first");
        int[] ab = new int[a.length + b.length];
        int ai = 0, bi = 0, abi = 0;
        while (ai < a.length && bi < b.length) {
            ab[abi++] = (a[ai] < b[bi]) ? a[ai++] : b[bi++];
        }
        while (ai < a.length) {
            ab[abi++] = a[ai++];
        }
        while (bi < b.length) {
            ab[abi++] = b[bi++];
        }
        System.out.println(Arrays.toString(ab));

        return ab[k];
    }
```

这种方法最容易想到，合并成有序数组后即可求任意k值，其时间复杂度为 **O**（**m+n**）， 空间复杂图为

(12758)
Tomcat7中WebSocket初
(11380)
Android图片处理：识别
(10670)
解决 Android 中使用List\
(10223)
怪胎：Android开发Imag (9315)
使用JDBC在MySQL数据 (9227)
Android 4主线程访问网络 (8748)
硬盘 IDE 切换到 AHCI (7165)
Linux下搭建Android开发 (5917)
平衡二叉树 之 AVL树 (5333)

评论排行

Android 真机连接本地PC (13)
解决 Android 中使用List\ (10)
平衡二叉树 之 AVL树 (8)
Android图片处理：识别 (5)
怪胎：Android开发Imag (5)
Linux下搭建Android开发 (3)
算法——寻找两个有序数 (3)
排序算法之归并排序 (2)
使用JDBC在MySQL数据 (2)
Android 4主线程访问网络 (2)

推荐文章

*Networking Named Content 全文翻译
* 边缘检测与图像分割
* 数据库性能优化之SQL语句优化
*阿里巴巴发布《2015移动安全漏洞年报》
* Java经典设计模式之七大结构型模式（附实例和详解）
*网络性能评价方法

最新评论

使用JDBC在MySQL数据库中快i
ahlly88: 学习了，很有用。这个方法我加100万条数据到mysql“共用去时间65161”，也就是一分钟6秒哦。
平衡二叉树 之 AVL树
qq_33933674: 代码有问题。请问楼主的RL，LR情况有测试过吗？
瀑布流布局神器——JQuery Mas
qq_33410364: 根据规划局
平衡二叉树 之 AVL树
火灵: 大师，AVL全称是什么啊？
算法——寻找两个有序数组的中
wangkai19952008: 思路很好，程序有问题
使用JDBC在MySQL数据库中快i
手一挥维生素: 我插入七万条数据用了297秒，连接字符串加了上边的字串后，插入同样数量的数据，时间变成了337秒。但...
平衡二叉树 之 AVL树
pan_nainai: thanks
Linux下搭建Android开发环境及A
mkh10086: 非常感谢，我的问题终于解决了
平衡二叉树 之 AVL树
王虹芸: 那RR和RL呢？？？
算法——寻找两个有序数组的中
小斯_wjc: k = 2测试结果是死循环。

O（m+n)

这里反思一下：真的需要合并数组吗？

思路二：采用扫描计数方法

```
[java] view plain  copy  print  ?
01. /**
02.      * 无需合并数组，利用计数机寻找K值
03.      *
04.      * @param a
05.      *            有序数组a
06.      * @param b
07.      *            有序数组b
08.      * @param k
09.      *            k值位置，0<=k<=a.length+b.length-1，k同时充当计数器
10.      * @return k值
11.      */
12.     public static int findKthByCounter(int[] a, int[] b, int k) {
13.         System.out.println("Find kth by counter");
14.         int ai = 0, bi = 0;
15.         int kth = 0; // 保存K值
16.         while (ai < a.length && bi < b.length && k >= 0) {
17.             kth = (a[ai] < b[bi]) ? a[ai++] : b[bi++];
18.             k--;
19.         }
20.         while (ai < a.length && k >= 0) {
21.             kth = a[ai++];
22.             k--;
23.         }
24.         while (bi < b.length && k >= 0) {
25.             kth = b[bi++];
26.             k--;
27.         }
28.         return kth;
29.     }
```

本算法是对算法一的改进，用一个临时变量保存K值，而不需要讲新合并的数组单独存储，节省了存储空间。其 时间复杂度为**O（m+n),** 空间复杂度为**O(1).**

到此都是线性时间复杂度，已经是非常高效了，但又没有更加高效的方法进一步降低时间复杂度呢？
这里注意到原数组有序特性，利用二分特性可以将复杂度降至对数级别。

思路三：递归二分

```
[java] view plain  copy  print  ?
01. /**
02.      * 递归二分查找K值
03.      *
04.      * @param a
05.      *            有序数组a
06.      * @param b
07.      *            有序数组b
08.      * @param k
09.      *            K值位置，0<=k<=m+n-1
10.      * @param aStart
11.      *            数组a初始查找位置
12.      * @param aEnd
13.      *            数组a结束查找位置
14.      * @param bStart
15.      *            数组b初始查找位置
16.      * @param bEnd
17.      *            数组b结束查找位置
18.      * @return k值
19.      */
20.     public static int findKth(int a[], int b[], int k, int aStart, int aEnd,
21.             int bStart, int bEnd) {
22.
23.         int aLen = aEnd - aStart + 1;
24.         int bLen = bEnd - bStart + 1;
25.
26.         // 递归结束条件
27.         if (aLen == 0) {
28.             return b[bStart + k];
```

```
        }
        if (bLen == 0) {
            return a[aStart + k];
        }
        if (k == 0) {
            return a[aStart] < b[bStart] ? a[aStart] : b[bStart];
        }

        // 将k按比例分配到a和b中，（k+1）=ka+kb,
        int ka = (k + 1) * aLen / (aLen + bLen);
        int kb = (k + 1) - ka;
        ka += aStart;
        kb += bStart;

        // 因为a和b有序，aStart-ka , bStart-kb yi
<span style="white-space:pre">      </span>// 最大值进行比较
        if (a[ka] > b[kb]) {
            k = k - (kb - bStart); // bStart - kb 这段应当排除，调整k值
            aEnd = ka; // 新k值可能存在于 aStart - ka
            bStart = kb; // 新k值可能存在于 kb - bEnd 之间
        } else {
            k = k - (ka - aStart);
            bEnd = kb;
            aStart = ka;
        }
        return findKth(a, b, k, aStart, aEnd, bStart, bEnd);
    }
```

本方法计算中值每次将范围缩小一半，故而 其 时间复杂度为 **lg（m+n）.**

3. 测试算法

[java] view plain copy print ?

```
public static void main(String[] args) {
        int A[] = { 0, 10, 30, 40, 50, 80, 89, 99, 101 };
        // int A[]={};
        int B[] = { -1, 33, 36, 56, 80, 83, 97, 98, 200 };
        // int B[] = {};
        int k = 0;
        int kth = 0;

        k = (A.length + B.length - 1) / 2;
        System.out.println("A.length=" + A.length + "\t" + Arrays.toString(A));
        System.out.println("B.length=" + B.length + "\t" + Arrays.toString(B));
        System.out.println("k-index = " + k);

        kth = findKthByMerge(A, B, k);
        System.out.println(kth);

        kth = findKthByCounter(A, B, k);
        System.out.println(kth);

        System.out.println("递归查找");
        kth = findKth(A, B, k, 0, A.length - 1, 0, B.length - 1);
        System.out.println(kth);
    }
```

输出结果如下：

[plain] view plain copy print ?

```
A.length=9  [0, 10, 30, 40, 50, 80, 89, 99, 101]
B.length=9  [-1, 33, 36, 56, 80, 83, 97, 98, 200]
k-index = 8
Find kth by merge array first
[-1, 0, 10, 30, 33, 36, 40, 50, 56, 80, 80, 83, 89, 97, 98, 99, 101, 200]
56
Find kth by counter
56
递归查找
56
```

顶 踩
7 0

上一篇 统计MySQL数据表大小
下一篇 从EXIF JPEG图片中提取GPS位置信息

我的同类文章

Algorithm（13）

- 平衡二叉树 之 AVL树 2013-12-12 阅读 5314
- 平衡二叉树 之 红黑树 2013-12-11 阅读 4241
- 查找算法 之 二叉查找树 2013-12-04 阅读 2830
- 二叉树遍历：递归+非递归... 2013-12-03 阅读 1072
- 树以及树的遍历和搜索 2013-11-12 阅读 783
- 排序算法之计数排序 2013-11-07 阅读 596
- 排序算法之归并排序 2013-11-07 阅读 1357
- 排序算法之快速排序 2013-11-06 阅读 701
- 排序算发之选择排序 2013-11-06 阅读 671

更多文章

猜你在找

有趣的算法（数据结构）
Ceph—分布式存储系统的另一个选择
Web无障碍——测试与工具
数据结构和算法
C语言在嵌入式开发中的应用

查看评论

3楼 wangkai19952008 2015-11-25 21:38发表


思路很好，程序有问题

2楼 小斯_wjc 2015-10-09 23:21发表


k = 2
测试结果是死循环。

1楼 小斯_wjc 2015-10-09 23:20发表


第3种方法，递归2分，有bug.
测试数据
A = {1,3,5};
B = {2,4,6}

发表评论

用 户 名： zhuqiuhui
评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap