算法讲解 | 多种排序算法，如何理解和选用

周行算协 周行算协 2020-11-08 12:42

排序算法

所谓排序算法，
即通过特定的算法因式
将一组或多组数据按照既定模式进行重新排序。
这种新序列遵循着一定的规则，
体现出一定的规律，
因此，经处理后的数据便于筛选和计算，
大大提高了计算效率。

稳定性

对于排序，我们首先要求其具有一定的稳定性，
即当两个相同的元素同时出现于某个序列之中，
则经过一定的排序算法之后，两者在排序前后的相对位置不发生变化。

换言之，即便是两个完全相同的元素，它们在排序过程中也是各有区别的，不允许混淆不清。

一些常见排序算法的复杂度和稳定性，本期推送对其中的几种进行梳理讲解

排序算法	平均时间复杂度	最坏时间复杂度	空间复杂度	是否稳定
冒泡排序	$O(n^2)$	$O(n^2)$	$O(1)$	是
选择排序	$O(n^2)$	$O(n^2)$	$O(1)$	不是
直接插入排序	$O(n^2)$	$O(n^2)$	$O(1)$	是
归并排序	$O(nlogn)$	$O(nlogn)$	$O(n)$	是
快速排序	$O(nlogn)$	$O(n^2)$	$O(logn)$	不是
堆排序	$O(nlogn)$	$O(nlogn)$	$O(1)$	不是
希尔排序	$O(nlogn)$	$O(n^s)$	$O(1)$	不是
计数排序	$O(n+k)$	$O(n+k)$	$O(n+k)$	是
基数排序	$O(N*M)$	$O(N*M)$	$O(M)$	是

冒泡排序

冒泡排序是一种交换排序，它的基本思想是，比较相邻两个记录的关键字，如果反序则交换，直到没有反序为止。
假设我们要将一个长度为n的数列从小到大排序：
先从前两位开始比较相邻的两个数，将大的数换到小的数的后面，之后依次进行同样的操作，直至最大的数被移到最后。
重复该操作，每次到第n-i位即可停止比较交换，直至第i大的数被移到倒数第i位。

代码如下：

```
void BuddleSort(int a*,int n)
{
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n-i;j++)
            if(a[j]>a[j+1])swap(a[j],a[j+1]);
}
```

插入排序

插入排序，一般也被称为直接插入排序。
插入排序是一种最简单的排序方法，它的基本思想是将一个数插入到已经排好序的有序表中，从而得到一个新的、长度增加1的有序表。
设前j-1位已经排好序，我们接下来要将第j位插入到这个有序表中，使之仍为有序的。因此设置一个指针，从j-1位到第1位依次与第j位比较，直到找到合适的位置。

伪代码如下：

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

归并排序

归并排序是建立在归并操作上的一种有效，稳定的排序算法。
该算法是采用分治法的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。

伪代码如下：

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

递归排序

```
MERGE(A, p, q, r)
 1  n1 = q − p + 1
 2  n2 = r − q
 3  let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n2
 7      R[j] = A[q + j]
 8  L[n1 + 1] = ∞
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

合并左右子序列

桶排序

首先讲讲计数排序：
计数排序不是基于比较的排序算法，其核心在于将输入的数据值转化为键存储在额外开辟的数组空间中。作为一种线性时间复杂度的排序，计数排序要求输入的数据必须是有确定范围的整数。

实现步骤如下：
1.统计数组中每个值为i的元素出现的次数，存入数组C的第i项；
2.对所有的计数累加（从C中的第一个元素开始，每项和前一项相加）；
3.反向填充目标数组：将每个元素i放在新数组的第C（i）项，每放一个元素就将C（i）减去1。

而桶排序是计数排序的升级版。它将数据分到有限数量的桶里，每个桶再分别排序

算法步骤如下：
1.设置一个定量的数组当作空桶；
2.输入数据，并且把数据一个一个放到对应的桶里去；
3.对每个不是空的桶进行排序
4.从不是空的桶里把排好序的数据拼接起来。

```
BUCKET-SORT(A)
1  let B[0..n − 1] be a new array
2  n = A.length
3  for i = 0 to n − 1
4      make B[i] an empty list
5  for i = 1 to n
6      insert A[i] into list B[⌊nA[i]⌋]
7  for i = 0 to n − 1
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

点击“阅读原文”，了解如何分析排序算法的复杂度吧！

阅读原文 阅读 111