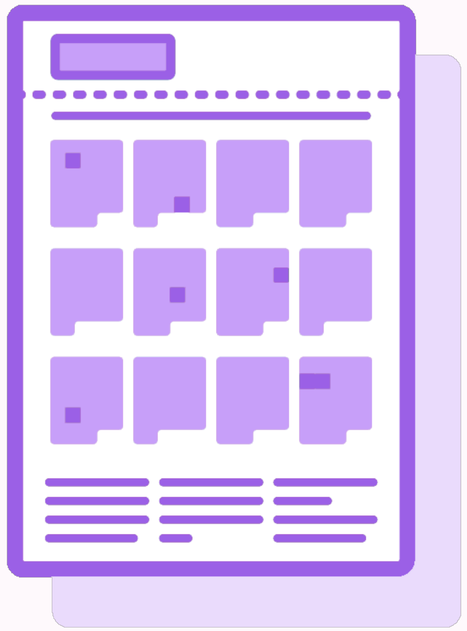
查找

斐多课堂 Phaedo Classes 数据结构 不挂科 第七讲



3大模块

7道题目

查找

模块1 / 基本概念

模块2 / 三种结构

模块3 / 效率指标

基本概念

小节1 / 定义

小节2 / 概念

基本概念

小节1 / 定义

小节2 / 概念

查找的定义

在数据集合中寻找满足某种条件的数据元素的过程称为查找

查找表的定义

用于查找的数据集合称为查找表

由同一类型的数据元素组成，可以是一个数组或链表等数据类型

基本概念

小节1／定义

小节2／概念

对查找表的操作

1. 查询某个特定的数据元素是否在查找表中

2. 检索满足条件的某个特定的数据元素的各种属性

3. 在查找表中插入一个数据元素

4. 从查找表中删除某个数据元素

静态查找&动态查找

静态查找表：如果一个查找表的操作只涉及操作1和操作2，无需动态地修改查找表

顺序查找、折半查找、散列查找等

动态查找：需要动态地插入或删除的查找表

二叉排序树的查找、散列查找等

关键字

数据元素中唯一标识该元素的某个数据项的值，使用基于关键字的查找，查找结果唯一。

平均查找长度

所有查找过程中进行关键字的比较次数的平均值

例题7-1

顺序查找适合于存储结构为（ ）的线性表。

A. 顺序存储结构或链式存储结构　　B. 散列存储结构

C. 索引存储结构　　D. 压缩存储结构

解析7-1

A

顺序查找是指从表的一端开始向另一端查找。它不要求查找表具有随机存取的特性，可以是顺序存储结构或链式存储结构

例题7-2 由n个数据元素组成的两个表：一个递增有序，一个无序。采用顺序查找算法，对有序表从头开始查找，发现当前元素已不小于待查元素时，停止查找，确定查找不成功，已知查找任一元素的概率是相同的，则在两种表中成功查找（ ）。

A. 平均时间后者小
B. 平均时间两者相同
C. 平均时间前者小
D. 无法确定

解析7-2 **B**

对于顺序查找，不管线性表是有序的还是无序的，成功查找第一个元素的比较次数为1，成功查找第二个元素的比较次数为2，以此类推，即每个元素查找成功的比较次数只与其位置有关（与是否有序无关），因此查找成功的平均时间两者相同。

三种结构

小节1 / 线性结构

小节2 / 树形结构

小节3 / 散列结构

三种结构

小节1 / 线性结构

小节2 / 树形结构

小节3 / 散列结构

线性结构「顺序查找」

基本思路

1. 从表的一端开始，顺序扫描线性表，依次将扫描到的关键字和给定值k比较

2.若当前扫描的关键字与k相等，则查找成功

3. 若扫描结束后，仍未发现关键字等于k的记录，则查找失败

线性结构「顺序查找」

例题6-5 数组a[]中有n个整数，没有次序，数组从下标1开始存储，请写出查找任一元素k的算法，若查找成功，则返回元素在数组中的位置；若查找不成功，则返回0。计算其平均查找长度以及算法的时间复杂度。

解析6-5 分析：
元素没有顺序，因此要扫描数组中的所有元素，逐个和k进行比较，相等时证明查找成功，返回元素位置；如果扫描结束时仍没有发现和k相等的元素，则查找不成功，返回0。
由此可以写出代码：

```
int Search(int a[],int n,int k)
{
int i;
for (i=1;i<=n;++i)
  if(a[i]==k)
    return i;//查找成功返回i
  return 0;//查找成功返回0
}
```

线性结构「折半查找」

折半查找要求线性表是有序的

基本思路:

1. 设R[low, ..., high]是当前的查找区间
2. 确定该区间的中间位置mid = (low+high)/2
3. 将待查的k值与R[mid]比较
4. 若相等，则查找成功，并返回该位置，否则需确定新的查找区间
5. 若R[mid]>k，则由表的有序性可知R[mid, ..., high]均大于k，因此若表中存在关键字等于k的记录，则该记录必定在mid左边的子表R[low, ..., mid-1]，故新的查找区间是左子表

线性结构「分块查找」

数据结构

又称为索引顺序查找，把线性表分成若干块，每一块中的元素存储顺序是任意的，但是块与块之间必须按照关键字大小有序排列。

对顺序表进行分块查找需要额外建立一个索引表，表中的每一项对应线性表中的一块，每个索引项都由键值分量和链值分量组成。

键值分量存放对应块的最大关键字，链值分量存放指向本块第一个元素和最后一个元素的指针。

线性结构「分块查找」

算法描述

1. 确定待查找的元素属于哪一块

2. 在块内精确查找该元素

例题7-3 折半查找和二叉排序树的时间性能（ ）。

A. 相同　　B. 有时不相同

C. 完全不同　　D. 无法比较

解析7-3 **B**

折半查找的性能分析可以用二叉判定树来衡量，平均查找长度和最大查找长度都是$O(log_2 n)$；二叉排序树的查找性能与数据的输入顺序有关，最好情况下的平均查找长度与折半查找相同，但最坏情况即形成单支树时，其查找长度为O（n）。

三种结构

小节1 / 线性结构

小节2 / 树形结构

小节3 / 散列结构

树形结构「二叉排序树」

定义

1. 若左子树不空，则左子树上左右关键字的值均小于根关键字的值

2. 若右子树不空，则右子树上所有关键字的值均大于根关键字的值

3. 左右子树又各是一棵二叉排序树

树形结构「二叉排序树」

存储结构

常用二叉链表进行存储

```
typedef struct BTNode
{
int key;//这里将data改为key，代表关键字
struct BTNode *lchild;
struct BTNode *rchild;
}BTNode
```

树形结构「二叉排序树」

查找关键字

将待查关键字先和根结点中的关键字比较，如果相等则查找成功；
如果小于则到左子树中查找；
如果大于则到右子树中查找；
如果来到了结点的空指针域，则查找失败。

树形结构「二叉排序树」

插入关键字

在查找关键字的算法基础上进行修改，在来到空指针的时候将关键字插入。
在插入过程中，如果待插入关键字已经存在，则返回0，插入不成功；
如果插入关键字不存在，则插入，返回1，插入成功。

树形结构「二叉排序树」

构造算法

建立一棵空树，将关键字逐个插入到空树中。

```
void CreateBST(BTNode *&bt, int key[],int n)
{
int i;
bt=NULL;//将树清空
for(i=0;i<n;++i)//调用插入函数，逐个插入关键字
    BSTInsert(bt,key[i]);
}
```

树形结构「二叉排序树」

删除关键字

1. 若p结点为叶子结点，直接删除

2. 若p结点只有右子树而无左子树，或者只有左子树而无右子树，将p删除，并将p的子树直接连接在原来p与其双亲结点f相连的指针上

3. 若p结点既有左子树又有右子树，先沿着p的左子树根结点的有指针一直往右走，直到其右子树的最右边的结点r，将p中的关键字用r中的关键字代替，若r是叶子结点，直接删除；若r是非叶子结点，则按照2的方法删除r

树形结构「平衡二叉树」

概念

以树中左右的结点为根的树的左右子树高度之差的绝对值不超过1。

平衡因子

一个结点的平衡因子为其左子树的高度与右子树的高度的差。

树形结构「平衡二叉树」

建立

与建立二叉排序树的过程基本一样，但每插入一个新的关键字都要进行检查，看树中是否出现了平衡因子绝对值大于1的结点，如果失去平衡需要进行平衡调整。

树形结构「平衡二叉树」

平衡调整

1. 找出插入新结点后失去平衡的最小子树

2. 调整这棵子树，使之成为平衡子树

失去平衡的最小子树：以距离插入结点最近，且以平衡因子绝对值大于1的结点为根的子树

树形结构「B树」

基本概念

B树的阶：B树中所有结点孩子结点个数的最大值，通常用m表示

条件：

1. 树中每个结点至多有m棵子树
2. 若根结点不是终端结点，则至少有两棵子树
3. 除根结点外的所有非叶结点至少有m/2上取整棵子树
4. 所有的非叶结点的结构
5. 所有的叶结点都出现在同一层次上，并且不带信息

树形结构「B树」

查找关键字

比较key与根结点中的关键字，若key等于k[i]，则查找成功

若key<k[i]，则到p[n]所指示的子树中进行查找

若key>k[i]，则到p[n]所指示的子树中进行查找

若k[i]<key<k[i+1]，则沿着指针p[i]所指示的子树继续查找

若遇到空指针，则证明查找不成功

树形结构「B树」

插入关键字

1. 定位：利用查找算法，找出插入该关键字的最底层的某个非叶结点

2. 插入：若插入后的关键字个数小于m，则直接插入；若插入后的结点关键字大于m-1，则必须对结点进行分裂

3. 分裂：取一个新结点，将插入key后的原结点从中间位置将其中的关键字分为两部分，左部分包含的关键字放在原结点中，右部分包含的关键字放到新结点中，中间位置的结点插入到原结点的父结点中,导致其父结点的关键字个数超过了上限，则继续分裂，直至这个过程传到根结点为止，导致B树高度增1。

树形结构「B树」

删除关键字

若所删除的关键字k不在终端结点中

1. 若小于k的子树中关键字个数>m/2上取整-1，则找出k的前驱值k‘，并且用k’来取代k，再递归地删除k‘
2. 若大于k的子树中关键字个数>m/2上取整-1，则找出k的后继值k‘，并且用k’来取代k，再递归地删除k‘
3. 若前后两个子树中关键字个数均为m/2上取整-1，则直接将两个子结点合并，直接删除k

树形结构「B树」

删除关键字

若被删除的关键字k在终端结点中

1. 直接删除关键字：若被删除关键字所在结点的关键字个数>m/2上取整-1，则直接删除。

2. 兄弟够借：若被删除关键字所在结点删除前的关键字个数=m/2上取整-1，且与此结点相邻的右（左）兄弟结点的关键字个数>=m/2上取整-1，需要调整该结点、右（左）兄弟结点以及其双亲结点，以达到新的平衡。

3. 兄弟不够借：被删除关键字所在结点删除前的关键字个数=m/2上取整-1，且此时与该结点相邻的右（左）兄弟结点的关键字个数=m/2上取整-1，则将关键字删除后与右（左）兄弟结点及双亲结点中的关键字进行合并。

树形结构「B+树」

基本概念

条件：

1.每个分支结点最多有m棵子树（子结点）

2. 非叶根结点至少有两棵子树，其他每个分支结点至少有m/2上取整棵子树

3. 结点的子树个数与关键字个数相等

4. 所有叶结点包含全部关键字及指向相应记录的指针，而且叶结点中将关键字按大小顺序排列，并且相邻叶结点按大小顺序相互链接起来

5. 所有分支结点中仅包含它的各个子结点中关键字的最大值及指向其子结点的指针

树形结构「B+树」

与B树的差异

B树	B+树
具有n个关键字的结点含有(n+1)棵子树	具有n个关键字的结点只含有n棵子树
m/2上取整-1<=n<=m-1	m/2上取整<=n<=m
叶结点包含的关键字和其他结点包含的关键字包含的关键字不重复	叶结点包含了信息和全部关键字

例题7-4 下列关于m阶B树的说法中，正确的是（ ）。

A. 跟结点至多有m棵子树
B. 所有叶结点都在同一层次上
C. 非叶结点至少有m/2（m为偶数）或（m+1）/2（m为奇数）棵子树
D. 根结点中的数据是有序的

解析7-4 **C**

除根结点外的所有非终端结点至少有[m/2]棵子树。对于根结点，最多有m棵子树，若其不是叶结点，则至少有2棵子树。

例题7-4 下列叙述中，不符合m阶B树定义要求的是（ ）。

A. 根结点最多有m棵子树
B. 所有叶结点都在同一层次上
C. 各结点内关键字均升序或降序排列
D. 叶结点之间通过指针连接

解析7-4 **D**

m阶B树不要求将各叶结点之间用指针连接，选项D实际描述的是B+树。

例题7-5 B+树不同于B树的特点之一是（ ）。

A. 能支持顺序查找
B. 结点中含有关键字
C. 根结点至少有两个分支
D. 所有叶结点都在同一层上

解析7-5 **A**

由于B+树的所有叶结点中包含了全部的关键字信息，且叶结点本身依关键字从小到大顺序连接，因此可以进行顺序查找，而B树不支持顺序查找（只支持多路查找）。

三种结构

小节1 / 线性结构

小节2 / 树形结构

小节3 / 散列结构

散列结构「散列表」

基本概念：

散列表：根据关键字而直接进行访问的数据结构，建立了关键字和存储地址之间的一种直接映射关系。

散列函数：把查找表中的关键字映射成该关键字对应的地址的函数，Hash(key)=addr。

散列结构「散列表」

构造方法：

1. 定义域必须包含全部需要存储的关键字，值域的范围依赖于散列表的大小或地址范围。

2. 计算出来的地址应该能等概率、均匀地分布在整个地址空间，减少冲突的发生。

3. 尽量简单，在较短的时间内计算出任一关键字对应的散列地址。

散列结构「散列表」

构造方法

直接定址法

直接取关键字的某个线性函数值为散列地址，散列函数为H(key)=a*key+b

适合关键字的分布基本连续的情况

散列结构「散列表」

构造方法

除留余数法

假定散列表表长为m，取一个不大于m但最接近或等于m的质数p，把关键字转换成散列地址

H(key)=key%p

关键：选好p以减少冲突

散列结构「散列表」

构造方法

数字分析法

设关键字是r进制数，而r个数码在各位上出现的频率不一定相同，可能在某些位上分布均匀些，每种数码出现的机会均等

在某些位上不均匀，则选取数码分布较为均匀的若干位作为散列地址

适合于已知的关键字集合

散列结构「散列表」

处理冲突的方法

开放地址法

线性探查法

从发生冲突的地址开始，依次探查d的下一个地址

容易产生**堆积问题**

散列结构「散列表」

处理冲突的方法

开放地址法

平方探查法

设发生冲突的地址为d，则用平方探查法所得到的新的地址序列为$d+1^2$，$d-1^2$，$d+2^2$，$d-2^2$...

不能探查到所有单元，但至少能探查到一半的单元。

散列结构「散列表」

处理冲突的方法

开放地址法

链地址法

把所有的同义词用单链表链接起来

散列结构「散列表」的性能分析

查找效率

三个因素：散列函数、处理冲突的方法、装填因子

装填因子=表中记录数n/散列表长度m

例题7-6 散列查找一般适用于（ ）的情况下的查找。

A. 查找表为链表
B. 查找表为有序表
C. 关键字集合比地址集合大得多
D. 关键字集合与地址集合之间存在对应关系

解析7-6 **D**

关键字集合与地址集合之间存在对应关系时，通过散列函数表示这种关系。这样，查找以计算散列函数而非比较多方式进行查找。

例题7-7 只能在顺序存储结构上进行的查找方法是。

A. 顺序查找表
B. 折半查找法
C. 树型查找法
D. 散列查找法

解析7-7 **B**

顺序查找可以是顺序存储或链式存储;折半查找只能是顺序存储且要求关键字有序；树形查找法要求采用树的存储结构，既可以采用顺序存储也可以采用链式存储；散列查找中的链地址法解决冲突时，采用的是顺序存储与链式存储相结合的方式。

查找

斐多课堂 Phaedo Classes 数据结构 不挂科 第七讲