**数据结构与算法**

**思考题**

|  |  |  |
| --- | --- | --- |
| **课程名称**：数据结构与算法实验 | **年级**：2015级 | **成绩**： |
| **指导教师**：陆正福 | **姓名**：刘鹏 |  |
| **上机实践名称**：线性存储结构 | **学号**：20151910042 | **日期**：2017-05-14 |
| **思考题编号**：No.04 | **组号**：01-01 | **时间**：上午3、4节 |

**查阅资料，思考并回答下述问题：**

1. **基本存储结构有哪两类？（要点提示：顺序结构，链式结构）**

答：顺序结构（数组）、链式结构。

1. **常用的基本操作有哪些？  
     
   要点提示:  
   插（insert/add a node）  
   删（remove/delete a node）  
   改（modify an element）  
   取（get an element）**

答：插删改取。

1. **作为顺序存储结构的数组适用于哪些高频度操作，不适用于哪些高频度操作？（要点提示：分别为改与取，插与删）试举例给出其应用场景。**

答：当经常改、取的时候，可以用线性数组，但是插入与删除，尤其是在中间插入与删除，数组为了保持下标寻址，会导致复杂度很高（线性）。

1. **作为链式存储结构的单链表适用于哪些高频度操作，不适用于哪些高频度操作？试举例给出其应用场景。**

答：单链表适合经常插入与删除，但是不适合删除尾部元素，因为这需要遍历单链表。而改、取也不合适，因为单链表不好定位，要找到某一个元素，必须顺着链条找过去，不能下标索引。

1. **作为链式存储结构的双链表适用于哪些高频度操作，不适用于哪些高频度操作？试举例给出其应用场景。**

答：插入删除很适合，但是改、取不太适合，仍然不能很好定位。不过双链表解决了删除尾部元素的问题，将之做到了常量操作。只要定义head与tail就可以了。tail的指向就是尾部元素，而head与tail仅仅做标记，不存储实际应用的元素。

1. **单链表上如果需要做尾部删除，应该如何设计？付出的处理代价是什么？**

答：单链表的尾部元素，如果需要进行删除，相对比较麻烦，因为要定位它的话，首先需要得到这个单链表对象的倒数第二个元素，这样倒着推到head，就会发现我们需要从头开始。如果要进行设计的话，可以存储一个特殊的指向tail的变量，这样就可以在push与pop中随时改变指向，时间复杂度也就变成了常量级别。

1. **双链表相对于单链表付出的存储代价是什么？能够达到什么效果？**

答：需要存储左右的引用。增加量是：地址变量存储空间 \* （元素个数 + 1）。带来的效果就是增加了一个自由度，可以进行反向指引。

1. **高频度操作选择或设计合适的存储结构是数据结构与算法课程的目标之一，试举例解释。**

答：很显然，如果高频度操作的复杂度很高，那么整个系统无疑会很慢。比如说，在队列结构中，需要经常出队尾部元素，这个时候用单链表就不太好，最好是用双链表

1. **为降低高频度操作的计算复杂度，需要增设适当的属性变量，试举例解释。**

答：增加状态变量，或者说是属性变量，是经常需要的。比如，对待数组，我们经常需要知道目前这个动态数组的长度，那么遍历一遍得到长度显然不好，这个时候可以设置一个值，专门用来实时记录长度。

1. **某些应用（例如数据库）而言，排序（sort），查找（search），选择（select），遍历（traverse）也是高频度基本操作，试解释这些操作。**

答：排序即根据某一原则，对同类型元素进行顺序划分；查找，即搜索集合中的目标元素，选择是选择其中某一类的符合某个条件的元素，遍历即访问完整个集合。

1. **不同的应用系统而言，通常有不同的高频度操作和低频度操作，高频度操作构成了关键操作。为关键操作设计相应的高效数据结构与高效算法构成设计关键，试举例说明。**

答：高频操作的时间复杂度是决定总体的时间复杂度的。可知，如果最常用的几个时间复杂度很低，那么整体的性能将很不错。比如，关联存取，经常需要进行key，value配对，这时，把配对工作做成常量级别。

1. **什么是算法?**

答：可以一步一步执行的，能在有限时间内完成的过程。

1. **衡量算法的好坏？(资源占用量，计算复杂度，时间复杂度)**

答：少占用内存，能完成任务，时间符合需求。

1. **如何衡量数据结构的好坏？(高频度操作的计算复杂度)**

答：数据结构直接作用于算法，好的结构有利于算法的实施，数据管理和存取有序高效，所配套的算法的时空复杂度低。

1. **解释算法运行时间的实验研究的局限性与合理性。**

答：有的算法没法实验，而且算法实验起来，与平台有很大关系，受约束很大，影响变量较多，所得数据有很多噪声，而且往往需要做多次实验然后用统计学方法进行分析，比较繁琐，代价较大。

1. **解释算法运行时间的渐近分析的合理性与局限性。**

答：渐进分析，也就是理论分析，按照基本的常量时间复杂度的操作数量与输入规模之间的关系来衡量算法好坏。

1. **列出常见的基本操作。**

答：加法，判断，交换值

1. **如果一个程序的运行时间中出现以 *b* 为基数的对数函数，通常是与下面的程序结构有关:（1）程序当中反复出现 *b* 个分支，程序只需走其中一个分支。（2）问题的规模随之减小为 1 / b。试举例说明。**

答：二元搜索。

1. **使用最坏情形时间复杂度的合理性。**

答：分析最坏的情况，有助于我们分析软件在遇到最坏情况的性能，有利于开发出在任何情况下都能胜任的软件。

1. **解释大 *O* 表示法**

答：最坏情形。

1. **解释大*Ω*表示法**

答：最好情形。

1. **解释大*Θ*表示法**

答：中间情形。

1. **如果选择最坏情形时间复杂度并且采用大 *O* 表示法，则对算法运行时间*f*(*n*)=*O*(*g*(*n*)) 可以大致做出什么判断？**

答：存在一个常数*c*，*f* (*n*) <= *c* \* *g*(*n*)