基数排序

基数排序（radix sort）属于“分配式排序”（distribution sort），又称“桶子法”（bucket sort）或bin sort，顾名思义，它是透过键值的部份资讯，将要排序的元素分配至某些“桶”中，藉以达到排序的作用，基数排序法是属于稳定性的排序，其时间复杂度为O (nlog(r)m)，其中r为所采取的基数，而m为堆数，在某些时候，基数排序法的效率高于其它的稳定性排序法。

经典的牺牲空间获得时间的排序

排序思路

将所有待比较数值统一为同样的数位长度，数位较短的数前面补零。然后，从最低位开始，依次进行一次排序。这样从最低位排序一直到最高位排序完成以后，数列就变成一个有序序列

排序的循环次数等于数组里数字的最高位数

基数排序说明:

1)基数排序是对传统桶排序的扩展，速度很快.

2)基数排序是经典的空间换时间的方式，占用内存很大，当对海量数据排序时，容易造成OutOfMemoryError 。

3)基数排序时稳定的。[注:假定在待排序的记录序列中，存在多个具有相同的关键字的记录，若经过排序，这些记录的相对次序保持不变，即在原序列中，r[i]=r[j]，且r[i]在r[j]之前，而在

排序后的序列中，r[i]仍在r[j]之前，则称这种排序算法是稳定的；否则称为不稳定的]

代码实现:

```
package Sort排序;

import java.util.Arrays;
import java.util.Date;
import java.util.Random;

/**
 * @author jndeng
 * @create 2019-12-26 17:00
 */
public class RadixSort {
    public static void main(String[] args) {
        int[] array = new int[8000000];
      for (int i = 0; i < array.length; i++) {
         array[i] = (int) (Math.random() * 8000000);
      }
        Date date = new Date();
        long time = date.getTime();

        radixSort(array);
        Date date1 = new Date();
        long time1 = date1.getTime();
        System.out.println("基数排序:");
        System.out.println(time1 - time);
    }

    public static void radixSort(int[] arr) {
        int[][] bucket = new int[10][arr.length];
        //记录每个桶放了几个数据
        int[] temp = new int[10];


        int max = arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }
        int maxlength=0;
        while (max  > 0) {
            maxlength++;
```

```
            max /= 10;
        }
        for (int j = 0; j < maxlength; j++) {
            for (int i = 0; i < arr.length; i++) {
                int a1 = arr[i]/(int)Math.pow(10,j) % 10;
                bucket[a1][temp[a1]++] = arr[i];
            }
            int t = 0;
            for (int i = 0; i < temp.length; i++) {
                int t1 = 0;
                while (t1 < temp[i]) {
                    arr[t] = bucket[i][t1];
                    t1++;
                    t++;
                }
                temp[i]=0;
            }
        }


    }
}
```