## 树

1.完全二叉树的最后一个非叶子节点的序号是 (n/2)-1

如果节点序号是i，那么它的左子树序号是 2\*i+1，右子树的序号是2\*i+2

2.对于所有二叉树叶子节点个数=出度为2的节点个数+1

## 子串

字串的数目是n(n+1)/2 + 1

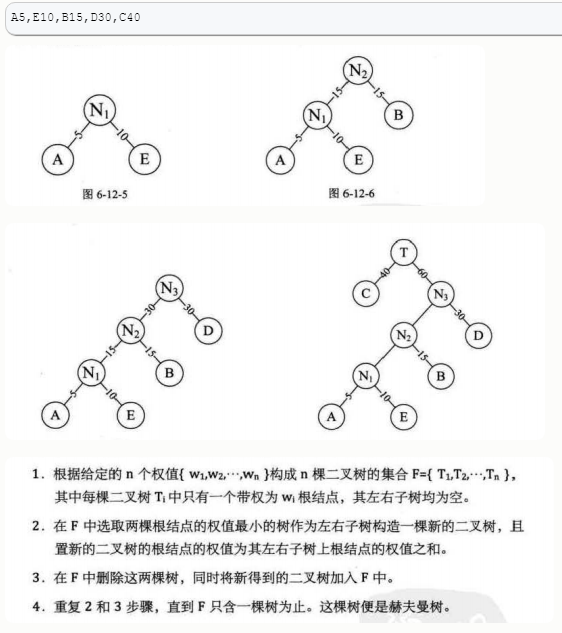
## 排序

一趟结束后能够确定一个元素的最终位置的排序方法有： 简单选择排序、快速排序、冒泡排序、堆排序

## 哈夫曼树

**要求一个字符的编码不能是另一个字符编码的前缀**

给定n个权值作为n个叶子结点，构造一棵二叉树，若该树的带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树(Huffman Tree)。哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。



## 计算机网络

**电路交换**有预留，且分配一定空间，提供专用的网络资源，提供有保证的服务，应用于**电话网**；

**分组交换**无预留，且不分配空间，存在网络资源争用，提供有无保证的服务。分组交换可用于**数据报网络和虚电路网络**。我们常用的**Internet就是数据报网络**，单位是Bit。

停止等待ARQ协议：

发送窗口Ws=1,接收窗口Wr＝1

连续ARQ协议

1、回退N帧ARQ协议：

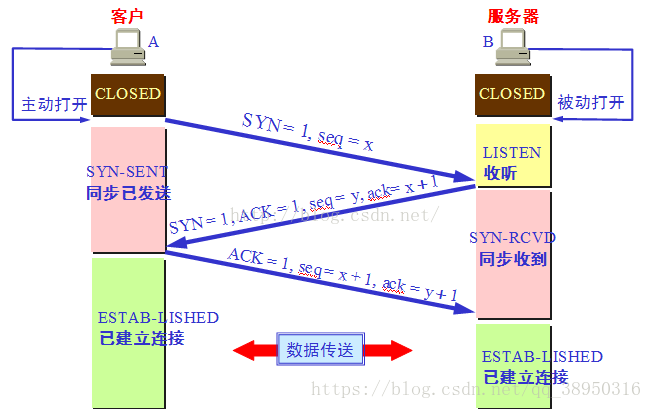
发送窗口Ws>1,接收窗口Wr=1

2、选择重发连续ARQ协议

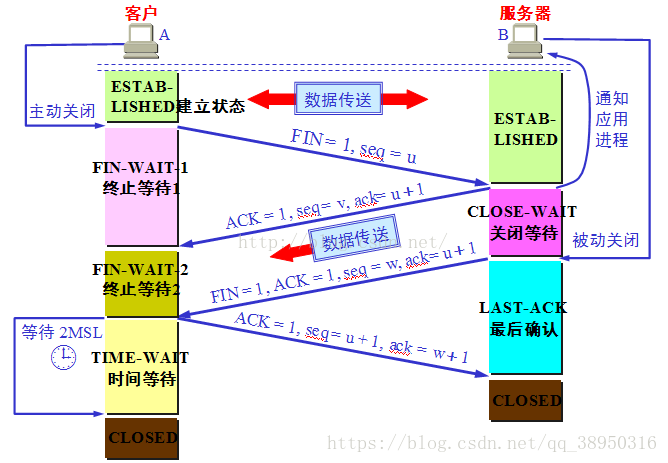
发送窗口Ws>1,接收窗口Wr>1,且Wr<=Ws

TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。

### 三次握手



### 四次挥手



### 集群主机

集群主机也叫集群空间，它不只是把空间开在一台服务器上，而是开到一个群组的所有服务器上，一般150台为一个群组。集群空间有数据同步和宕机检测与智能解析域名的功能。

一旦当前访问的服务器不能正常工作时，智能系统就会把客户的域名解析到能正常工作的服务器上。

### Netstat

Netstat是控制台命令,是一个监控TCP/IP网络的非常有用的工具，它可以显示路由表、实际的网络连接以及每一个网络接口设备的状态信息。Netstat用于显示与IP、TCP、UDP和ICMP协议相关的统计数据，一般用于检验本机各端口的网络连接情况。

### VPN的实现原理

<https://jingyan.baidu.com/article/02027811886aee1bcd9ce54f.html>

### 数据库

1 、第一范式（1NF）  
指数据库表的每一列(即每个属性)都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。**简而言之，第一范式就是无重复的列。**2、 第二范式（2NF）  
第二范式（2NF）**要求数据库表中的每个实例或行必须可以被唯一地区分**。

第二范式（2NF）要求实体的属性完全依赖于主关键字。

所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。

**简而言之，第二范式就是非主属性依赖于主关键字。**

3 、第三范式（3NF）

在满足第二范式的基础上，切不存在传递函数依赖，那么就是第三范式。

简而言之，**第三范式就是属性不依赖于其它非主属性。**

### 内连接

关键字：inner join … on…

组合两个表中的记录，返回关联字段相符的记录，也就是返回两个表的交集（阴影）部分。

### 左连接（左外连接）

关键字：left join … on …/ left outer join … on …

left join 是left outer join的简写，它的全称是左外连接，是外连接中的一种。

左(外)连接，左表(a\_table)的记录将会全部表示出来，而右表(b\_table)只会显示符合搜索条件的记录。右表记录不足的地方均为NULL。

### 右连接（右外连接）

关键字：right join … on … / right outer join … on …

right join是right outer join的简写，它的全称是右外连接，是外连接中的一种。

与左(外)连接相反，右(外)连接，左表(a\_table)只会显示符合搜索条件的记录，而右表(b\_table)的记录将会全部表示出来。左表记录不足的地方均为NULL。

### 全连接（全外连接）

关键字：outer join … on …

SELECT A.PK AS A\_PK,A.Value AS A\_Value,B.PK AS B\_PK,B.Value AS B\_Value

FROM table\_a A

LEFT JOIN table\_b B

ON A.PK = B.PK

UNION

SELECT A.PK AS A\_PK,A.Value AS A\_Value,B.PK AS B\_PK,B.Value AS B\_Value

FROM table\_a A

RIGHT JOIN table\_b B

ON A.PK = B.PK;

### Mysql 数据库引擎MyISAM与InnoDB

MyISAM：不支持事务，只支持表级锁，适合执行大量的select，insert

InnoDB：支持事务和外键、行级锁，适合执行大量的update和insert，并发

### ****索引的最左匹配特性****

（使用到索引）==（条件中出现联合索引最左前缀字段）

### truncate与delete

#### 相同点

效果上，delete from tableA，与truncate table tableA的作用是一样的（如果delete后面不添加where条件过滤），而且与drop table tableA，接着create table tableA的效果也是一样的。

#### 不同点

**1、语言上**

delete是DML，truncate是DDL，执行truncate需要drop权限。

**2、效率上**

delete是一行一行的删除，所以执行速度不快。由于truncate是DDL，是通过删除表然后重建表实现的，执行速度很快。

**3、事务上**

delete过程如果出现错误，事务是可以回滚的，而truncate无法回滚。

**4、效果上**

* delete可以删除符合条件的数据行，而truncate只能删除整体
* delete可以返回被删除的记录数，而truncate只返回0，没有任何意义
* 在存储引擎为InnoDB的数据库服务器中，如果一个表的主键作为了另一个表的外键，那么truncate无法删除这个表中的数据，无论这个外键是否存在，这是由于drop表限制的。而delete可以，只要这个表的主键在另一个表的外键中不存在即可。
* truncate删除一个表的数据后，这个表的auto\_increment列重新设置为初始大小，而delete则使用删除之前的顺序。索引同理。

### Java

解析：在同一类里面,首先静态代码块在类加载的时候就执行了,而且只执行一次,不论你是否new了对象,而构造代码块和构造方法是在每次创建对象时都会被调用，并且构造代码块的执行次序优先于类构造函数。

**ConstantValue**属性：

在java程序中，只有被static 和 final 共同修饰的 基本类型 + String（可以生存在常量池中的类型） 可以具备ConstantValue属性，拥有这个属性可以在编译期间对该变量进行赋值。不需要等到类加载器对类进行加载。

1、如果一个 static final 变量是"编译期常量",就像 public static final String a = "JD";那样，那么这个值不需要对 Test2 类进行初始化就可以读取。

2、但是,如果只是将一个变量的域设置为 static 和 final 的,那不足以确保这种行为。例如,对 public static final String a = new String("JD");的访问将强制对 Test2 类进行初始化,因为它不是一个"编译期常量"。

3、如果一个 static 变量域不是 final,那么在对它进行访问时，总是要求在它被读取之前,要先进行链接 (为这个域分配存储空间) 和初始化 (初始化该存储空间) 就像 public static String a = "JD";。