刷题

目录

[一、数据结构 2](#_Toc490236255)

[1.链表 2](#_Toc490236256)

[2.栈 3](#_Toc490236257)

[3.队列 4](#_Toc490236258)

[4.数组 5](#_Toc490236259)

[5.字符串 5](#_Toc490236260)

[6.树 6](#_Toc490236261)

[7.堆 7](#_Toc490236262)

[8.散列（哈希表） 7](#_Toc490236263)

[二、算法 8](#_Toc490236264)

[1.插入排序 8](#_Toc490236265)

[2.希尔排序 8](#_Toc490236266)

[3.堆排序 9](#_Toc490236267)

[4.归并排序 9](#_Toc490236268)

[5.快速排序 9](#_Toc490236269)

[三、前端 9](#_Toc490236270)

[1.浏览器内核 9](#_Toc490236271)

[2.Html5标准 10](#_Toc490236272)

[3.BootStrap 11](#_Toc490236273)

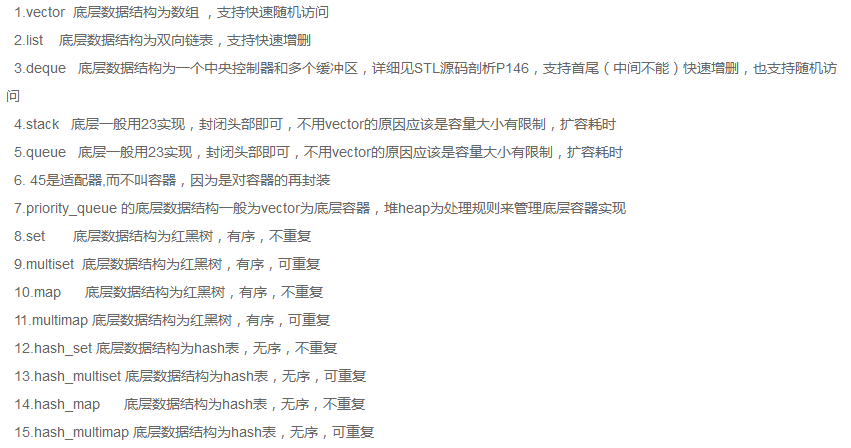
[4.前端基础 11](#_Toc490236274)

[四、计算机系统 13](#_Toc490236275)

[五、计算机网络 13](#_Toc490236276)

一、数据结构

1.链表

▲

▲深度用栈，广度遍历用队列

▲在链队列中,即使不设置尾指针也能进行入队操作

▲稀疏矩阵压缩的存储方法是: 三元组、十字链表

三元组：稀疏矩阵压缩方式，将矩阵中非0元素的行列号以及值构成一个三元组

（行号，列号，非零值）

稀疏矩阵转置，三元组行列号互换，并且按照原列号重新排序

十字链表：除了有行列号、值以外还有指向同一行和列下一个非零结点的指针域

▲ 无向图存储：邻接矩阵、邻接表、多重邻接表

有向图存储：邻接矩阵、邻接表、十字链表

▲对矩阵压缩存储是为了减少存储空间

▲静态查找表的查找方法主要有：有序表查找及  有序表、静态树表、索引顺序表等查找方法

▲在（非空）广义表中：

1、表头head可以是原子或者一个表

2、表尾tail一定是一个表

▲

▲C语言标识符规则：

（1）只能由字母、数字、下划线组成

（2）首字符必须是字母或下划线

（3）有长度限制，只识别前一定长度的字符

（4）区分大小写

▲数据流图中带有箭头的线段表示的是数据流

▲n圆盘的Hanoi塔移动次数f(n) = 2^n – 1

2.栈

▲以下选项中应用到栈：

递归

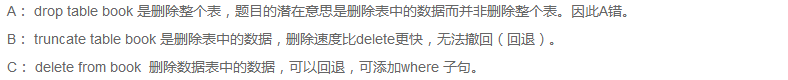
快速排序（非递归程序用栈实现）

表达式求值

树的遍历

▲通常使用栈来处理函数或过程的调用

▲堆栈溢出一般是由什么原因导致的？循环的递归调用，大数据结构的局部变量

▲

▲关于堆和栈的区别：

栈的大小是固定的，堆的大小受限于系统中有效的虚拟内存

栈的空间由系统决定何时释放，堆需要自己决定何时去释放

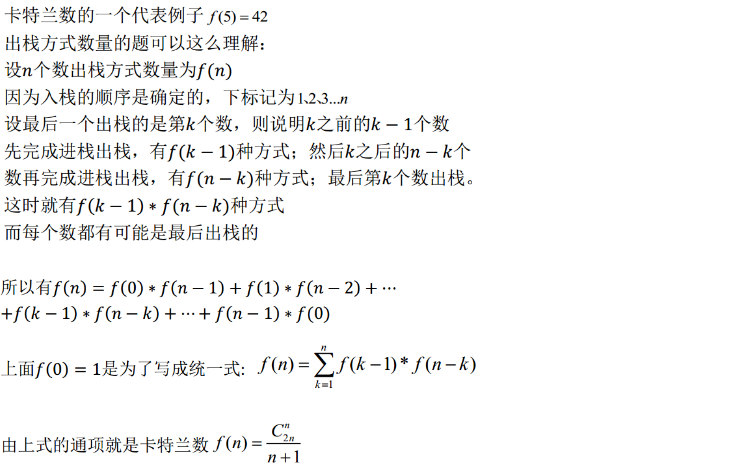
堆的使用容易产生碎片，但是用起来最方便

▲将一个递归算法改为对应的非递归算法时，通常需要使用栈

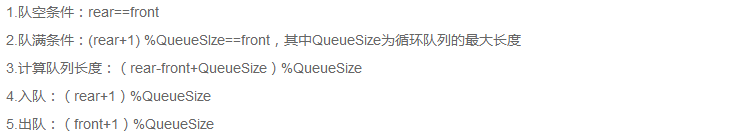
▲尾递归可以转换为循环的形式消除栈

▲DFS是图的深度优先遍历算法，用栈

▲n个元素出栈顺序



3.队列

▲

▲开源软件中经常被用作队列的是：Redis

▲有n个元素的序列，若使用筛选法建堆，则从位置为n/2取下整的元素开始建堆

4.数组

▲数组特殊矩阵压缩存储

（1）对称矩阵

在n阶对称矩阵中，Aij=Aji

我们只用为每一对对称元素分配一个存储空间

所以可以将n2个元素压缩存储到n(n+1)/2个元的空间中

一维数组A[n(n+1)/2]作为n阶对称矩阵的压缩存储结构

a[k]与矩阵元素之间存在着对应的关系：当i>=j时，k=i\*(i+1)/2+j;当i<j时，k=j\*(j+1)+i;

（2）上下三角矩阵

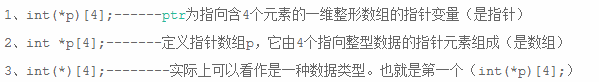
上、下三件矩阵是指矩阵上（下）三角（不包括对角线）中的元素均为常数c或0的n阶矩阵。则除了和对称矩阵一样，只储存其下（上）三角中的元素之外，再加一个储存常数c的储存空间

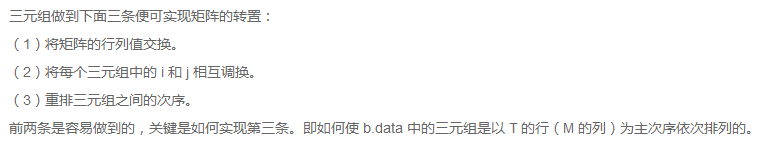
（3）对角矩阵

对角矩阵是指所有的非0元素都集中在以主对角线为中心的袋装区域中

即除了主对角线上和直接在对角线上、下方若干条对角线上的元素之外，所有其他的元素皆为0

假设对角矩阵的半宽为d,则对于n阶矩阵需要n\*(2\*d+1)-d\*d-d+1个储存空间

▲

▲

5.字符串

▲KMP算法



对应的部分匹配值指最后一位能匹配值所对应的部分匹配值！

▲字符串最后以 ’ \0 ’ 结束。strcpy将'1234\0'复制到目的地址

strlen()遇到 ’ \0 ’ 停止

6.树

▲n个结点的完全二叉树深度为[logn]+1

▲深度为K的满二叉树叶子结点数为2^(K-1)，前K-1层结点称为分支节点

▲最小生成树可以有多个， 他们的权值相等且最小即可

▲二叉树与树是不同的，二叉树不等价于分支树最多为二的有序树。当一个结点只包含一个子节点时，对于有序树并无左右孩子之分，而对于二叉树来说依然有左右孩子之分，所以二叉树与树是两种不同的结构

▲二叉树的度是当前节点下方边的总和，可能为：0、1、2

n个结点的二叉树，有n-1条边，度总和为n-1

假设1度结点为x个，2度为y，0度为z

x+2y=n-1,x+y+z=n

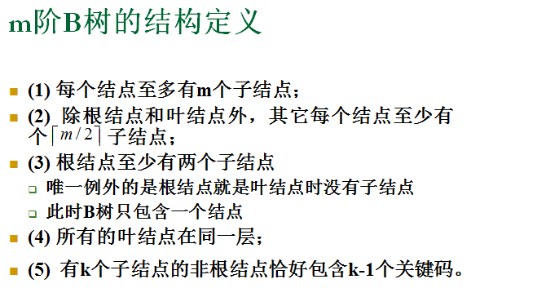
推出y=z-1

▲线索二叉树结点，左子树为空时，左指针指向后序遍历前驱

右子树为空时，右指针指向后序遍历后继

▲哈夫曼树，左右孩子权值之和为父结点权值，孩子权值只能小于父结点权值

▲n阶B-树，每个树根结点中所含的关键字数目最多允许为n-1个，最少允许为n/2-1个（向上取整ceil）



▲AVL二叉平衡树的插入，4种情况破坏平衡，单双旋转

7.堆

▲堆是完全二叉树，但不保证是平衡二叉树，因为堆中左右子树的高度差并不保证小于等于1

▲优化队列是逻辑结构，堆是数据结构，数组是存储方式

▲插入新元素、删除堆顶、大根堆、小根堆

▲堆化数组时间复杂度为O(n)，堆排序时间复杂度为O(n\*logn)

8.散列（哈希表）

▲

▲Hash 表的查找效率取决于散列函数、处理冲突的方法和装填因子。显然，冲突的产生概率与装填因子（表中记录数与表长之比）的大小成正比，即装填得越满越容易发生冲突

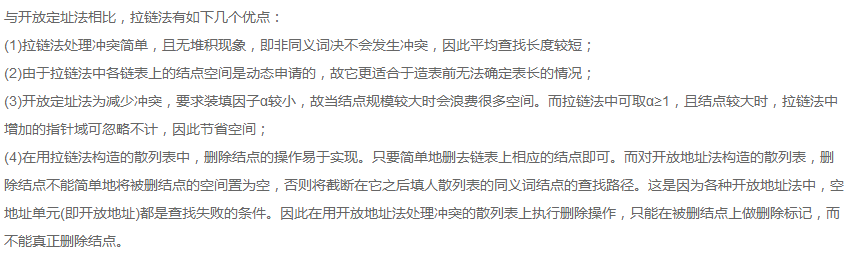
▲开哈希表-------链式地址法

闭哈希表-------开放地址法

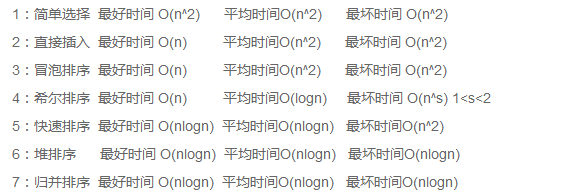
▲处理哈希冲突方法有：开放地址法（线性探测法、线性补偿探测法、随机探测），拉链法，公共溢出区，再散列法

▲既希望较快的查找又便于线性表动态变化的查找方法是索引顺序查找

▲有B+Tree/Hash\_Map/STL Map三种数据结构。对于内存中数据，查找性能较好的数据结构是（Hash\_Map），对于磁盘中数据，查找性能较好的数据结构是（B+Tree）



二、算法



1.插入排序

插入排序由N-1趟排序组成

对于P=1趟到P=N-1趟，插入排序保证从位置0到位置P上的元素为已排序状态

遍历元素，把每个元素左移到当前它应该出现的位置

2.希尔排序

它通过比较相距一定间隔的元素来工作

各趟比较所用的距离随着算法的进行而减少

直到比较相邻元素的最后一趟排序为止

（缩小增量排序）

增量序列：h1，h2…

3.堆排序

先堆化数组，再将堆顶A[0]与最后一个元素A[n-1]置换

再堆化数组，将堆顶A[0]与A[n-2]置换，重复，直到只剩下一个未排序元素

4.归并排序

将两个已排序的数组合并

但是它很难用于主存排序，主要问题在于需要线性附加内存，在整个算法中还要花费将数据拷贝到临时数组再拷贝回来的工作

5.快速排序

取一个枢纽元，将数组中大于和小于它的与元素分别进行快速排序，重复

6.递归

▲常见优化手段有：尾递归、迭代、循环

尾递归是指，在函数返回的时候，调用自身本身

并且，return语句不能包含表达式

这样，编译器或者解释器就可以把尾递归做优化，使递归本身无论调用多少次，都只占用一个栈帧，不会出现栈溢出的情况。 尾递归调用时，如果做了优化，栈不会增长，因此，无论多少次调用也不会导致栈溢出。 遗憾的是，大多数编程语言没有针对尾递归做优化

▲可以用非递归的方法实现二叉树前序遍历

▲程序调优的方法：

A缓存命中率提高，减少访问存储器的开销

B多线程并不能使IO操作并行化。可以通过异步读写、合并读写、计算代替读写等方式优化

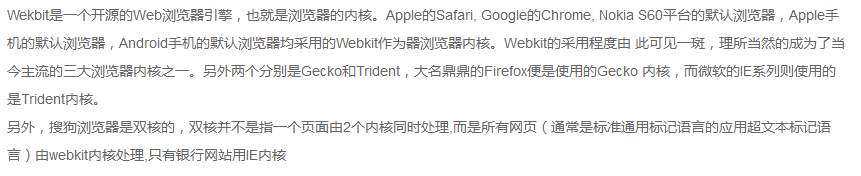
C可以减少连接和断开数据库的开销

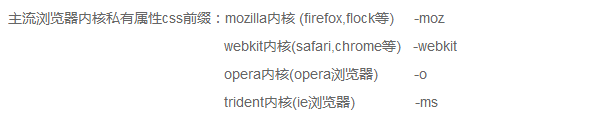
D可以减少运行时对程序栈进行操作的开销

E减少远程调用的次数

三、前端

1.浏览器内核

▲

▲▲

2.Html5标准

▲HTML5 新增的表单元素：datalist、output

▲



3.BootStrap

4.前端基础

▲

1.只要是img src引用的都会请求；

2. background: url()所属样式如果挂靠在某个元素上，就一定会请求；单独写了这么一个样式并不请求。

▲

▲

在html中，帧元素（frameset）的优先级最高，表单元素比非表单元素的优先级要高。

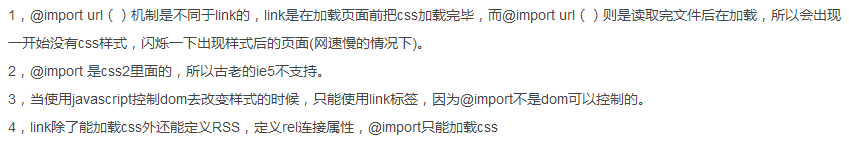
表单元素包括：文本输入框，密码输入框，单选框，复选框，文本输入域，列表框等等；

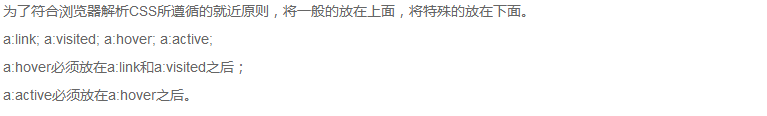
非表单元素包括：连接（a），div,table,span等。

所有的html元素又可以根据其显示分成两类：有窗口元素以及无窗口元素。有窗口元素总是显示在无窗口元素的前面。

有窗口元素包括：select元素，object元素，以及frames元素等等。

无窗口元素：大部分html元素都是无窗口元素。

▲

▲

▲常见的浏览器端的存储技术：cookie、localStorage、userData

▲

1.Ajax的优势：1.可搜索性 2.开放性 3.费用 4.易用性 5.易于开发。

2.Flash的优势：1.多媒体处理 2.兼容性 3.矢量图形 4.客户端资源调度

3.Ajax的劣势：1.它可能破坏浏览器的后退功能   2.使用动态页面更新使得用户难于将某个特定的状态保存到收藏夹中 ，不过这些都有相关方法解决。

4.Flash的劣势：1.二进制格式 2.格式私有 3.flash 文件经常会很大，用户第一次使用的时候需要忍耐较长的等待时间  4.性能问题

四、计算机系统

▲ 进程时间片用完可以降低其优先级，完成I／O的进程应该提升其优先级，处于就绪队列等待调度的进程一般不会改变其优先级

五、计算机网络

▲

局域网的网络地址192.168.1.0/24，局域网络连接其他网络的网关地址是192.168.1.1。主机192.168.1.20访问172.16.1.0/24网络时，其路由设置是：

route add -net 172.16.1.0 gw 192.168.1.1 netmask 255.255.255.0 metric 1

route add -net [目的子网] gw [网关地址] netmask [子网掩码] metric [路由跳数]

▲ping属于ICMP协议

六、数据库

▲Mysql复制分成三步：   
(1)    master将改变记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；   
(2)    slave将master的binary log events拷贝到它的中继日志(relay log)；

(3)    slave重做中继日志中的事件，将改变反映它自己的数据。

mysql支持master-slave复制，也支持master-master复制

一般情况下，异步复制的性能比半同步复制好，但后者相对更为安全

在有多个slave参与的半同步复制中，master并不一定需要等待全部slave返回

▲

BEGIN或START TRANSACTION；显示地开启一个事务；

COMMIT；也可以使用COMMIT WORK，不过二者是等价的。COMMIT会提交事务，并使已对数据库进行的所有修改称为永久性的；

ROLLBACK；有可以使用ROLLBACK WORK，不过二者是等价的。回滚会结束用户的事务，并撤销正在进行的所有未提交的修改；

SAVEPOINT identifier；SAVEPOINT允许在事务中创建一个保存点，一个事务中可以有多个SAVEPOINT；

RELEASE SAVEPOINT identifier；删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常；

ROLLBACK TO identifier；把事务回滚到标记点；

SET TRANSACTION；用来设置事务的隔离级别。InnoDB存储引擎提供事务的隔离级别有READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ和SERIALIZABLE。