***题目也很重要***

很多数据结构操作都建立在遍历的基础上，如查找，插入，删除

存储结构 数组 链表 抽象类型

初始化，创建空表，next,child空指针

链表结点存在哪:动态申请单个结点的空间 ，或者把结点放数组（顺序无所谓，数据查找时不靠数组下标），然后每个结点用指针指向彼此

一、选择问题 指数 对数 级数 模运算

归纳法 反证法 递归:基准，推进 递归就是循环，或者叫指针遍历?

大O

ADT抽象数据类型：结构 ，方法 对表，集合，图处理 size

表 a1,a2 an 表大小为n 空表大小为0 ai+1是ai后继元素 ai-1是前驱元素

操作集合: 增删查改 大小 遍历获取，遍历查找 创建 第一个 最后一个 上一个 下一个 是否表末尾

当前元素指针

域

数组 长度 取数据 修改 插入 查找 反转 是否空 删除项

链表 数据 next指针 表头 读取 插入删除 创建 位置 遍历next

下标实现的链表（静态链表） next中存游标 插入 找到空的元素位

多项式ADT 系数数组（空位置0），次数递增 或者链表 多项式加减乘除微分

双链表 前指针

循环链表 尾指向头

基数排序 桶太大，多次桶排序 : 最低位优先桶排序 次低位优先桶排序（排序次数等于位数）

桶排序 留一个数组，大小等于有序列中最大的数，初始为0 每当有一个数，对应下标元素+1 数组元素等于出现次数

多重表 循环表?

栈 栈顶指针 数组栈（比较常见）

链表栈 表头，next 指针指向null

创建空栈 销毁 清空 获取栈顶 个数

栈递归

斐波数列

Bag

平衡符号 检查符号是否成对， [] () [()] :读取所有的数据，遇到括号入栈，遇到反括号出栈，

如果和反括号不对应或者栈空都报错 读完文件栈非空报错

后缀表达式（逆波兰）

计算：数入栈 遇到符号出栈两数，计算后入栈

中缀转后缀 只用+ \* () 数字直接输出 符号入栈 左括号入栈 遇到右括号弹出

遇到优先级低符号，栈中高优先级和等优先级**全部出栈 (优先级最高，遇到)才弹出**

实现函数调用 主调函数的当前位置 局部变量 寄存器状态 需要保存，使用栈

栈溢出，越过空间

尾递归与溢出 递归时 无限地开栈，保存返回地址，参数

使用goto循环和函数内部参数赋值消除 栈（此时不会无限）

队列 数组队 循环数组，队头和队尾指针

链表队

循环队列

应用:先到先用 消息循环 排队

串 比较

顺序串 空一个字节保存长度 或者末尾用/0

链串 一个或多个字符data,next

堆分配

模式匹配算法（找子串）

简单匹配： 子串首字符匹配成功后，后面的字符逐个比较 先匹配首字符，再比较后面，最少需要两个索引（主串当前，子串当前 可以加个主串对应子串头的索引 ） 匹配成功则返回或继续 匹配失败继续向下

Kmp: 要求子串首字符与后面不相等abcd 跳过成功匹配序列 从第一个不成功的主串字符继续，j归0

如果子串有重复” abcabc 跳到主串第二个**非重复块（abc）** 并从匹配不成功的子串字符开始比较（j有怎样的回退规律？）

如果aaaab呢？ Aaax

发现j回退只跟子串有关 全不相等 归0 部分相等

全过程中 主串索引不回退 子串

Next数组 next[j] [] 长度是子串长度 j是子串索引位置 ，从1开始

Next[j] 是指一轮匹配后，匹配停止位置 j对应的的回溯索引，下次匹配从next[j]开始

串操作

四、树 ——严格递归

空树的定义

回溯法

二叉查找树 递归定义 根 子树 每个子树都有根 根就是结点

N个节点 n-1条边 每个节点都连着父亲 叶结点 兄弟 祖父 孙子

路径 长度 节点到自己的路径长度为0 根到节点路径只有一条 节点深度是跟到节点的长度 节点高度是节点到其他树叶最长路径

有序树:左右子树不能交换

度:节点连的子树数

森林：不相交的树

树转二叉树

森林转二叉：先把树转二叉，二叉作为前一棵二叉右孩

二叉转树

二叉转森林

树和森林的遍历:前 后 一棵棵树来

ADT：构造空树，销毁，清空，是否为空，返回深度，根结点，某个结点值，赋值结点，返回节点左右，兄弟，插入

存在n1到n2的路径:祖先 真祖先

关于结点删除:应该删除什么?叶结点 有子树

树存储结构: 可以顺序，但是顺序不代表关系[]，只是单纯保存，加入新结点的话也是加在尾部，或者直接插空

纯链式（随机存储，空间分配比较麻烦，方便动态添加）

树实现 双亲 儿子兄弟表示法 孩子

数组 保存所有结点[a,b,c,d,e,f] 数据 指针都指向父(下标或者指针) 如果无父就是根（不方便找孩子）

【a,b,c,d,e,f,g】 两个指针 父和左孩

指向多个孩子 :关于空间分配，如果孩子数固定，可能会浪费空间给空指针 动态孩子数

:所有数据放数组[a,b,c,d,e,f,g] 然后每个元素用链表串所有孩子 firstchild data|nextchild

结合双亲表示法,[a,b,c,d,e,f,g] 包含父 firstchild data|nextchild

孩子兄弟 data firstchild rightsib 把复杂树变为二叉树

二叉树:顺序 二叉链表 头指针，left right data

生成:递归赋值 扩展二叉树 空指针返回

遍历（递归） 目录文件 先序 根左右

后序: 左右中

中序: 根开始（不访问）左中右

层序:按顺序遍历

递归的参数转换() ，要求参数有规律地变化，要有 **递归外面是形参，里面调用的时候是实参**

如果还存在子结点，就继续向下遍历，否则返回数据?

顺序二叉 :数组形成的二叉树，（一般用于完全二叉），二叉链表

满二叉树，完全二叉树（按顺序排列的）

二叉树 data +left+right指针

表达式树 ：叶节点是操作数，其他节点都是操作符 （函数式编程，数学表达式常用），存在二叉 ，也可能有单目运算（其实单目可认为是+1操作等）

每个操作符代表一个返回值，兄弟操作符连接就是返回的连接

每个符号和其叶都可以用括号括起来

中序遍历:中缀表达式 后序遍历：后缀表达式 中缀变后缀

后缀表达式变表达式树：读入表达式：操作数 就创建单节点，指针入栈 遇到操作符，两个指针弹出，和操作符节点连接，操作符节点指针入栈

树创建 如果不存在根节点，创建 ，如果存在，按照规则找空地方插 创建空树

找最大 最小 插入子树（结点） 删除

随机生成树

线索二叉树 指针域有空的，把空指针域指向前驱，后继元素（根据遍历顺序），加上一个标记（区分是子树还是线索） 相当于双向链表

线索化也是递归添加的，方便遍历

AVL平衡二叉树 （是一种排序树） 左右同高 单旋转 双旋转，

子树左右深度差不超过1 ，而且是排序树

每次插入时检查是否平衡，找到最小不平衡树，调整链接，然后旋转

多路查找树(B树) 数据库文件非常多，不能存内存，减少读取次数

一个结点存多个元素

2-3树 :分为2结点（二或无子）、3 结点（2元素，三或无子）左小右大（大小看**父情况**，动态给子分配2结点或3结点） 叶子同层

2-3-4 4 （3元素 四或无）

B 2-3 2-3-4 是B树的阶 3阶 4阶 1000阶 这样可以降深度 B遍历需要往返

查找 调整阶数 与磁盘页面大小匹配

B+ 中序后继者重复出现数据

伸展树

二叉查找树 遍历

赫夫曼树 只有叶结点带权 权是数字 ，常表示频率等，

路径，节点到节点的分支数 树的路径长:树根到每结点路径和

带权路径，路径长乘权 树的带权路径，带权路径和

树带权路径最小就是赫

如何合成: 先排序 最小的两结点连父成新树 父取代子进行排序 最小两节点连父成新树……

编码频率（权） 构成赫树 左分支权为0右为1 按照经过的路径重编码 （长短不等，必须不存在易混的前缀码）

解码也用赫树

五、散列

避免顺序查找，直接由key得到内存位置（或者索引） 位置=f(key)

然后把数据存到相应位置（需要估计数据key长度）

散列 hash 字典 关联数组 对象 变量名可以这样分配内存吗？(感觉没必要)

查找:f(key)读出地址，到地址确认key是否正确?

适合找**相等**记录

冲突:关键字相同，地址相同?

Hash 不支持排序 关键字key 关键字被散列函数映射到地址值

冲突的处理原则 均分，每次处理要求是同样的key，同样的结果

直接定址：线性处理

抽取: 取关键字一部分来算（适合比较大的数据）

平方取中间几位（不知分布，位数不大）

折叠:等量分割，叠加求和，取后几位（长关键字，不知分布）

除留余数: key mod p（小于） 可以折叠平方取中再取余

随机数 random(key) 需要伪随机种子

分离链接法 链表 二叉查找树 另一个散列表 装填因子

有冲突不换地方 用指针存所有同义的key,顺序连接，把首key散列

公共区溢出:找个公共的地方，放溢出表（用来顺序查找）

开放定址: 线性探测 冲突了就找下一个空地址 会导致不断偏移

平方探测 :二次探测， 位移用平方

随机探测: 随机数（伪随机，种子相同）

双散列 再散列 可扩散列 : 冲突的时候换一个散列

关键字是整数，是串

六、优先队列（堆） heap 入队出队

插入 删除最小元素 贪婪算法

简单链表实现（遍历最小比较难） 或者每次插入都对链表排序（插入太复杂）

用二叉查找树实现（比较复杂）

二叉堆 完全填满的二叉树（除了最底层，完全二叉树），可以用数组表示完全二叉树

1234567

堆需要数组，需要最大堆大小，当前堆大小

堆序性 子树也是堆 任意节点小于后面 任意节点大于父 最小在根部

插入 如果大于父，则直接插入 不然向上和父交换，直到成功位置:上滤

删除最小：最小在顶上， 删除后，保留空位 把较小的孩子换上来，不断循环到最底层，下滤

降低关键字的值（降低优先级），需要上滤 增加关键字的值

删除:降低再删除最小

创建堆:连续插入

二项树

D堆 二叉堆的增加 ，有d个孩子，二叉堆属于2堆 d堆也可以用数组（完全d叉树），不能用2进制移位会比较慢 （除非是2的幂比如4堆）

堆的合并merge 合并需要把一个数组拷贝到另一个数组，还要重排序？

左式堆 0路径长

斜堆

二项队列 不是树 而是多个堆序树 组成森林 每棵树都是二项树,高度

七、排序 关键字之间的大小 当关键字为字符串

趟

经常发生元素交换

冒泡: 从头 两两比较，大则交换， 把最大值换到最后

简单选择 比较 交换

插入排序

希尔排序

堆排序（完全二叉树）

归并排序

快速排序

决策树

桶式排序

八、不相交集ADT

九、图论算法

顶点非空 边（集）可空 无向图 顶点集V 边集E 有向图 有向边

无向完全图 任意两点都有边 顶点不到自身，边不重复，简单图

有向完全图 稀疏图 稠密图 权 网

顶点的度 入度 出度 邻接点

路径 是顶点序列，路径不唯一，路径长度，边的数

连通图，任两点连通

子图极大连通 要是连通图，极大顶点数 连通分量

生成树

图没位置

构造图（V E） 销毁图 返回点位置 取值 修改赋值 返回邻接 增加新点 删除点 增加边 深度优先遍历

邻接距阵 数组存顶点 二维存边 v1 v2 v3 v4 点与点是否有边 0，1 对于有向图，v1,v2有方向，例如:v1v2为1，v2v1为0

带权，将1换成权值

邻接表 顶点数组 所有的邻点构成链 next 出度 逆临接表

十字链表:方便有向图 顶点数组

邻接多重表 无向图 方便删改边

边集数组 顶点数组+ 边数组(begin（顶点下标）,end，权)

深搜DFS 访问一个点，再访问没访问的邻接点 不断向下

广搜BFS 类似层序 访问数据，再访问各邻接点

生成树:全部顶点和最少的边(n-1)

最小生成树 权值最小的生成树

克鲁斯卡尔 直接用边 边集数组 需要判断是否生成环路

普里姆：顶点起点，逐步找最小权值边

最短路径 顶点之间权和最小路径 直线不最短

迪杰斯特拉算法 一步步求顶点最短距离，比较该点和所有相连点，取最短;然后下一个点开始

（要求路径是通的）

弗洛伊德算法（所有顶点到所有顶点）

无权最短

负边值

无圈图（没有回路）

网络流

拓扑排序 AOV网 顶点：活动 有向

Vi到vj所有路径中vi都在vj之前，就是拓扑序列

有平行,没有回路

拓扑排序:就是在网中找拓扑序列 AOV**要求输出所有点?**

AOE网，带权AOV 有一个起点和终点 :源点,汇点，顶点是事件（完成），弧是活动（正进行），权是时间

平行进行的活动 洗碗，煮饭

关键路径 源点到汇点最大长度的路径 关键路径上的活动是关键活动

邻接表

NP完全性

十、算法设计技巧

贪婪 Huffman 近似装箱问题，联机算法 下项合适算法 最佳合适算法

分治 分解问题成相似子问题，递归解决 最近点问题 选择问题

矩阵相乘

动态规划 分治 先求解子问题，记录解到表

最优二叉查找树

图中所有点的最短路径

随机化算法

随机化 随机发生器 种子seed 线性同余发生器

跳跃表

素性测试（费马定理）

回溯 穷举，删除大量步骤 递归?

AI博奕 极大极小策略 a-b裁剪 博弈树（实际上没有这种树）

十一、摊还分析

二项队列

斐波堆

时间界

十二、高级数据结构及实现

自顶向下伸展树

红黑树

底向上插入

顶向下

顶向下删除

查找

根据值，在表中找到数据 （找位置，找一行）

查找表（可能是任何数据结构的，数组等）

关键字key 如果关键字表中唯一，主关键字 primary key 其他的是次关键字

找到了就是成功

静态查找表: 只查找 是否存在，查找相关行，关联数据 动态查找表:边查找边修改

顺序查找 线性查找 从头找到尾，逐个比较 foreach while a[i]=key

有序表查找:前提是数据排了序，可以进行有意识排序（对于有的不方便） 折半查找（二分查找），在中间找

插值查找 每次都把找到的值和key比较 如果表分布均匀，还不错

斐波那 查找 黄金分割 裴数组

无序数据:

线性索引查找 关键字和记录关联

稠密索引 每个记录对应一项索引（关键字有序） 可以用二分，插值，找索引

分块索引 块内无序 块间有序 块全部大于第一块

分块索引表 项:最大关键码 块中项数 块首指针 有序找到块 再顺序找

倒排索引（搜索引擎） 关键字+对应的内容序号

二叉排序树 如果是有序的表，不方便插入删除

使用二叉树 ，小的在左边，大的在右边，需要递归子树

中序遍历时就形成有序数组 查找 插入 删除（）

动态存储管理