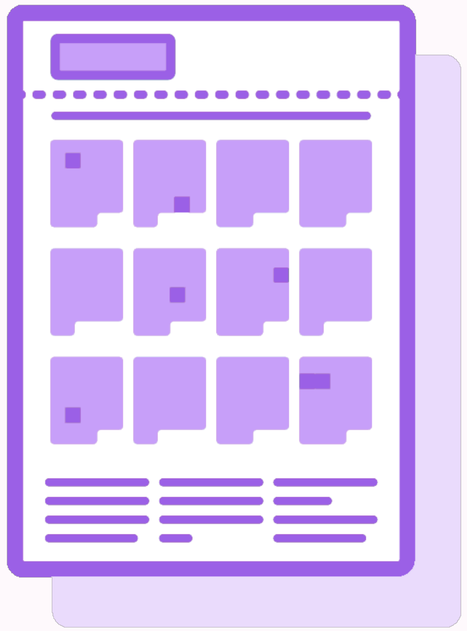
排序

斐多课堂 Phaedo Classes 数据结构 不挂科 第八讲



3大模块

7道题目

排序

模块1／ 排序的基本概念

模块2／ 内部排序

模块3／外部排序

排序的基本概念

小节1 / 排序的定义

小节2 / 稳定性

排序的基本概念

小节1 / 排序的定义

小节2 / 稳定性

排序的定义

将原本无序的序列重新排列成有序序列的过程

内部排序

在排序期间元素全部存放在内存中的排序

外部排序

在排序期间元素无法全部同时存放在内存中，必须在排序的过程中根据要求不断地在内、外存之间移动的排序

排序的基本概念

小节1 / 排序的定义

小节2 / 稳定性

稳定性

当待排序列中有两个或两个以上相同的关键字时，若排序前和排序后这些关键字的相对位置没有发生变化，则稳定

对于不稳定的排序算法，只需举出一组关键字的实例，说明它的不稳定性即可

内部排序

小节1 / 插入排序

小节2 / 交换排序

小节3 / 选择、归并、基数排序

内部排序

小节1 / 插入排序

小节2 / 交换排序

小节3 / 选择、归并、基数排序

直接插入排序

算法思想

每趟将一个待排序的关键字按照其值的大小插入到已经排好的部分有序序列的适当位置上，直到所有待排关键字都被插入到有序序列中为止

稳定，适用于顺序存储和链式存储

直接插入排序

时间复杂度

R[j+1]=R[j]

最坏的情况：$O(n^2)$

最好的情况：$O(n)$

平均时间复杂度：$O(n^2)$

直接插入排序

空间复杂度

所需辅助存储空间不随待排序列规模的变化而变化，是个常量

空间复杂度：O(1)

折半插入排序

算法思想

每趟将一个待排序的关键字按照其值的大小插入到已经排好的部分有序序列的适当位置上，直到所有待排关键字都被插入到有序序列中为止。

采用折半查找法查找插入位置

稳定

折半插入排序

时间复杂度

最坏的情况：$O(n^2)$

最好的情况：$O(n\log_2 n)$

平均时间复杂度：$O(n^2)$

折半插入排序

空间复杂度

所需辅助存储空间不随待排序列规模的变化而变化，是个常量

空间复杂度：O(1)

例题8-1 对五个不同的数据元素进行直接插入排序，最多需要进行的比较次数是（ ）。

A. 8　　B. 10

C. 15　　D. 25

解析8-1 **B**

直接插入排序在最坏的情况下要做n（n-1）/2次关键字的比较，当n=5时，关键字的比较次数为10（未考虑与哨兵的比较）。

希尔排序

算法思想

将待排序列按某种规则分成几个子序列，分别对这几个子序列进行直接插入排序

又称作缩小增量排序

如果增量为1，就是直接插入排序

不稳定，适用于线性表为顺序存储的情况

希尔排序

时间复杂度

最坏的情况：$O(n^2)$

最好的情况：约为$O(n^{1.3})$

希尔排序

空间复杂度

所需辅助存储空间不随待排序列规模的变化而变化，是个常量

空间复杂度：O(1)

例题8-2 希尔排序属于（ ）。

A. 插入排序　　B. 交换排序

C. 选择排序　　D. 归并排序

解析8-2 **A**

希尔排序是对直接插入排序算法改进后提出来的，本质上仍是鱼插入排序的范畴。

例题8-3 / 对序列{15，9，7，8，20，-1，4}用希尔排序方法排序，经一趟后序列变为{15，-1，4，8，20，9，7}，则该次采用的赠量是（ ）。

A．1　　B．4

C．3　　D．2

解析8-3 / **B**

希尔排序排序将序列分为若干组，记录只在组内进行交换。由观察可知，经过一趟后9和-1交换，7和4交换，可知增量为4。

内部排序

小节1 / 插入排序

小节2 / 交换排序

小节3 / 选择、归并、基数排序

冒泡排序

算法思想

首先第一个关键字和第二个关键字比较，如果第一个大，则二者交换，否则不交换；
然后第二个关键字和第三个关键字比较，如果第二个大，则二者交换，否则不交换...
一直按这种方式进行下去，最终最大的那个关键字被交换到了最后。

稳定

冒泡排序

时间复杂度

最坏的情况：$O(n^2)$

最好的情况：$O(n)$

平均时间复杂度：$O(n^2)$

例题8-4 对n个不同的元素利用冒泡法从小到大排序，在（ ）情况下元素交换的次数最多。

A. 从大到小排列好的　　B. 从小到大排列好的

C. 元素无序　　D. 元素基本有序

解析8-4 **A**

通常情况下，冒泡排序最少进行一次冒泡，最多进行n-1次冒泡。初始序列为逆序时，需进行n-1次冒泡，并且需要交换的次数最多。初始序列为正序时，进行一次冒泡（无交换）就可以终止算法。

快速排序

算法思想

通过多次划分操作实现排序。以升序为例：

每一趟选择当前所有子序列中的一个关键字作为枢轴，将子序列中比枢轴小的移到枢轴前边，比枢轴大的移到枢轴后边；
当本趟所有子序列都被枢轴以上述规则划分完毕后会得到新的一组更短的子序列，它们称为下一趟划分的初始序列集。

不稳定

快速排序

时间复杂度

最坏的情况：$O(n^2)$

最好的情况：$O(n\log_2 n)$

平均时间复杂度：$O(n\log_2 n)$

快速排序

空间复杂度

快速排序是递归进行的，递归需要栈的辅助，需要的空间比前面几类排序算法大

空间复杂度：$O(\log_2 n)$

内部排序

小节1 / 插入排序

小节2 / 交换排序

小节3 / 选择、归并、基数排序

简单选择排序

算法思想

从头至尾顺序扫描序列，找出最小的一个关键字，和第一个关键字交换，接着从剩下的关键字中继续这种选择和交换，最终使序列有序。

不稳定

简单选择排序

时间复杂度

平均时间复杂度：$O(n^2)$

简单选择排序

空间复杂度

所需辅助存储空间不随待排序列规模的变化而变化，是个常量

空间复杂度：O(1)

堆排序

算法思想

将一个无序序列调整为一个堆，就可以找出这个序列的最大（或最小）值，然后将找出的这个值交换到序列的最后（或最前），这样有序序列关键字增加一个，无序序列关键字减少一个，对新的无序序列重复这样的操作，实现排序。

不稳定

堆排序

时间复杂度

平均时间复杂度：$O(n\log_2 n)$

堆排序

空间复杂度

所需辅助存储空间不随待排序列规模的变化而变化，是个常量

空间复杂度：O(1)

归并排序

算法思想

假定待排序表含有n个记录，则可以看成是n个有序的子表，每个子表长度为1，然后两两归并，得到n/2（上取整）个长度为2或1的有序表；再两两归并，...如此重复，直到合并成一个长度为n的有序表为止。

稳定

归并排序

时间复杂度

平均时间复杂度：$O(n\log_2 n)$

归并排序

空间复杂度

需要转存整个待排序列

空间复杂度：O(n)

基数排序

算法思想

最高位优先：先按最高位排成若干子序列，再对每个子序列按次高位排序。

最低位优先：不必分成子序列，每次排序全体关键字都参与。不通过比较，而是通过“分配”和“收集”。

稳定

基数排序

时间复杂度

平均时间复杂度：O(d(n+r))

基数排序

空间复杂度

空间复杂度：$O(r_d)$

例题8-5 /

下列排序方法中，不属于内部排序方法的是（ ）。

A. 插入排序　　B. 选择排序

C. 拓扑排序　　D. 冒泡排序

解析8-5 /

C

拓扑排序是将有向图中所有结点排成一个线性序列，虽然也是在内存中进行的，但它不属于我们这里提到的内部排序范畴，也不满足前面排序的定义。

例题8-6

以下排序方法中，（）在一趟结束后不一定能选出一个元素放在其最终位置上。

A. 简单选择排序　　B. 冒泡排序

C. 归并排序　　D. 堆排序

解析8-6

C

前面我们知道插入排序不能保证在一趟结束后一定有元素放在最终位置上。事实上，归并排序叶不能保证。例如，序列{6，5，7，8，2，1，4，3}进行一趟2路归并排序（从小到大）后为{6,5,7,8,1,2,3,4},显然它们都未被放在最终位置上。

例题8-7 以下排序方法中，排序过程中比较次数的数量级与序列初始状态无关的是（ ）。

A. 归并排序　　B. 插入排序

C. 快速排序　　D. 冒泡排序

解析8-7 **A**

前面我们知道选择排序的比较次数与序列初始状态无关，归并排序也与序列的初始状态无关，读者还应能从算法的原理方面来考虑为什么和初始状态无关。

外部排序

小节1 / 外部排序的方法

小节2 / 最佳归并树

小节3 / 败者树

外部排序

外部排序的方法

由于外存设备的不同，可分为磁盘文件排序和磁带文件排序两大类

两类排序的基本步骤相类似，不同之处在于初始归并段在外存介质中的分布方式，磁盘是直接存取设备，磁带是顺序存取设备。

外部排序通常采用归并排序算法

外部排序

最佳归并树

归并过程可以用一棵树来形象地描述，这棵树称为归并树

树中结点代表当前归并长度。初始记录经过置换-选择排序之后，得到的是长度不等的初始归并段，归并策略不同，所得的归并树也不同树的带权路径长度也不同。

最佳归并树的优化

为了优化归并树的带权路径长度，可将之前讲过的赫夫曼树运用到这里

对于k路归并算法，可以用构造k叉赫夫曼树的方法来构造最佳归并树

外部排序

小节1 / 外部排序的方法

小节2 / 最佳归并树

小节3 / 败者树

败者树

引入败者树，则每次不需要k-1次比较，只需要约$\log_2 k$次即可

败者树的建立

对当前读入的k个记录，构造k个叶子结点，任意两个结点为一组，建二叉树，如果结点树不是偶数，则多余的那个结点放在下一趟处理。

如果当前参与建树的是两个叶子结点，则以败者（记录值较大者）所在归并段的序号构造新结点作为其父结点（T），胜者（记录值较小者）所在归并段的序号构造新结点作为T的父结点建一棵二叉树。

如果当前参与建树的是两个结点是非叶根结点（必为单分支结点），则以败者为这两个根结点子树的心根结点（T），胜者为T的父结点，建立一棵二叉树。

败者树的建立

如果当前参与建树的两个结点一个是叶子结点，一个是非叶根结点，则有两种情况：

如果叶子结点是胜者，则叶子结点挂在非叶根结点的空分支上，并以叶子结点记录值所在归并段序号构造新结点作为非叶根结点的父接待你建一棵二叉树。

如果叶子结点败者，则以叶子结点记录值所在归并段序号构造新结点作为叶子结点和非叶根结点子树的新根结点（T），原非叶根结点作为T的父结点建一棵二叉树。

排序

斐多课堂 Phaedo Classes 数据结构 不挂科 第八讲