复习提纲

第一章　数据结构概述

基本概念与术语（ **P3**）

1.　数据结构　是一门研究非数值计算程序设计问题中计算机的操作对象以及他们之间的关系和操作的学科　**.**

2.　数据　是用来描述现实世界的数字　**,**字符**,**图像**,**声音**,**以及能够输入到计算机中并能被计算机识别的符号的集合

2. 数据元素　是数据的基本单位

3. 数据对象　相同性质的数据元素的集合

4. 数据结构　包括三方面内容 **:**数据的逻辑结构 **.**数据的存储结构 **.**数据的操作 **.**

（**1**）数据的逻辑结构　指数据元素之间固有的逻辑关系　**.**

（**2**）数据的存储结构　指数据元素及其关系在计算机内的表示

(3) 数据的操作　指在数据逻辑结构上定义的操作算法　**,**如插入**,**删除等 **.**

5.时间复杂度分析

1、名词解释：数据结构、二元组

2、根据数据元素之间关系的不同，数据的逻辑结构可以分为
集合、线性结构、树形结构　和图状结构 四种类型。

3、常见的数据存储结构一般有四种类型，它们分别是　　___顺序存储结构 _____、____链式存储结构 _____、___索引存储结构 _____和___散列存储结构 _____。

4、以下程序段的时间复杂度为　**___O(N2)_____**。

```
int i,j,x;
for(i=0;i<n:i++)      n+1
    for(j=0;j<n;j++)      n+1
        x+=i;
```

第二章　线性表

1. 顺序表结构　由 **n(n>=0)**个具有相同性质的数据元素　**a1,a2,a3** …… **,an**组成的有穷序列

```
//顺序表结构
#define MAXSIZE 100
typedef int DataType;
Typedef struct{
DataType items[MAXSIZE];
Int length;
}Sqlist,*LinkList;

//初始化链表
void InitList(LinkList *L){
    (*L)=(LinkList)malloc(sizeof(LNode));
    if(!L){
    cout<< "初始化失败！";
    return;
    }
    (*L)->next=NULL;
}

//插入数据
void InsertList(LinkList L,int pos,DataType x){
    LinkList p=L,q;
    int i=0;
    while(p&&i<pos-1){
        p=p->next;
        i++;
    }
    if(!p||i>pos-1){
        cout<< "插入位置错误　";
        return;
        }
    InitList(&q);
    q->next=p->next;
    p->next=q;
    q->data=x;
}

//销毁链表
```

```
void DestoryList(LinkList L){
    LinkList t;
    while(L){
        t=L;
        L=L->next;
        free(t);
    }
}

//遍历链表
void TraverseList(LinkList L){
    LinkList t=L;
    while(L){
        t=t->next;
        cout<<t->data<< ”  ”;
    }
    cout<<endl;
}

//删除元素
void DeleteList(LinkList L,int pos){
    LinkList p=L,q;
    int i=0;
    while(p&&i<pos-1){
        p=p->next;
        i++;
    }
    if(!p||i>pos-1){
    cout<< 删除位置错误 !! ”;
    return;
    }
    q=p->next;
    p->next=q->next;
    free(q):
}
```

第三章　栈和队列

1. 栈

（**1**）　栈的结构与定义

（**2**）　顺序栈操作算法：入栈、出栈、判断栈空等

（**3**）　链栈的结构与定义

2. 队列

（**1**） 队列的定义

1、一个栈的入栈序列为 “ ABCDE”，则以下不可能的出栈序列是（）

A. BCDAE B. EDACB C. BCADE D. AEDCB

2、栈的顺序表示仲，用 TOP 表示栈顶元素，那么栈空的条件是（）

A. TOP==STACKSIZE B. TOP==1 C. TOP==0 D. TOP==-1

3、允许在一端插入，在另一端删除的线性表称为 ____队列____。插入的一端为____队尾____，删除的一端为 _____队头___。

4、栈的特点是 ____先进后出 ____，队列的特点是 ____先进先出 ____。

5、对于栈和队列，无论他们采用顺序存储结构还是链式存储结构，进行插入和删除操作的时间复杂度都是 ____O(1)____。

6、已知链栈 Q，编写函数判断栈空，如果栈空则进行入栈操作，否则出栈并输出。（要求判断栈空、出栈、入栈用函数实现）

```
//判断栈空 (完成题目要求 )
void EmptyStack(LinkStack Q){
    LinkStack t;
    char x=a; //假设链栈存储字符型数据
    if(Q->next){t=Pop(Q,x);cout<<t->data;}
    else Push(Q,x);
}

//初始化栈
void InitStack(LinkStack *Q){
    *Q=(LinkStack)malloc(sizeof(SNode));
    if(!Q){
        cout<< "初始化错误 ";
        return;
    }
    (*Q)->next=NULL;
}

//入栈
void Push(LinkStack Q,datatype x){
    LinkStack t;
    InitStack(&t);
    t->data=x;
    t->next=Q->next;
    Q->next=t;
```

```
}

//出栈
void Pop(LinkStack Q,datatype &x){
      LinkStack t=Q->next;
      if(!t){
            cout<< 栈为空 ,无法出栈 ”;
            return;
      }
      Q->next=t->next;
      x=t->data;
       free(t);
}
```

基本概念

数据结构的研究对象是什么？

数据，数据元素 （数据结构中讨论的 "基本单位 "、数据整体中相对独立的单位、

数据元素的特点：相对性），数据结构，数据类型和抽象数据类型 ，数据对象

数据结构是什么？

定义 ：数据元素以及它们之间存在一种或多种特定的关系。

特点 ：数据元素集合相同， 而其上的关系不同， 则构成的数据结构不同。

逻辑结构是什么？主要有哪几类？

逻辑结构 ：对数据元素之间存在的逻辑关系的描述，它可以用一个数据元素的集合和定义在此集合上的若干关系表示。

存储结构是什么？

存储结构：是数据逻辑结构在计算机中的表示和实现，故又称数据 "物理结构"。

什么是算法？

定义：是对问题求解过程的一种描述，是为解决一个或一类问题给出的一个确定的、有限长的操作序列。

五大特性： 有穷性、确定性、可行性、输入、输出

线性表

线性表的定义 **?**

线性表是由 **n(n≥0)**个属性相同数据元素 $a_1,a_2\cdots a_n$ 组成的一个有限序列，

线性表或是空表，或可以表示为 $A=(a_1, a_2, \cdots, a_i, \cdots, a_n)$ 其中 $a_i(i=1, 2, \cdots, n)$是线性表中的一个元素。

如何在顺序存储结构表示的线性表中实现插入元素操作？

```
int insertElement(List_Array *list_ptr, char *element)
{//把新字符串插入到线性表的最后位置
  if(list_ptr->count == LISTMAX)
        return (-1); //   到达最大大小
         else {
            strcpy(list_ptr->list[list_ptr->count],element);
           list_ptr->count++; //下一个元素
         return (1); //   成功返回
  }
}
```

如何在顺序存储结构表示的线性表中实现元素删除操作？

```
int deleteElement(List_Array *list_ptr, int pos)
{ int k;
  //检查下标  pos位置上是否存在数据
  if (pos < 0 || pos > list_ptr->count-1)
        return (-1); // 出错
  else {//将  pos位置后所有元素向前移动
        for (k = pos; k < list_ptr->count - 1;k++)
              strcpy(list_ptr->list[k],list_ptr->list[k+1]);
        list_ptr->count--;
        return (1); //   删除成功
  }
}
```

如何在顺序存储结构表示的线性表中找到元素后继？

物理地址上紧接着该元素后一个级即该元素的后继

用数组的下标加 **1** 即可找到 **.**

如何在顺序存储结构表示的线性表中找到元素前驱？

物理地址上紧接着该元素前一个即该元素的前驱

用数组下标减 **1** 即可找到

链表

什么是链接存储结构？

通过指针管理的一组存储单元,　　（这组存储单元的内存地址可以是连续的，也可以是不连续的）。

链接存储结构中的每个存储单元称为　　“结点 ”，结点包含一个数据域和一个指针域；

链接存储结构中的结点通过指针域指示后继结点的内存地址；

访问链接存储结构通常由第一个结点开始，逐一访所有结点。

如何将新结点添加到单链表中？

表头位置

① **Node *t=new Node;**② **t->Data=d;** ③ **t->next=head;**④ **head=t;**

表尾位置

① **Node *t =new Node;**② **t->data=d;**③ **last->next=t;**④ **last=t;**

两个结点中间

查找单链表中指定结点？

设置一个跟踪链表结点的指针 **p**，初始时 **p** 指向链表中的第一个结点， 然后顺着 **next** 域依次指向每个结点，每指向一个结点就判断其是否等于指定结点，若是则返回该结点地址。 否则继续往后搜索， 直到 **p** 为 **NULL** ，表示链表中无此元素，返回 **NULL** 。算法的时间复杂度为 **O(n)**。

如何删除单链表中的结点？

要删除链表中第 **i** 个结点，

首先在单链表中找到删除位置 **i－1** 前一个结点，并用指针 **p** 指向它，指针 **t** 指向要删除的结点。

将指针 **p** 所指结点的指针域修改为所 **t** 指结点的后继结点的地址。

从链表中删除链接关系后的结点需动态的释放（ **delete)**。

Node *t,*p; ① **t=p ->next;** ② **p->next=t->next ;** ③ **delete t;**

如何用单链表表示线性表？

如何实现链接存储结构表示的线性表的操作？

插入、删除、查找

栈和队列

什么是栈？

栈（**Stack**）是限定只能在表的一端进行插入和删除操作的线性表。

栈中允许插入和删除运算的一端称作 栈顶 **(top)**

不允许插入和删除的另一端称作栈底 **(bottom)**

如何实现栈的入栈和出栈操作？

栈顶表示（两种存储结构）

入栈、出栈

什么是队列？

队列 **(queue)**是限定只能在表的一端进行插入，在表的另一端进行删除的线性表、队尾 **(rear)**—— 允许插入的一端 、

队头 **(front)**—— 允许删除的一端

如何实现队列的入队和出队操作？

队头、队尾（两种存储结构）

循环队列已满标志

队列已满标志

bFull=true; （表示队列为满）

bFull=false; （表示队列为空）

设空单元

(rear+1)%Max==front （表示队列为满）

front=rear(表示队列为空）

栈的应用

算术表达式三种形式

前缀表达式 **=**运算符＋操作数 **1**＋操作数 **2**

中缀表达式 **=**操作数 **1**＋运算符＋操作数 **2**

后缀表达式 **=**操作数 **1**＋操作数 **2**＋运算符

中缀表达式、后缀表达式

中缀表达式转换成后缀表达式

排序

什么是直接插入排序法？

排序过程 、代码如何实现

依次将待排序数据元素按其关键字的大小插入到有序区的适当位置上.

什么是简单选择排序法？

排序过程、代码如何实现

将乱序的序列分成两组 ,一组有序 (刚开始元素个数为 0),一组无序 .每次都选取无序区域中关键字最小的数据元素插入到有序区最后面 .

什么是快速排序法？

排序过程 、如何实现

选取一个元素为中轴 ,然后将无序序列中大于中轴的元素一道中轴元素右边 ,小于中轴的元素移到中轴的左边 .移动完后 ,将中轴元素的左边的无序序列和右边的无序序列分别重复以上过程 (递归).直到全部有序为止.

什么是二路归并排序法？

如何归并两个有序表

排序过程 、如何实现

先将相邻的两个有序子序列合并 ,并存放于一个临时数组中 ,合并完成后再复制回原序列 .合并时 ,依次比较两个子序列相对应的数据元素的关键字值 ,将关键字值较小的数据元素复制到临时数组中 ,然后再比较下一个关键字 .反复如此 ,直至一个子序列复制完成 ,再将另一个非空的子序列剩余部分复制到临时数组中 .

内部查找

什么是二分查找（折半查找）？

前提条件

查找的表为有序表

查找过程如何实现？

首先确定待查找区间的中间位置 ,然后把待查找关键字 key 与中间位置上数据元素的关键字 mkey 做比较 ;若 key=mkey,则查找成功 ;若 key<mkey,则在待查找区间的前半自取件继续这样的查找 ;若 key>mkey,则在待查找区间的后半自取件继续这样额查找 ;直到找到或查找区间的上界小于下届 (没找到)为止.

什么是散列查找？

冲突、同义词

冲突：在构造哈希表时 ,不同的关键字可能得到同一个哈希地址 ,这种现象称为冲突 .在构造哈希表时 ,冲突在所难免 .

同义词：把具有不同关键字而有相同哈希地址的数据元素称作同义词.

开放地址法解决冲突

开放定址法是使用某种探查技术在哈希表中形成一个探查序列 ,当冲突发生时 ,沿此序列举个单元地查找 ,直到找到空闲单元地址的方法.方法主要有：

线性探查法

平方探查法

双哈希函数探查法

链接法解决冲突

做法是：

把所有关键字为同义词的数据元素存在同一个单链表中 .

树与二叉树

什么是树？

根、树的度、结点

树是由 **n(n>=0)**个元素构成的有限集合 .其中,**n=0** 称为空树 ;**n>0** 称为非空树 .对于任意一棵非空树 ,都满足一下条件 ：

1. 有且仅有一个称为根的节点 ,它比较特殊 ,没有前驱结点 ；
2. 其余结点被分成 **m(m>=0)**个互不相交的有限集 **T1,T2, …..Tm,** 其中每一个集合 **Ti(i<=m)** 优势一棵树 ,称为根的子树 .

书中所有结点的度的最大值称为树的度 .

树中每个数据元素存放的空间称为结点 .这和链表中的结点一样.

双亲结点、叶子结点、兄弟结点

结点的前驱称为该结点的双亲结点 .

度为 **0** 的结点称为叶子结点

具有同一双亲的孩子结点互称为兄弟结点 .

树的四个性质

性质 **1** 树中的结点等于所有结点的度数加 **1**

性质 **2** 度为 **k** 的树中第 **i** 层上至多有 k^{i-1} 个结点

性质 **3** 深度为 **h** 的 **k** 叉树至多有 $(k^h-1)/(k-1)$ 个结点

性质 **4** 具有 **n** 个结点的 **k** 叉树的最小深度为 $(\log_k(n(k-1)+1)$

什么是二叉树？

二叉树的四个性质

性质 **1** 二叉树上的终端结点等于双支结点数加 **1**

性质 **2** 二叉树中第 **i** 层上至多有 2^{i-1} 个结点

性质 **3** 深度为 **h** 的二叉树至多有 2^h-1 个结点

性质 **4** 对完全二叉树中编号为不的结点 **(1≤i ≤n,n 1,n**为结点数 **)**：若 **i ≤** $\lfloor n/2 \rfloor$**,** 即 **2i ≤ n,** 编号为 **i** 的结点为分支结点否则为叶子结点, 若 **n** 为奇数, 则树中每个分支结点既有左孩子又有右孩子，若 **n** 为偶数，则编号最大的分支结点 **(**编号为 **n/2)**只有左孩子，没有右孩子，其余分支结点左、右孩子都有，若编号为 **i** 的结点有左孩子，则左子结点的编号为 **2i**；若编号为 **i** 的结点有右孩子则右子结点为 **2i+1**，除树根结点外， 若一个结点的编号为 **i**，则它的双亲结点的编号为 $\lfloor i/2 \rfloor$

性质 **5** 具有 **n** 个（**n>0**）结点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor+1$

二叉树的遍历方法（先序、中序、后序）

先序遍历 **:**头结点 左子树 右子树

中序遍历 **:**左子树 头结点 右子树

后续遍历 **:**左子树 右子树 头结点

附**:**层序遍历 **:**按照每个元素的下标依次遍历

什么是二叉搜索树

如何生成二叉搜索树

二叉搜索树或者是空树 ,或者是具有以下性质的二叉树 :

1.若左子树非空 ,则左子树上所有结点的关键字值均小于它的根节点的关键值.

2.若右子树非空 ,则右子树上所有结点的关键字值均大于它的根节点的关键值.

3.左右子树本身又是一颗二叉排序树 .

如何在二叉搜索树实现数据查找

类似于折半查找 ,过程为 :

设待查找数据元素为 a ,要比较的二叉排序树根节点的关键字值为 b

若 a=b,则查找成功 .

若 a<b,则继续查找左子树 .

若 a>b,则继续查找右子树 .

图

什么是图？ **((**重点要看看书 **))**

有向图、无向图、路径

由没有方向的边构成的图称为无向图 .

由有有方向的边构成的图称为有向图 .

由顶点 vi 经过一系列的边或弧能够到达顶点 vj,则称这一系列的边或弧为顶点 vi 到顶点 vj 的路径 .

有向边、无向边

有方向的边称为有向边 ,一般称为弧 .有向边的始点称为弧尾 ,有向边的终点称为弧头

没有方向的边称为无向边 ,简称边 .

图的存储结构

重点 **:**邻接矩阵、邻接表

邻接矩阵式表示顶点之间相邻关系的矩阵 .它以矩阵的行和列表示顶点 ,以矩阵中的元素表示边或弧 .邻接矩阵式图的顺序存储结构 **.(P216)**

邻接表是图的链式存储结构 .邻接表由边表和顶点表组成 **.(P221)**

图的遍历方法？

深度优先遍历 **(P226)**

广度优先遍历 **(P228)**