尝试回顾以前学习的数据结构算法，常记录，常总结：

首先想到的是排序算法，第一个想到的是冒泡排序：

那么我们将回归一下冒泡排序算法：

它重复地走访过要排序的元素列，依次比较两个相邻的元素，如果他们的顺序错误就把他们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换。

这个算法的名字由来是因为越大的元素会经由交换慢慢“浮”到数列的顶端（升序或降序排列），就如同碳酸饮料中二氧化碳的气泡最终会上浮到顶端一样，故名“冒泡排序”。

知识点：

1.一对一对元素比较，把大的放在右边，小的放在左边。

2. 对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。

3. 针对所有的元素重复以上的步骤，除了最后一个。

4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

首先咱们调包一下：

import java.util.Arrays;

public class pubble\_learning3 {

public static void main(String[] args) {

long startTime = System.currentTimeMillis();

int[] arr = {54,26,93,17,77,31,44,55,20};

System.out.print("冒泡排序之前数组打印");

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+" ");

}

Arrays.sort(arr); //系统自带

System.out.print("\n");

System.out.print("冒泡排序之后数组打印");

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+" ");

}

long endTime = System.currentTimeMillis();

long usedTime = endTime-startTime;

System.out.println();

System.out.print("方法所用毫秒数");

System.out.print(usedTime);

}

}

另外，自己手撕一下代码

|  |  |
| --- | --- |
| Python | java |
| import timeit  li = [54,26,93,17,77,31,44,55,20]  print(li)  time\_start=timeit.default\_timer()  ###我们对算法进行分开解析  for j in range(len(li1)-1): ##外层循环  for i in range(len(li1)-1-j): ##内层循环遍历  if li1[i]>li1[i+1]: ##判断顺序  li1[i],li1[i+1]=li1[i+1],li1[i]  print(li1)  print(li1)  time\_end=timeit.default\_timer()  print('算法耗时',time\_end-time\_start,'s') | public class pubble\_learning{  public static void main(String[] args) {  long startTime = System.currentTimeMillis();  //输入一个数组  int arr[]={54,26,93,17,77,31,44,55,20};  //输出未排序前的数组  System.out.print("冒泡排序之前数组打印");  for(int i = 0;i<arr.length;i++){  System.out.print(arr[i]+" ");  }  //换行  System.out.println();  //进行冒泡排序  for(int i =0;i<arr.length-1;i++){  //第二层循环，最大的数排到数组底部  for(int j = 0;j<arr.length-i-1;j++){  if(arr[j]>arr[j+1]){  int temp;  temp = arr[j];  arr[j] = arr[j+1];  arr[j+1] = temp;  }  //这个部分是打印输出变化的部分  for(int k = 0;k<arr.length;k++){  System.out.print(arr[k]+" ");  }  System.out.print("\n");  }  }  //输出查看结果  System.out.print("\n");  System.out.print("冒泡排序之后数组打印");  for(int i = 0;i<arr.length;i++){  System.out.print(arr[i]+" ");  }  long endTime = System.currentTimeMillis();  long usedTime = endTime-startTime;  System.out.println();  System.out.print("方法所用毫秒数");  System.out.print(usedTime);  }  } |

如果给出的排序是正序的话，那么我们只需要遍历n-1次就好，所以时间复杂度为

如果给出的排序是逆序的话，那么我们需要遍历n-1次，每次需要n-i次比较，所以时间复杂度为

当然可以优化这个排序。

1. 假设我们现在排序ar[]={1,2,3,4,5,6,7,8,10,9}这组数据，按照上面的排序方式，第一趟排序后将10和9交换已经有序，接下来的8趟排序就是多余的，什么也没做。所以我们可以在交换的地方加一个标记，如果那一趟排序没有交换元素，说明这组数据已经有序，不用再继续下去。（加标记）

|  |  |
| --- | --- |
| Python | java |
| import timeit  li = [54,26,93,17,77,31,44,55,20]  li1=li  print(li)  time\_start=timeit.default\_timer()  ###我们对算法进行分开解析  for j in range(len(li1)-1): ###j 是每词遍历的次数  flag = False  for i in range(len(li1) - 1 - j):  if li1[i]>li1[i+1]:  li1[i],li1[i+1]=li1[i+1],li1[i]  print(li1)  flag = True  if not flag:  break  print(li1)  time\_end=timeit.default\_timer()  print('算法耗时',time\_end-time\_start,'s') | public class pubble\_learning1 {  public static void main(String[] args) {  long startTime = System.currentTimeMillis();  //输入一个数组  int arr[]={54,26,93,17,77,31,44,55,20};  //输出未排序前的数组  System.out.print("冒泡排序之前数组打印");  for(int i = 0;i<arr.length;i++){  System.out.print(arr[i]+" ");  }  //换行  System.out.println();  //进行冒泡排序  for(int i =0;i<arr.length-1;i++){  //第二层循环，最大的数排到数组底部  boolean flag=false;  for(int j = 0;j<arr.length-i-1;j++){  if(arr[j]>arr[j+1]){  int temp;  temp = arr[j];  arr[j] = arr[j+1];  arr[j+1] = temp;  }    for(int k = 0;k<arr.length;k++){  System.out.print(arr[k]+" ");  }  System.out.print("\n");  }  if(!flag) { //没有交换，输出结果  break;  }  }  //输出查看结果  System.out.print("\n");  System.out.print("冒泡排序之后数组打印");  for(int i = 0;i<arr.length;i++){  System.out.print(arr[i]+" ");  }  long endTime = System.currentTimeMillis();  long usedTime = endTime-startTime;  System.out.println();  System.out.print("方法所用毫秒数");  System.out.print(usedTime);  }  } |

2.双向排序：从左到右，最大放置右边，在从右到左，最小放在左边。

|  |  |
| --- | --- |
| Python | java |
| import timeit  li = [54,26,93,17,77,31,44,55,20]  li1=li  print(li)  time\_start=timeit.default\_timer()  ###我们对算法进行分开解析  for j in range(len(li1)-1): ###j 是每词遍历的次数  flag = True  if flag:  flag = False  for i in range(len(li1)-1-j):  if li1[i]>li1[i+1]:  li1[i],li1[i+1]=li1[i+1],li1[i]  flag = True  for j in range(len(li1) - 2 - i, 0, -1):  if li1[j - 1]>li1[j]:  li1[j], li1[j - 1] = li1[j - 1], li1[j]  flag = True  else:  break  print(li1)  print(li1)  time\_end=timeit.default\_timer()  print('算法耗时',time\_end-time\_start,'s') | public class pubble\_learning2 {  public static void main(String[] args) {  long startTime = System.currentTimeMillis();  int []array= {54,26,93,17,77,31,44,55,20};  //输出未排序前的数组  System.out.print("冒泡排序之前数组打印");  for(int i = 0;i<array.length;i++){  System.out.print(array[i]+" ");  }  //换行  System.out.println();  for(int i=0;i<array.length;i++)  {  for(int j=0;j<array.length-i-1;j++) {  if(array[j]>array[j+1])  {  int temp=array[j];  array[j]=array[j+1];  array[j+1]=temp;  }  }  for(int j=array.length-i-1;j>i;j--) {  if(array[j]<array[j-1])  {  int temp=array[j];  array[j]=array[j-1];  array[j-1]=temp;  }  }  for(int k = 0;k<array.length;k++){  System.out.print(array[k]+" ");  }  System.out.print("\n");  }  //输出查看结果  System.out.print("\n");  System.out.print("冒泡排序之后数组打印");  for(int i = 0;i<array.length;i++){  System.out.print(array[i]+" ");  }  long endTime = System.currentTimeMillis();  long usedTime = endTime-startTime;  System.out.println();  System.out.print("方法所用毫秒数");  System.out.print(usedTime);  }  } |