链滴

常见排序算法

作者：ellenbboe

原文链接：https://ld246.com/article/1565144352929

来源网站：链滴

冒泡排序

```
    public int[] bubbleSort(int[] a){
        if(a == null ||a.length==0)
        {
            return a;
        }
        for(int i = 0;i<a.length-1;i++)
        {
            for(int j = i+1;j<a.length;j++)
            {
                if(a[i] > a[j])
                {
                    Util.Util.swap(a,i,j);
                }
            }
        }
        return a;
    }
//感觉不是很正宗呢 -.-
//两个循环+一个交换
```

选择排序

```
package sort;

public class Selection_sort {
    public int[] sort(int[] a){//不断的挑选出未排序的最小的数

        for (int i=0;i<a.length-1;i++)
        {
            int flag = a[i];
            int minindex = i;
            for (int j = i+1;j<a.length;j++)
            {
                if(a[j] < a[minindex])
                {
                    minindex = j;
                }
            }
            Util.Util.swap(a,i,minindex);
        }
        return a;
    }

}
```

快速排序

```
package sort;
```

```
import Util.Util;
//分割数组,一边小于pivot,一边大于pivot,然后进行递归
public class quickSort {
    //得调用递归,之前不用递归出不来...
    public int[] sort(int[] a,int low,int high){
        if(low<high)
        {
            int index = partition(a,low,high);
            sort(a,low,index-1);
            sort(a,index+1,high);
        }
        return a;

    }

    public int partition(int[] a,int low,int high)
    {
        int pivot = a[low];
        while(low<high)
        {
            while(low<high && pivot <= a[high])
            {
                high--;
            }
            a[low] = a[high];//由于有个副本pivot,所以直接覆盖了
            while(low<high && pivot >= a[low]){
                low++;
            }
            a[high] = a[low];
        }
        a[low] = pivot;
        return low;
    }


}
```

直接插入排序

```
package sort;

import Util.Util;

public class insertionSort {
    //插入排序也是一种思想很简单的排序方法，
    // 它通过比较当前元素和其之前已排好序的元素的大小，
    // 找到合适的位置插入，并把插入位置后的元素往后移动。
    public int[] sort(int[] a)
    {
        for (int i = 1;i<a.length;i++)
        {
            int now = a[i];
            int index = i;
```

```
            for(int j = i;j>=1;j--)
            {
                if(now < a[j-1])
                {
                    a[j] = a[j-1];
                    index = j-1;
                }
            }
            a[index] = now;
        }
        return a;
    }

}

//把向后移动的代码改成交换的代码也是可以实现排序的
```

希尔排序

```
package sort;

public class ShellSort {
    //和直接插入排序差不多,只是多了步长,并不断减少步长
    public int[] sort(int[] a)
    {
        int section = a.length/2;
        while(section>=1)
        {
            for(int i = section;i<a.length;i++)
            {
                int now = a[i];
                int index = i;
                for(int j = i-section;j>=0;)
                {
                    if(now < a[j])
                    {
                        a[j+section] = a[j];
                        index = j;
                    }
                    j = j-section;
                }
                a[index] = now;
            }
            section = section/2;
        }
        return a;
    }
}
```

归并排序

```
package sort;
```

```
//主要是合并的函数起到排序的作用,利用两个指针进行排序
public class MergeSort {
    public int[] sort(int[] a,int left,int right)
    {
        if(left < right)
        {
            int mid = (left+right)/2;
            sort(a,left,mid);
            sort(a,mid+1,right);
            merge(a,left,mid,right);
        }
        return a;
    }

    public void merge(int[] a, int left, int mid, int right)
    {
        int[] temp = new int[right-left+1];
        int i = left;
        int j = mid+1;
        int t = 0;//临时数组指针
        while(i<=mid&&j<=right)
        {
            if(a[i] < a[j])
            {
                temp[t++] = a[i++];
            }else{
                temp[t++] = a[j++];
            }
        }
        while(i<=mid)
        {
            temp[t++] = a[i++];
        }
        while(j<=right)
        {
            temp[t++] = a[j++];
        }
        t = 0;
        while(left<=right)
            a[left++] = temp[t++];
    }
}
```

堆排序

```
package sort;

import Util.Util;
//0开始的数组,子节点是2*下标+1,2*下标+2
//找到子节点中最大的数,然后和父节点比较大小并交换
public class HeapSort {
    public int[] sort(int[] a)
    {
        for(int i = a.length/2-1;i>=0;i--) //a.length/2-1是最后一个父节点的位置
```

```
        {
            HeapAdjust(a,i,a.length-1);
        }

        for(int i = a.length-1;i>=0;i--)
        {
            Util.swap(a,0,i);
            HeapAdjust(a,0,i-1);
        }

        return a;
    }

    public void HeapAdjust(int[] a,int l,int h)
    {
        int temp = a[l];
        for(int j = 2*l+1;j<=h;j=j*2+1)//得到子节点的最大值
        {
            if(j+1 <=h && a[j] < a[j+1])
            {
                j++;
            }
            if(temp < a[j])
            {
                a[l] = a[j];
                l = j;
            }
        }
        a[l] = temp;
    }

}
```

其他排序还有 桶排序,计数排序,基数排序(都是非比较排序),想偷懒就不写了
ongue