此文章已于 16:33:10 2017/12/19 重新发布到 HealerJean梦想博客

排序算法

类别 java面试 ; java

## 1、直接插入排序

### 1、解释：从前到后，依次插入进行排序



### 2、代码

|  |
| --- |
| /\*\*  \* 1、直接插入排序  \*/  @Test  **public** **void** insertionSort() {  **int**[] a = { 49, 38, 65, 97, 76, 13, 27, 50 };  System.*out*.println("----------插入排序开始：---------");  *print*(a);  **int** tmp;  **for** (**int** i = 1; i < a.length; i++) {  **for** (**int** j = i; j > 0; j--) {  **if** (a[j] < a[j - 1]) {  tmp = a[j - 1];  a[j - 1] = a[j];  a[j] = tmp;  }  }  System.*out*.printf("第"+i+"趟排序结果,");  *print*(a);  }    System.*out*.print("最终插入排序结果： ");  *print*(a);  System.*out*.println("--------------------");  } |

## 2、冒泡排序

### 1、代码

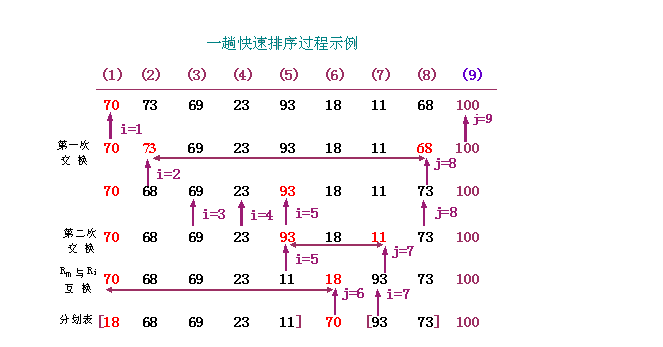
|  |
| --- |
| /\*\*  冒泡排序  \*/  @Test  **public** **void** maopao(){  **int** str[] = { 49, 38, 65, 97, 76, 13, 27, 50 };  System.*out*.println("----------冒泡排序 开始：---------");  **for** (**int** i = 0; i <str.length-1; i++){ //最多做n-1趟排序  **for**(**int** j = 0 ;j <str.length - i - 1; j++){ //对当前无序区间str[0......length-i-1]进行排序(j的范围很关键，这个范围是在逐步缩小的)  **if**(str[j] > str[j + 1]){ //把小的值放到前面  **int** temp = str[j];  str[j] = str[j + 1];  str[j + 1] = temp;  }  }  **int** n = i+1;  System.*out*.printf("第"+n+"趟排序结果,");  *print*(str);  }  System.*out*.print("最终排序结果：");  *print*(str);  } |

## 2、冒泡排序优化

|  |
| --- |
| /\*  \* 冒泡排序优化一  \* 设置一个标记来标志一趟比较是否发生交换  \* 如果没有发生交换，则数组已经有序  \* \*/  @Test  **public** **void** bubbleSort1() {  **int** arr[] = { 49, 38, 65, 97, 76, 13, 27, 50 };  **int** n = arr.length;  **int** i = 0;  **int** j = 0;  **int** tmp = 0;  **int** flag = 0;  **for** (i = 0; i < n; ++i) {  flag = 0;  **for** (j = 0; j < n - 1 - i; ++j) {  **if** (arr[j] > arr[j + 1]) {  flag = 1;  tmp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = tmp;  }  }  **if** (flag == 0) {  **break**;  }  }    *print*(arr);  }    /\*  \* 冒泡排序优化二  \* 用一个变量记录下最后一个发生交换的位置，后面没有发生交换的已经有序  \* 所以可以用这个值来作为下一次比较结束的位置  \* \*/  @Test  **public** **void** bubbleSort2() {  **int** arr[] = { 49, 38, 65, 97, 76, 13, 27, 50 };  **int** n = arr.length;  **int** i = 0;  **int** j = 0;  **int** k = 0;  **int** tmp = 0;  **int** flag = n;  **for** (i = 0; i < flag; ++i) {  k = flag;  flag = 0;  **for** (j = 0; j < k; ++j) {  **if** (arr[j] < arr[j + 1]) {  flag = j;  tmp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = tmp;  }  }  }  *print*(arr);  } |

## 3、快速排序

### 1、图解： 一般以首元素为枢纽，分别从两端向中间逼近，通过该枢纽与其他记录的比较和交换。



|  |
| --- |
| /\*  \* 4.快速排序  \*/  **public** **void** QuickSort(**int** []a, **int** low, **int** high){    **int** i=low,j=high;  **if**(i<j){  **int** po = a[low];  **while**(i<j){  **while**(i<j && po<a[j]){ //从后往前推 j++  j--;  }  **if**(i<j){  **int** temp = a[i];  a[i] = a[j];  a[j] = temp;  i++;  }      **while**(i<j && po>a[i]){ //从前往后推 i++  i++;  }  **if**(i<j){  **int** temp = a[i];  a[i] = a[j];  a[j] = temp;  j--;  }  }  QuickSort(a,low,j-1); //从小到J  QuickSort(a,j+1,high); //从J到到  }  }    @Test  **public** **void** kspx(){  **int** []a = { 49, 38, 65, 97, 76, 13, 27, 50 };  **int** low = 0;  **int** high = a.length-1;  QuickSort(a, 0, a.length-1);  *print*(a);  } |

## 4、希尔排序

解释：

|  |
| --- |
| /\*\*  2、希尔排序.最小缩量排序 比如 8/8 个数字 4 2 1  \*/  @Test  **public** **void** sort(){  **int**[] as = { 49, 38, 65, 97, 76, 13, 27, 50 };  System.*out*.println("----------希尔排序开始：---------");  *print*(as);  //增量  **int** incr = as.length/2;  **while**(incr >=1){  **for**(**int** i=0;i<as.length;i++){  //进行插入排序  **for**(**int** j=i;j<as.length-incr;j=j+incr){  **if**(as[j]>as[j+incr]){  **int** temple = as[j];  as[j] = as[j+incr];  as[j+incr] = temple;  }  }  }  //设置新的增量  incr = incr/2;  }  *print*(as);    } |

## 5、简单选择排序

### 解释：每次选择最小的放到最前面

|  |
| --- |
| /\*  \* 4.简单选择排序 每次选择最小的放到最前面  \*/  @Test  **public** **void** SelectSort()  {  **int** []a = { 49, 38, 65, 97, 76, 13, 27, 50 };  **int** aCount = a.length;  **int** min;  **for**(**int** i = 0; i < aCount - 1; ++i)  {  min = i;  **for**(**int** j = i + 1; j < aCount; ++j)  {  **if**(a[j] < a[min]) //使得min总是指向最小元素  min = j;  }  **if**(min != i) //即min有移动过  {  **int** temp = a[i];  a[i] = a[min];  a[min] = temp;  }  }  *print*(a);  } |

## 6、其他的排序

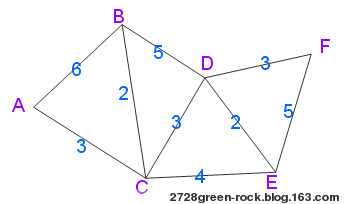
### 归并排序

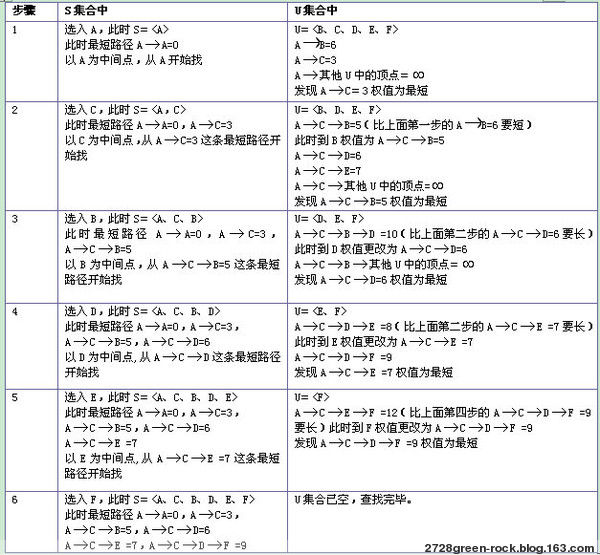
### 堆排序

# 最短路径（不止一个，sb了吧，哈哈）

## 1、最短路径寻找的方法

### 1、具体图例与算法执行步骤：（就从A开始，到各节点的最短路径)。



[](http://photo.blog.sina.com.cn/showpic.html#blogid=7f09a7160101th2g&url=http://album.sina.com.cn/pic/002keSDYgy6HQiBKrbP69)