## **简介**

在编程中，算法占据及其重要的位置，好的算法可以极高的提高程序运行的效率，追求一个优秀的算法也是我们应具有的品质，这样自己编写的代码才会更加简练、更加高效。  
排序算法是很多算法的基础，也是我们在编写程序时最易遇到的，这次来聊聊多种排序算法，理解它们的优劣，并将它们运用起来

## **桶排序**

第一次见到这个排序算法时在《啊哈，算法》一书中看见的，然后自己通过Google深入了解了一番，它是一种非基于比较的排序方法，如我们常见的冒泡排序、归并排序或快速排序都是基于比较的排序方法，也就是元素与元素之间要进行比较，然后根据比较的结果来排序，这种排序方法其时间复杂度不可能小于O(NlogN)，这个结论可以通过斯特林公式来证明，证明过程如下

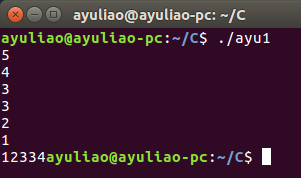
如果有N个数需要进行排列，那么就会有N!种排列情况，也就是基于比较的排序算法会有N!种情况，最少要比较log(N!)次，log(N!)=O(NlogN)(斯特林公式)

这里先讲一个比较简单的桶排，通过一个数组就可以实现

通过桶排为：4,3,3,2,1这5个数进行从小到大的排序

使用桶排序的C语言代码如下

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 | #include<stdio.h> int main(){  int book[20],i,j,t,n;  for(i=0;i<20;i++){  book[i]=0;  }   scanf("%d",&n);//有多少个数   for(i=1;i<=n;i++){  scanf("%d",&t);  book[t]++;   }   for(i=1;i<=20;i++){  for(j=1;j<=book[i];j++){  printf("%d",i);  }  }  return 0; } |

你会发现桶排序的逻辑非常简单，就是将数组中的每个元素都当成一个桶，往里面放相应的数值，因为数组本来就是有顺序的，所以直接显示就好，重复的数字重复显示  
[](http://obfs4iize.bkt.clouddn.com/Selection_011.png)

可以看出它的时间复杂度为O(n)(输出语句没考虑，单纯考虑桶排序的代码)

这种算法虽然简单，但是使用的情况比较受限，如果要排序的数直接间隔非常大，如对1,800,10000,50,9100这五个数进行排序，使用简单的桶排序就要非常大的空间，这样就会浪费比较多的内存，这里可以对要排序的数值离散化一下，如10000对应10,9100对应9之类的

还有桶排序无法排序比较复杂的结构，如：aa 177cm,bb 166cm,dd 144cm，要通过不同是升高来排序，则简单桶排序就无法实现需求了

可以看[三种线性排序算法 计数排序、桶排序与基数排序](https://www.byvoid.com/zhs/blog/sort-radix" \t "http://lmwen.top/2017/03/20/%E5%A6%99%E7%94%A8%E5%A4%9A%E7%A7%8D%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/_blank)这篇博客来进一步了解

## **冒泡排序**

冒泡排序应该是我们最常见的一种排序方式，实话说它比较低效，冒泡的核心思想是对比相邻的两个元素，然后通过对比结果来判断是否需要交换两者的位置，冒泡对这种算法的描述非常形象，数值间的交换就像是一个气泡一直往上“冒”，直到相应的位置，这种排序方法的时间复杂度比较高，为O(n^2)，简单的实例如下

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 | #include<stdio.h> int main(){  int a[100],i,j,t,n;  scanf("%d",&n);  for(i=1;i<=n;i++){  scanf("%d",&a[i]);//输入  }    //冒泡的核心，相邻的元素进行比较，然后判断是否要交换顺序  for(i=1;i<=n-1;i++){  for(j=1;j<=n-i;j++){  if(a[j]<a[j+1]){  t=a[j];  a[j]=a[j+1];  a[j+1]=t;  }  }  }  for(i=1;i<=n;i++){  printf("%d",a[i]);  }  return 0; } |

前面已经提过了，冒泡算法的时间复杂度比较高，可以通过下面两种方式来优化一下冒泡排序  
方法一：设置一个标志位，每次交换一次都重置一下这个标志位，当标志位没有被重置，就说明已经排序完成了，不必再比下去了，通过标志位改进后，只有在最坏的情况下（要排成正序时，输入的内容是倒序的）才会跟初始的冒泡方法有同样的时间复杂度

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 | #include<stdio.h> int main(){  int a[100],i,j,t,n;  scanf("%d",&n);  for(i=1;i<=n;i++){  scanf("%d",&a[i]);//输入  }  //方法一核心  bool flag = true;  int k = n-1;  while(flag){  flag = false;  for(j=1;j<=k;j++){  if(a[j]<a[j+1]){  t=a[j];  a[j]=a[j+1];  a[j+1]=t;  flag = true;  }  }  k--;  }   for(i=1;i<=n;i++){  printf("%d",a[i]);  }  return 0; } |

方法二：记录一轮交换下来标记的最后的位置，下次在遍历时，只用遍历到该位置就ok了

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 | #include<stdio.h> int main(){  int a[100],i,j,t,n;  scanf("%d",&n);  for(i=1;i<=n;i++){  scanf("%d",&a[i]);//输入  }  //方法二核心  int k,flag =n;  while(flag>0){  k=flag;  flag = 0;//如果一轮下来，没有交换元素，说明数据以及按相应的顺序排列好了，就不必进行下一轮了，所以flag为0  for(j=1;j<k;j++){  if(a[j]<a[j+1]){  t=a[j];  a[j]=a[j+1];  a[j+1]=t;  flag = j;  }  }  }   for(i=1;i<=n;i++){  printf("%d",a[i]);  }  return 0; } |

其实插入排序、简单选择排序的时间序列也为O(n^2)

时间复杂度为O(n^2)的算法都不是非常好的选择，下一章讲讲快速排序、堆排序和归并排序

可以看下面这篇博文先了解一下  
[十种排序算法总结（冒泡、插入、选择、希尔、归并、堆、快速，计数，桶，基数）](http://blog.csdn.net/jnu_simba/article/details/9705111" \t "http://lmwen.top/2017/03/20/%E5%A6%99%E7%94%A8%E5%A4%9A%E7%A7%8D%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/_blank)