

线性表

2.1 线性表的类型定义

- n个数据元素的有序序列 —— 复杂一些的，则数据元素可以由若干 数据项 组成，这时候常把数据元素称为 记录，含有大量记录的线性表称为 文件
- 线性表中的元素可以是各种各样的，但是同一线性表中的元素必定是同一特性，即属于同一数据对象
- 相邻数据元素有序偶关系，即先驱（直接先驱）、后继（直接后继），加不加直接是一个意思
- a_i 是数据元素的话，i 叫做位序
- 线性表可以根据需要增长或缩短
- 算法
 - ① 合并线性表
void union(List &La, List &Lb) —— $O(m*n)$
 - ② 有序排列线性表 合并成 有序排列线性表
void MergeList(List La, List Lb, List &Lc) —— $O(m+n)$

2.2 线性表的顺序表示和实现

- 顺序表
 - 线性表的顺序存储结构或顺序映像
 - “物理位置上相邻”
 - 一组地址连续的存储单元存储
 - 存储位置
 - 起始位置/基地址 a_1
 - $LOC(a_{i+1}) = LOC(a_i) + l$ (每个元素占 l 个单元)
 - $LOC(a_i) = LOC(a_1) + (i - 1) * l$
 - 随机存取
- 算法 —— c语言实现
 - ① 线性表的动态分配存储结构
Status InitList_Sq(SqList &L)
 - ② 顺序表的插入
Status ListInsert_Sq(SqList &L, int i, ElemType e) —— $O(n)$
newbase = (ElemType *)realloc(L.elem, (L.listsize + LISTINCREMENT)* sizeof (ElemType))
 - ③ 顺序表的删除
Status ListDelete_Sq(SqList &L, int i, ElemType &e) —— $O(n)$
 - ④ 返回第1个值与e满足compare关系的元素的位序
int LocateElem_Sq(SqList L, ElemType e, Status (*compare)(ElemType, ElemType)) —— $O(n)$
 - ⑤ 有序顺序表合并
void MergeList_Sq(SqList La, SqList Lb, SqList &Lc) —— $O(m+n)$

2.4 一元多项式的表示和相加

- 何时适用？只需要读出来计算，不需要修改 —— 顺序存储 —— 指数i隐含在系数pi的序号里面
- 何时适用？需要修改 —— 链式存储
- (系数项pi, 指数项ei)
- ① 建立一元多项式有序链表
void CreatePolyn(polynomail &p, int m)
- ② 多项式加法：Pa = Pa + Pb
void AddPolyn(polynomail &Pa, polynomail &Pb)
- ③ 多项式乘法
-
- 算法

2.3 线性表的链式表现和实现

- 线性链表/单链表
 - 非顺序映像或链式映像
 - a_1 的存储映像称为 结点
 - 数据域
 - 指针域 —— 指针/链
 - 头结点，即哨兵节点
 - 非随机存取
- 算法 —— c语言实现
 - ① 带头结点
Status GetElem_L(LinkList L, int i, ElemType, &e) —— $O(n)$
 - ② 带头结点的单链表的插入
Status ListInsert_L(SqList &L, int i, ElemType e) —— $O(n)$
 - ③ 带头结点的单链表的删除
Status DeleteInsert_L(SqList &L, int i, ElemType e) —— $O(n)$
 - ④ 带头结点的单链表的建立
void CreateList_L(LinkList &L, int n) —— $O(n)$
 - ⑤ 有序带头结点的单链表的合并
void MergeList_L(SqList &La, SqList &Lb, SqList &Lc) —— $O(m+n)$
- 静态链表
 - 用数组实现链表
 - 数据域
 - 指针域 —— int cur;
 - 分配新结点怎么操作 —— 建立一个备用链表，管理空闲结点的分配
 - 算法 —— c语言实现
 - ① 静态链表查找
int LocateElem_SL(SLinkList S, ElemType e) —— $O(n)$
 - ② 将数组空间初始化成一个链表
void InitSpace_SL(SlinkList \$space) —— $O(n)$
 - ③ 从备用空间取得一个结点
int Malloc_SL(SlinkList &Space) —— $O(1)$
 - ④ 将空闲结点链到备用链表上
void Free_SL(SLinkList &space) —— $O(1)$
 - ⑤ (A-B) U (B-A)
void different(SlinkList &space, int &S) —— $O(n*m)$
- 循环链表
 - 遍历边界：p->next == head
 - 算法 —— 合并循环链表 —— $O(1)$
- 双向链表
 - 克服找先驱的问题
 - 可以有循环表
 - 算法 —— c语言实现 —— 双向链表插入结点
Status ListInsert_Du1(DuLinkList &L, int i, ElemType e)
- 重新定义线性链表的操作 —— P37

区别在空间复杂度