位图排序

原创

shangzhihaohao 于 2015-04-12 16:25:56 发布
分类专栏： 算法 java


算法 同时被 2 个专栏收录
15 篇文章 0 订阅
订阅专栏


java
12 篇文章 0 订阅
订阅专栏

基于比较的排序时间复杂度至少为O(nlgn)，在时间上堆排序和快速排序基本上都达到了比较排序的极限，如果要获取更快的排序速度，就需要找不是基于比较的排序方法，位图排序就是其中的一个。

位图排序是效率最高的排序算法，其时间复杂度是O(n)，空间复杂度也非常小，但是要求输入的数据不能重复，而且要知道数据的范围。

位图排序的思想比较简单，用计算机的每一位表示一个数，一个int类型的变量就能表示32个数。比如说集合{1,2,5,7}能用二进制串01010110表示，因为这个串中的第1,2,5,7位为1（低位在右端），其他位为0。

利用这种思想我们把一个数组看成一个二进制串，每输入一个数将其所对应的位置1，输出时依次判断每一位，如果是1，就输出对应的数据。位图排序的关键子程序是置位程序和测试程序，分别对应输入和输出。

下边是位图排序的java实现：

```
public class BitSort {
    static int WORDLENGTH = 32;
    static int SHIFT = 5;
    static int MASK = 0x1F;
    static int MAX = 10000000;
    static int[] A = new int[(1 + MAX / WORDLENGTH)];

    public static void main(String[] args) {
        int[] data = {568746,12354,45798,1245,8,17,1,5};
        bitsort(data);
    }

    public static void bitsort(int[] array) {
        for (int i = 0; i < array.length; i++)
            set(array[i]);
        for (int i = 0; i < MAX; i++)
            if (test(i))
                System.out.println(i);
    }

    //将A[i>>SHIFT]的第(i & MASK)位置1
    public static void set(int i) {
        A[i >> SHIFT] |= (1 << (i & MASK));
    }

    //测试A[i>>SHIFT]的第(i & MASK)位置是否为1
    public static boolean test(int i) {
        return (A[i >> SHIFT] & (1 << (i & MASK)))
                == (1 << (i & MASK));
    }
}
```

在java中还可以使用bitset很容易的实现位图排序：

```
import java.util.BitSet;

public class BitSort {
    static int MAX = 10000000;

    public static void main(String[] args) {
        int[] data = { 5746, 14, 498, 125, 8, 17, 1, 5 };
        BitSet bitSet = new BitSet(MAX);
        for (int i : data) {
            bitSet.set(i);
        }
        for (int i = 0; i < MAX; i++) {
            if(bitSet.get(i)){
                System.out.println(i);
            }
        }
    }
}
```