班级 计算机204 学号 202007020625 姓名 於俊涛 报告日期 2021/10/19

**实验题目： 线性表的关键算法实现及性能分析**

**一、实验目的**

**1. 能够运用高级程序设计技术实现线性表及其关键算法**

**2. 能够通过对比分析的方法分析AAA、BBB等2-3个关键因素对XXX数据结构关键算法性能的影响规律**

**二、实验原理**

1. **线性表**的基本原理

定义该数据结构的逻辑结构及主要性质，给出该数据结构的抽象数据类型定义；

采用线性表的顺序存储结构。

typedef int Position;

typedef struct LNode \*List;

struct LNode {

ElementType Data[MAXSIZE];

Position Last;

};

2.关键算法设计原理及性能影响因素分析

逐一介绍相关数据结构的关键算法，从理论上分析不同因素对该算法时空性能的影响规律，如问题规模、算法设计、存储方案、递归或非递归实现等等。

**三、实验方案设计**

1.存储方案设计

顺序存储结构，所占空间较小，容量需事先分配。查找运算O（n/2）,读运算O（1），

插入运算O（n/2），最好为0，最坏为n，删除运算O（[n-1]/2）;

链式结构，容量动态分配，查找运算O（n/2）,读运算O（[n-1]/2），

插入运算O（1），最好为0，最坏为n，删除运算O（1）;

分析不同存储方案的优势和不足，选择恰当的存储结构

2.算法的设计和实现

List MakeEmpty()

{

List lst=(List)malloc(sizeof(struct LNode));

lst->Last=-1;

return lst;

}

Position Find( List L, ElementType X )

{

// List curson=L;

for(int i=0;i<=L->Last;i++)

{

if(L->Data[i]==X)

return i;

}

return ERROR;

}

bool Insert( List L, ElementType X, Position P )

{

if(L->Last==MAXSIZE-1)

{

printf("FULL");

return false;

}

if(P>L->Last+1||P<0)

{

printf("ILLEGAL POSITION");

return false;

}

for(int i=L->Last;i>=P;i--)

{

L->Data[i+1]=L->Data[i];

}

L->Data[P]=X;

L->Last++;

return true;

}

bool Delete( List L, Position P )

{

if(P>(L->Last)||P<0){

printf("POSITION %d EMPTY",P);

return false;

}

for(Position i=P;i<L->Last;i++)

{

L->Data[i]=L->Data[i+1];

}

L->Last--;

return true;

}

int n;

Status ListCreate\_Sq(SqList &L)

{

scanf("%d",&n);

L.elem = (ElemType\*)malloc(n\*sizeof(ElemType));

if(!L.elem)

exit(OVERFLOW);

L.length = n;

L.listsize = n;

for(int i = 0; i < n; i++)

scanf("%d", &L.elem[i]);

return OK;

}

void ListReverse\_Sq(SqList &L)

{

int temp;

for(int i = 0; i < n/2; i++)

{

temp = L.elem[i];

L.elem[i] = L.elem[n-1-i];

L.elem[n-1-i] = temp;

}

}

List Delete( List L, ElementType minD, ElementType maxD ){

int last=-1,a=L->Last;

for(int i=0;i<=L->Last;i++){

if(L->Data[i]<=minD || L->Data[i]>=maxD){

last++;

L->Data[last]=L->Data[i];

}

else a--;

}

L->Last=a;

return L;

}//时间复杂度O（n）

选用伪代码或流程图对ADT中的操作进行描述，并分析算法性能

3.测试方案

List MakeEmpty(); 创建并返回一个空的线性表。

Position Find( List L, ElementType X );返回线性表中X的位置。若找不到则返回ERROR；

bool Insert( List L, ElementType X, Position P );将X插入在位置P并返回true。若空间已满，则打印“FULL”并返回false；如果参数P指向非法位置，则打印“ILLEGAL POSITION”并返回false；

bool Delete( List L, Position P );将位置P的元素删除并返回true。若参数P指向非法位置，则打印“POSITION P EMPTY”（其中P是参数值）并返回false。

Status ListCreate\_Sq(SqList &L);用于创建一个顺序表

void ListReverse\_Sq(SqList &L);在不引入辅助数组的前提下将顺序表中的元素进行逆置

List Delete( List L, ElementType minD, ElementType maxD );删除所有值大于min而且小于max的元素。删除后表中剩余元素保持顺序存储，并且相对位置不能改变。

设计各个操作的测试数据用例和预期实验结果