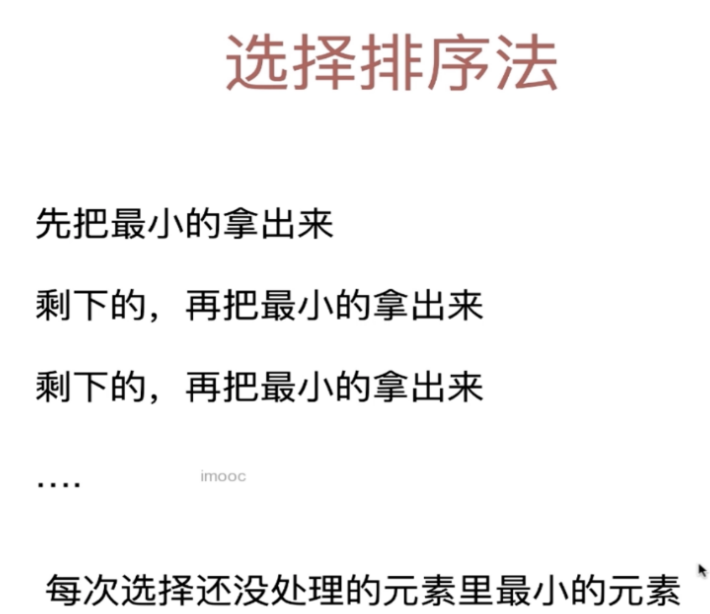
数据结构与算法2-选择排序

笔记本: 我的笔记
创建时间: 2020/9/18 21:45 **更新时间:** 2020/10/3 10:05
作者: liuhouer
标签: 算法

基础排序算法：选择排序



复杂度 O（n^2） 最多次数是【n个 * n个】

```
public class SelectionSort {


    private SelectionSort(){}


    public static void sort(int[] arr){


        // arr[0...i) 是有序的; arr[i...n) 是无序的
        for(int i = 0; i < arr.length; i ++){


            // 选择 arr[i...n) 中的最小值的索引
            int minIndex = i;
            for(int j = i; j < arr.length; j ++){
                if(arr[j] < arr[minIndex])
                    minIndex = j;
            }


            swap(arr, i, minIndex);
        }
    }


    private static void swap(int[] arr, int i, int j){
```

```
        int t = arr[i];
        arr[i] = arr[j];
        arr[j] = t;
    }


    public static void main(String[] args){


        int[] arr = {1, 4, 2, 3, 6, 5};
        SelectionSort.sort(arr);
        for(int e: arr)
            System.out.print(e + " ");
        System.out.println();
    }
}
```

使用泛型实现， 实现comparable接口后重写compareTo方法 即可灵活实现

```
public class SelectionSort {


    private SelectionSort(){}


    public static <E extends Comparable<E>> void sort(E[] arr){


        // arr[0...i) 是有序的; arr[i...n) 是无序的
        for(int i = 0; i < arr.length; i ++){


            // 选择 arr[i...n) 中的最小值
            int minIndex = i;
            for(int j = i; j < arr.length; j ++){
                if(arr[j].compareTo(arr[minIndex]) < 0)
                    minIndex = j;
            }


            swap(arr, i, minIndex);
        }
    }


    private static <E> void swap(E[] arr, int i, int j){


        E t = arr[i];
        arr[i] = arr[j];
        arr[j] = t;
    }


    public static void main(String[] args){


        Integer[] arr = {1, 4, 2, 3, 6, 5};
        SelectionSort.sort(arr);
        for(int e: arr)
            System.out.print(e + " ");
        System.out.println();
    }
}
```

性能测试帮助类

```
public class SortingHelper {


    private SortingHelper(){}


    public static <E extends Comparable<E>> boolean isSorted(E[] arr){


        for(int i = 1; i < arr.length; i ++)
            if(arr[i - 1].compareTo(arr[i]) > 0)
                return false;
        return true;
    }


    public static <E extends Comparable<E>> void sortTest(String sortname, E[]
arr){


        long startTime = System.nanoTime();
        if(sortname.equals("SelectionSort"))
            SelectionSort.sort(arr);
        long endTime = System.nanoTime();


        double time = (endTime - startTime) / 1000000000.0;


        if(!SortingHelper.isSorted(arr))
            throw new RuntimeException(sortname + " failed");
        System.out.println(String.format("%s , n = %d : %f s", sortname,
arr.length, time));
    }
}

-----------------------------------------------------------------------------------------

public static void main(String[] args){


        int[] dataSize = {10000, 100000};
        for(int n: dataSize){
            Integer[] arr = ArrayGenerator.generateRandomArray(n, n);
            SortingHelper.sortTest("SelectionSort", arr);
        }
    }
-------------------------------------------------------------------------------


import java.util.Random;


public class ArrayGenerator {


    private ArrayGenerator(){}


    public static Integer[] generateOrderedArray(int n){


        Integer[] arr = new Integer[n];
        for(int i = 0; i < n; i ++)
            arr[i] = i;
        return arr;
    }


    // 生成一个长度为 n 的随机数组，每个数字的范围是 [0, bound)
    public static Integer[] generateRandomArray(int n, int bound){


        Integer[] arr = new Integer[n];
```

```
        Random rnd = new Random();
        for(int i = 0; i < n; i ++)
            arr[i] = rnd.nextInt(bound);
        return arr;
    }
}
```