山东大学网络空间安全学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202400460042 | 姓名：江李阳 | | 班级：1班 |
| 实验题目：使用线性表结构实现多项式求和运算、将两个正序的有序链表合并成一个逆序的有序链表; 将两个正序的有序链表（数值为整数，从小到大排列）合并成一个逆序的有序链表（数值从大到小排列） | | | |
| 实验学时：2 | | 实验日期：2025.2.28 | |
| 实验目的：理解线性表数据结构的基本原理与特性，深入理解线性表原理，掌握顺序储存与链式储存特性，熟悉其在实际运算中的运用，锻炼算法设计与编程能力。对比理解顺序表和链表这两种线性表的时间开销和空间开销的差别，明确各自的优缺点和应用场景。 | | | |
| 硬件环境：内存：16384MB 处理器：13th Gen intel(R) Core(TM) i5-13500HX (20 CPUs),~2.5GHz | | | |
| 软件环境：windows系统，visual studio | | | |
| 实验步骤与内容：  分析问题(必做)：  题目要求实现两个多项式的加法运算，由于各个多项式由一个个前后的单项式组成，因此考虑使用线性结构储存多项式。每个单项式由次数和系数构成，线性表中每个元素要能储存这两部分信息。  简化问题模块为：储存多项式->输入多项式信息->求和运算->输出和多项式信息  顺序表：   1. 储存：使用整型数组，由于数组的下标从0开始递增至n-1,类似多项式中的单项式的次数特点。因此通过数组中的标储存单项式的次数，对应的数值储存系数，当不存在这个系数的项时，令系数为0即可。通过传入未知数的最大次数n开辟大小为n+1的整型数组（对应下标0~n,下标0即次数0，对应常数项）。 2. 运算：   构建和多项式，长度根据两个相加的多项式的最高次数来确定开辟的和多项式的大小。从最低次数位开始遍历数组并相加，得到和多项式的对应次数项的系数。多出的部分直接把系数赋值给和多项式即可。   1. 输入和输出：   多项式类包括储存多项式的数组arr和记录数组长度的整型变量len,含参最高次数n构造函数为arr开辟大小为n+1的数组并提示输入系数。  输出函数遍历多项式，把系数不为零的项的系数和次数分别输出。   1. 代码实现：   根据以上模块化思路编写代码，分别实现构造Arr类，输入多项式，多项式求和，输出多项式并测试输入样例。  链表：   1. 储存：   将多项式中的每个单项式视为一个节点Node，Node类中有属性int型系数coefficient、无符号整型（unsigned int，因为次数总是非负的）次数frequency以及后继节点next。多项式用类Linkedlist储存，属性有头节点Node型指针head。   1. 运算：   分别将从两个多项式头节点开始往后读取多项式的每一项。由于是按次数从高到低输入的，因此如果两个多项式的最高次数相等，就直接将系数相加，同步地往后读取，否则就直接将次数高的那一项的系数赋给和多项式，并往后读取这个次数较高的多项式的项，直到出现两个多项式有项次数相同时，开始同步运算求和并同步往后读取。当其中某个多项式读完，开始出现nullptr时，直接将另一个还有项的多项式赋给和多项式，直至两个多项式都读完为nullptr。其中在系数相加时做了验证，只有当系数和不为0时才将这一项计入和多项式。   1. 输入输出：   利用节点Node的含参数coefficient、frequency的构造函数创建节点，Linkedlist类中封装initiallist（）函数，提示输入多项式的项数n,再循环n次调用Node含参构造函数，完成时提示多项式输入成功。其中为了每次在链表末尾插入节点，封装一个返回尾节点的gettail()函数。输出则遍历链表，分系数和次数输出每个节点的信息，直至到尾节点。   1. 代码实现;   分别完成以上模块的代码。核心部分为求和add（）函数，按两个多项式最高次数区分。调试并验证测试输入样例。  总结比较：  1.顺序表实现的时间开销是O(max(m,n)),m、n分别是两个待求和多项式的最高次数+1(因为在相加部分先遍历了一遍较短的多项式，再对较长的多项式的多出部分遍历，总的时间开销是O(max(m,n))),空间开销是O(max(m,n))  2.链表实现的时间开销是O((p+q)^2),空间开销是O(p+q),p、q分别是两个多项式的项数。  对比可知为保证数组能存下多项式。根据多项式的最大次数n，开辟n+1的空间(次数最多从0~n)，这样能保证存下，但由于部分次数项不存在，用0来存储系数，导致了空间浪费。而链表在储存多项式时，不会提前开辟空间，而是根据输入动态添加，且和多项式的某项系数为0时不会将这一项存入，保证了空间充分利用。但在存入数据时，每次都要遍历一遍链表找到尾节点，再在尾节点后面添加数据，导致时间开销较大。  分析问题(选做)：  题目要求将两个正序的有序链表（数值为整数，从小到大排列）合并成一个逆序的有序链表（数值从大到小排列），这里的解决问题分为两步：连接整合，排序。  简化模板为：整合->排序   1. 整合：   直接将一个链表的尾节点与另一个链表的头节点相连即可，尾节点的获取通过gettail()函数遍历链表获得。   1. 排序：   参考了数组的冒泡排序法，通过两边循环链表，挨个比较链表中相邻的而两个节点的数据的大小，判断是否需要交换值，因为只有一个数据，因此只交换值，不交换节点。每次内层循环减少遍历的节点个数。   1. 代码实现：   按照上述思路编写代码，其中核心部分尾冒泡排序，每次外层循环都声明了一个布尔变量swap，用于标记是否发生了交换，如果循环遍历一次都未发生交换，说明这个链表已经有序，直接退出循环，减少遍历次数。  最后测试代码功能。  总结：这里的关键在于使用了冒泡排序法，时间开销为O(n^2)，如果充分利用已知两个链表已将排好序的特点，可以进一步优化，降低时间开销，但这样算法就不具有通用性了。 | | | |
| 结论分析与体会：本次实验加深了我对线性结构中数组和链表的用法和特点有了进一步的理解，在实际问题中对比两者的优缺点。同时我还初步利用模块化思想，在面对较复杂的问题时，把它看成是一个个简单功能的模块的结合，通过逐个解决这些模块达到最终解决问题的方法，把问题简单化，在问题解读和具体代码实现两个方面都得到了便利。通过编写代码还锻炼了我的实践能力，为日后解决更复杂的问题打下基础。同时理解了算法的时间开销和空间开销之间的相互转换关系，如为了获得链表的尾节点通过遍历链表会增大时间开销，但通过维护一个尾节点就可以把这个时间开销降为O(1),但会增加空间开销，在实际问题中应做好取舍，合适的算法对高效解决问题有很大决定性作用。 | | | |