**数据结构和算法**

* **介绍**

什么是数据结构？

数据结构就是数据在内存上的组织形式。

什么是算法？

算法就是对数据进行处理的过程。

数据结构和算法能起到什么作用？

现实世界的数据的存储、程序员的工具、现实世界的建模。

**数据结构概述**：

|  |  |  |
| --- | --- | --- |
| **数据结构** | **优点** | **缺点** |
| 数组 | 插入块，如果知道下标，如果知道下标可以非常快的存取 | 查找慢，删除慢，固定大小 |
| 有序数组 | 比无序数组查找快 | 删除和插入慢，大小固定 |
| 栈 | 提供先进先出的存取 | 存取其它项很慢 |
| 队列 | 提供先进先出的存取 | 存取其它项很慢 |
| 链表 | 插入块，删除快 | 查找慢 |
| 二叉树 | 查找，插入，删除都很快（如果树保持平衡） | 删除算法复杂 |
| 红黑树 | 查找，插入，删除都很快，树总是平衡的 | 算法复杂 |
| 2-3-4树 | 查找，插入，删除都很快。类似的树对磁盘存储有用 | 算法复杂 |
| 哈希表 | 如果关键字已知存取极快，插入块 | 删除慢，如果不知道关键字存取很慢，对存储空间使用不充分 |
| 堆 | 插入、删除都快，对最大数据项的存取很快 | 对其他数据项存取慢 |
| 图 | 对现实世界建模 | 有些算法慢且复杂 |

**算法的概述：**

算法是对数据的处理过程。对于大多数数据结构来说都需要知道：

* 插入一条新数据项。
* 寻找某一特定的数据项。
* 删除某一特定的数据项。

另外，递归与排序也属于算法的范畴。

* **数组**

数组是应用最广泛的数据存储结构，而插入、查找、删除是大多说数据结构最基本的操作。

如下数组，它是一个长度为7的数组，它里面存储了4个数据：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d |  |  |  |

0 1 2 3 4 5 6

* **插入**

插入一条数据的操作只需要一步就能完成，因为我们已知数组的大小和数组中存储数据的个数，所以我们可以直接的知道数组中下一个空位的位置，进而直接将数据插入到数组的下一个空位上即可。（个人认为这里缺少最重要一步就是判断数组大小是否够用，如果不够用还需要扩展数组等复杂的操作）。

* **查找**

查找一条特定的数据项，就是从数组的第一个位置开始往后将数组的每一个元素依次的拿出来与要查找的特定的数据进行比较，当查到想要的数据后将数据返回。

我们忽略比较的步骤，假如数组的长度为N，查询数组第一个数据项最快可以忽略不计，查找数组最后一个数据项需要N步最慢。所以我们在一个数组中查询一条数据使用的**平均长度**为N/2步。所以我们必须平均查找一半的数据项来查询特定的数据。

* **删除**

删除一个数据项，首先需要遍历数组查找到这个数据项，总共需要N/2步，查找到后将该位置置空。

删除算法中暗含着一种假设，就是假设数组中不能有洞（即数组的数据必须是连续排列的，连续的数据项中间不能有空的单元格），如果数组中有洞，那么其他的算法也会变得十分复杂并且效率很低，因为需要判断每个数据项是否为空。

所以删除算法中为了不出现洞，当删除掉一个非尾部的数据项之后，还需要将这个被删除的数据项后面的数据依次向前移动一位来填补这个洞。所以，**删除需要平均查找N/2个数据项并且需要平均移动剩下的N/2个数据项填充出现的洞，总共需要N步。**

**重复值为题**

以上讲的三种算法除了插入都是在没有重复值的前提下，在实际使用中需要考虑数据项是否可以重复的问题。那么下面讲解数组中考虑重复值的情况下的算法：

* **插入**

不管允不允许有重复值（同时也跟数组中有没有洞没关系），数组中插入一条数据只需要一步（插入一条数据项算为一次移动），就是将新数据项直接添加到数组的第一个空位上。注意这里将要插入的数据项是指绝对不会与数组中的数据项有重复，这就需要在插入之前遍历整个数组然后与将要插入的数据项进行对比（而检测这一步是没有算到插入的步骤的）。

* **查找**

允许有重复值的情况下，需要查找整个数组，即需要N步才能查完所查询的数据项（如果只查找第一个出现的位置，那么需要N/2步）。

* **查找**

在有重复值的情况下，删除数组中全部的特定数据项步骤比较复杂，首先需要N步的查找，然后（可能）需要移动多于N/2个数据项来补洞。有重复值的情况下删除操作的平均时间依赖于重复值在整个数组中分布的情况。

说明：因为在有重复值的情况下，删除数组的特定数据，只需要遍历一边数组，也就是检查N步，每检查到一个特定的数据项就将其删除，最后在将数组中的洞通过移动数据项来填补，所以可能需要移动多于N/2个数据项。当然将数组中的全部特定项删除的过程当中，每删除一个数据项，就移动后面的数据项补洞和全部删除后在补洞这两种方式是看自己如何选择的，它们总步数是一样的。

* **总结**

移动：指数据项由一个单元格移到另一个单元格。

比较：包括指针移动到下一个位置然后将数据项取出来与特定数据项进行比较。

|  |  |  |
| --- | --- | --- |
|  | 不允许重复 | 允许重复 |
| 查找 | N/2次比较 | N次比较 |
| 插入 | 一次移动（插入前检测数组，需要N次比较） | 一次移动 |
| 删除 | N/2次比较，N/2次移动 | N次比较，可能多于N/2次移动 |

通常认为N和N/2之间的差异不是很重要，除非正在对程序进行微调以达到最优。更重要的是操作是否需要一步、N步、log(N)步或者N的2次方步。

* **有序数组**

数组中的数据项按关键字升序或者降序排列，这样的数组被称为有序数组。假设有序数组中不允许出现重复值，那么有序数组极大地提高了查找的效率，同时也提高了删除的效率，但是降低了插入的效率，这些事有序数组的特性。

* **查找**

**线性查找：**无序数组是通过线性查找来检索数据的，线性查找是指通过遍历数组中的数据项来检索特定的数据项。

**二分查找：**有序数组的查找使用的是二分查找，假设数组升序排序，二分查找首先获取数组中间位置的数据项与目标项比较，如果与目标项匹配则查找成功，如果中间项比目标项大，则继续从中间项下一个数据项开始到数组最后一项这一半区域的数据项以相同的方式查找。每查找一次查找范围就会缩小一半，并且选择这个范围的中间点作为对比值。直到这个范围无法在分割，还没有找到数据项则停止查找。

* **插入**

有序数组的插入比较麻烦，假设数据项按升序排列，首先要找到合适的位置：刚好在稍小值的后面，稍大值的前面。然后将所有比待插数据项大的值向后移以便腾出位置。

* **删除**

有序数组的删除，可以通过二分查找找到要删除的数据项，然后将这个数据项后面的数据项向前移一位来补充删除数据项的洞。

通过二分查找可以提高数组的查找效率，但是由于插入时需要找到合适的位置，所以插入新的数据项不但要执行查找的步骤，还要将需要插入的位置它后面的数据项向后移动而腾出空间。这样插入操作的效率的就变得很低。删除操作通过使用二分查找来删除数据项相对于无序数据也提高了效率，但是它依旧需要通过移动删除项后面的数据项来补洞而效率低下。