数据结构定义整理

数据结构：

一门研究非数值计算的程序设计问题中的操作对象，以及他们之间关系和操作等相关问题的学科

绪论

算法：解决特定问题求解步骤的描述，在计算机中表现为指令的有限序列，并且每条指令表示一个或者多个操作。

数据元素： 是组成数据的有一定意义的基本单位，通常作为整体进行处理，也被称为记录。

数据项：一个数据元素可以由若干个数据项组成，是数据不可分割的最小单位。

逻辑结构：值数据对象中数据元素之间的相互关系。

数据结构：指相互之间存在一定关系的数据元素的集合

物理结构：是指数据的逻辑结构在计算机中的存储形式。

顺序存储结构：把数据元素存放在地址连续的存储单元里，其数据间的逻辑关系和物理关系是一致的。

链式存储结构：是把数据元素存放在任意的存储单元里，这组存储单元可以是连续的，也可以是不连续的。

数据类型：是指一组性质相同的集合及定义在此集合上的一些操作的总称。

原子类型

结构类型

抽象数据类型：是指一个数学模型及定义在该模型上的一组操作。

数据描述

操作声明

数据的逻辑结构：

集合、线性结构、树、图

数据结构形式定义为:

(D,S)

D：节点的集合{a,b,c,d,e}

S：节点之间的连接 {R} R = {(a,b),(b,c),(c,d),(d,e)}

注：圆括号没有方向，尖括号有方向

算法特性：

具有0个或者多个输入

至少有一个输出

有穷性

确定性（不会出现二义性）

可行性

设计要求：

正确性

可读性

健壮性：当输入数据不当时，算法也能做出相关处理，而不是产生异常或者莫名其妙的结果

效率高

存储量低

一个算法的运行时间，依赖于算法的好坏和问题的输入规模。

线性表

定义：零个或者多个数据元素的有限序列

ADT 线性表（List）

Operation

InitList(\*L);

ListEmpty(L)

ClearList(\*L)

GetElem(L,i,\*e)

LocateElem(L,e)

ListInsert(\*L,i,e)

ListDelete(\*L,i,\*e)

ListLength(L)

endADT

顺序存储结构体

# define MAXSIZE 20

typedef struct

{

int data[MAXSIZE];

int length;

}SqList;

顺序存储的优缺点：

优点：

无须为表示表中的元素之间的逻辑关系而增加额外的存储空间

可以快速存取表中任意位置的元素

缺点：

插入删除操作需要移动大量元素

当线性表变化较大时，难以确定存储空间的容量

造成存储空间的“碎片”

链式存储结构体

typedef struct Node

{

int data;

struct Node \*next;

}Node,\*LinkList;

链式存储和顺序存储的优缺点的比较:

存储方式：

顺序存储结构用一段连续的存储单元一次存储线性表的数据元素

单链表采用链式存储结构，用一组任意的存储单元存储线性表的元素

查找：

顺序表存储结构O（1）

单链表O（n）

插入和删除：

顺序存储结构需要平均移动表长一半的元素，时间为O（n）

单链表在选出某位置的指针后，插入和删除时间仅为O（1）

空间性能：

顺序存储结构需要预分配存储空间，分大了，浪费，小了，容易溢出

单链表不需要分配存储空间，只要有就可以分配，元素的个数也不受限制

栈

定义：

限定在表尾进行删除操作的线性表.

顺序存储结构体：

# define MAXSIZE 50

typedef struct

{

int data[MAXSIZE];

int top; //用于栈顶指针

}SqStack;

队列

定义：

只允许在一端进行插入操作，而在另一端进行删除操作的线性表.

串

定义：

由零个或者多个字符组成的有限序列，又名字符串。

树和森林

度：

节点拥有的子树数称为节点的度。度为0的节点称为叶节点或者终端节点；度不为0的节点称为非终端节点或者分支节点。除了根节点以外，分支节点也称为内部节点。树的度是树内各节点度的最大值。

森林：

是m(m>=0)颗互不相交的树的集合。

二叉树：

定义：

是n（n>=0）个节点的有限集合，该集合或者为空集（称为空二叉树）,或者由一个根节点和两颗互不相交的、分别称为根节点的左子树和右子树的二叉树组成。

遍历：

从根节点出发，按照某种次序依次访问二叉树中所有节点，使得每个节点访问依次且仅被访问一次。

哈夫曼树：

定义：

给定n个权值作为n个叶子结点，构造一颗二叉树，若该树的带权路径达到最小，称这样的二叉树为最优二叉树，也成为哈夫曼树。

具有n0个节点的哈弗曼数分支数为2(n0 - 1).

线索二叉树：

定义：

指向前驱和后继的指针称为线索，加上线索的二叉链表称为线索链表，相应的二叉树称为线索二叉树

线索化：

对二叉树以某种次序遍历使其变为线索二叉树的过程。

线索化的过程就是在遍历过程修改空指针的过程。

线索二叉树是一种物理结构

平衡二叉树（AVL）：

定义：

它是一颗空树，或者左右子树的高度之差的绝对值不超过1的二叉排序树。

图

定义：由顶点的有穷非空集合和顶点之间的边的集合组成。

简单图：图中，不存在顶点到自身的边，且同一条边不重复出现

完全无向图：任意两个顶点都存在边

带权的图通常称为网。

连通图：任意两个顶点之间都是连通的。

连通分量：无向图中最大的连通子图。

强连通分量：在有向图中，如果任意一对顶点都存在路径，则称为强连通分量。

遍历：

深度优先（DFS）：栈的应用

广度优先（BFS）：队列的应用

最下生成树：生成树中权值和最小

Kruskal算法更适合稀疏图

Prim算法更适合稠密图

最短路径：两顶点之间经过的边上的权值之和最少的路径。

迪杰斯特拉算法（区别于prime算法，它是借助已经存在的路径）

路径长度：路径上各个活动持续的时间之和

关键路径：从源点到汇点具有最大长度的路径

关键活动：关键路径上的活动

路径：由顶点和相邻顶点序偶构成的边所形成的序列

AOE网：在一个表示工程的带权有向网中，用顶点表示事件，用有向边表示活动，用边上的权值表示活动的持续事件，这种有向网的边表示活动的网，我们称之为AOE网。

AOV网：在一个表示工程的有向图中，用顶点表示活动，用弧表示活动之间的优先关系，这样的有向网为顶点表示活动的网，我们称之为AOV网(有向无回路)。

设G = (V,E)是一个具有n个顶点的有向网，V中顶点的序列v1,v2…vn,满足若从顶点vi到vj有一条路径，则在顶点序列中顶点vi必在vj之前。则我们称这样的顶点序列为一个拓扑序列。

拓扑序列：其实就是对一个有向网构造拓扑序列的过程。构造时会有两个结果，如果此网的全部顶点被输出，则说明它是不存在环（回路）的AOV网；如果输出顶点少了，哪怕是一个，也说明这个网存在环（回路），不是AOV网。

拓扑序列：

设G = (V,E)是一个具有n个顶点的有向图，V中的顶点v1,v2....vn,满足若从顶点vi到vj有一条路径，则在顶点中顶点vi必在vj之前，则我们称这样的顶点序列为拓扑序列.

拓扑排序主要是为了解决一个工程是否顺序进行的问题，但是有时候我们需要解决工程完成需要的最短时间问题。

AOV和AOE网络的区别：

AOV网是顶点表示活动的网，它只描述活动之间的制约关系，而AOE网是用边表示活动的网，边上的权值表示活动持续的时间。AOE网是要建立在活动之间的制约关系没有矛盾的基础上，再来分析完成整个工程至少需要多少时间，或者为缩短完成工程所需要的时间，应当加快那些活动等问题。

查找

折半查找（二分查找）只适用于有序的顺序表

哈希查找冲突处理方法：

链地址法

开放地址法(线性探测法)

一旦发生了冲突，立马寻找下一个空的散列地址，只要散列表足够大，空的散列地址总会找到.

|  |  |
| --- | --- |
| 查找 | 时间复杂度 |
| 顺序查找 | O（n） |
| 二分查找 | O(logn) |
| 二叉排序树 | O(logn) |
| 哈希查找 | O(1) |

平均查找长度：

顺序查找：（n+1）/2；失败的查找长度是n

折半查找：

充要条件：查找表顺序存储且有序

静态查找：

对查找表只进行如下操作：

1. 查询某个“特定的”数据元素是否在查找表中
2. 检索某个“特定的”数据元素的各种属性

动态查找：

若在查找过程中同时插入查找表中不存在的数据元素，或者从查找表中删除已存在的某个数据元素。

查找效率最低的数据结构是堆。

二叉排序树：

又称为二叉查找树。它或者是一颗空树，或者具有下列性质的二叉树。

若左子树不空，则左子树上所有节点的值均小于它的根节点值；

若右子树不空，则右子树上所有节点的值均大于它的根节点的值.

它的左右子树分别为二叉排序树.

排序

基本内部排序

|  |  |  |  |
| --- | --- | --- | --- |
| 排序算法 | 平均情况 | 辅助空间 | 稳定性 |
| 冒泡排序 | O（n^2） | O(1) | 稳定 |
| 选择排序 | O（n^2） | O(1) | 稳定 |
| 直接插入排序 | O（n^2） | O(1) | 稳定 |
| 希尔排序 | O (nlogN~n^2) | O(1) | 不稳定 |
| 堆排序 | O (nlogn) | O(1) | 不稳定 |
| 归并排序 | O (nlogn) | O(n) | 稳定 |
| 快速排序 | O (nlogn) | O（logn） | 不稳定 |
| 简单选择排序（直接） | O（n^2） | O(1) | 不稳定 |

|  |  |  |  |
| --- | --- | --- | --- |
| 插入排序 | 选择排序 | 交换排序 | 归并排序 |
| 直接插入排序 | 选择排序 | 冒泡排序 | 归并排序 |
| 希尔排序 | 堆排序 | 快速排序 |  |

什么是递归程序？

一个函数在结束本函数之前，直接或者间接调用函数本身，称为递归。

递归的优点：

程序结构简单、清晰，易证明其正确性。

递归的缺点：

执行过程中占用内存空间较多，运行效率低。

递归调用过程需要借助栈这种数据结构实现。

串

next数组

next[j] = k: 表示j之前有最大长度为k的相同的前缀和后缀。

若k = 0或-1，则跳到下标为k的位置，跳过了j - next[j]。

最长前缀：

第一个字符开始，不包括最后一个字符。

最长后缀：

最后一个字符结束，不包括第一个字符

nextval数组

if next[i+1] = next[i] + 1

nextval[i] = nextval[next[i]].

若长度为n的字符串，则它拥有的子串的数目是(n+1)\*n/2.

KMP算法的时间复杂度是O（m+n）,m和n分别是主串和子串的长度.

移动原则：

主串不动，移动子串

B树：

一种平衡的多路查找树.