算法相关

数组排序（一维数组，二维数组）

快排

function quick\_sort($arr)  
{  
//判断参数是否是一个数组  
if(!is\_array($arr)) return false;  
//递归出口:数组长度为1，直接返回数组  
$length=count($arr);  
if($length<=1) return $arr;  
//数组元素有多个,则定义两个空数组  
$left=$right=array();  
//使用for循环进行遍历，把第一个元素当做比较的对象  
for($i=1;$i<$length;$i++)  
{  
//判断当前元素的大小  
if($arr[$i]<$arr[0]){  
$left[]=$arr[$i];  
}else{  
$right[]=$arr[$i];  
}  
}  
//递归调用  
$left=quick\_sort($left);  
$right=quick\_sort($right);  
//将所有的结果合并  
return array\_merge($left,array($arr[0]),$right);

二分查找

function getValue4($num,$arr,$start = 0,$end = 100){  
//采用二分法查找  
$middle = floor(($end + $start) / 2);  
  
//判断  
if($arr[$middle] == $num){  
//已经找到了，递归的出口  
return $middle + 1;  
}elseif($arr[$middle] < $num){  
//要查找的元素在数组的后半段  
$start = $middle + 1;  
//边界值  
if($start >= $end){  
//没有找到，但是已经超出边界值，递归出口  
return false;  
}  
//调用自己去查找：递归点  
return getValue4($num,$arr,$start,$end); //getValue4($num,$arr,51,100)  
}else{  
//要查找的元素在数组的前半段  
$end = $middle - 1;  
//判断边界值  
if($end < 0)return false;  
  
//调用自己：递归点  
return getValue4($num,$arr,$start,$end); //getValue4($num,$arr,0,49)  
}  
  
//都没有找到  
return false;  
}

mysql相关

mysql存储引擎（b+树了解一下）  
1）MyISAM：  
每个MyISAM在磁盘上存储成三个文件。第一个文件的名字以表的名字开始，扩展名指出文件类型。.frm文件存储表定义。数据文件的扩展名为.MYD (MYData)。  
MyISAM表格可以被压缩，而且它们支持全文搜索。不支持事务，而且也不支持外键。如果事物回滚将造成不完全回滚，不具有原子性。在进行updata时进行表锁，并发量相对较小。如果执行大量的SELECT，MyISAM是更好的选择。  
MyISAM的索引和数据是分开的，并且索引是有压缩的，内存使用率就对应提高了不少。能加载更多索引，而Innodb是索引和数据是紧密捆绑的，没有使用压缩从而会造成Innodb比MyISAM体积庞大不小  
MyISAM缓存在内存的是索引，不是数据。而InnoDB缓存在内存的是数据，相对来说，服务器内存越大，InnoDB发挥的优势越大。

优点：查询数据相对较快，适合大量的select，可以全文索引。  
缺点：不支持事务，不支持外键，并发量较小，不适合大量update

2）InnoDB：（参数说明：[Mysql存储引擎之Innodb重要参数说明](http://www.cnblogs.com/kevingrace/p/6133818.html" \t "_blank)）  
这种类型是事务安全的。.它与BDB类型具有相同的特性,它们还支持外键。InnoDB表格速度很快。具有比BDB还丰富的特性,因此如果需要一个事务安全的存储引擎，建议使用它。在update时表进行行锁，并发量相对较大。如果你的数据执行大量的INSERT或UPDATE，出于性能方面的考虑，应该使用InnoDB表。  
优点：支持事务，支持外键，并发量较大，适合大量update  
缺点：查询数据相对较快，不适合大量的select  
对于支持事物的InnoDB类型的表，影响速度的主要原因是AUTOCOMMIT默认设置是打开的，而且程序没有显式调用BEGIN 开始事务，导致每插入一条都自动Commit，严重影响了速度。可以在执行sql前调用begin，多条sql形成一个事物（即使autocommit打开也可以），将大大提高性能。

3）memory，表存在内存里面，可以用hash索引

Mysql数据事务隔离级别

一、事务的基本要素（ACID）

1、原子性（Atomicity）：事务开始后所有操作，要么全部做完，要么全部不做，不可能停滞在中间环节。事务执行过程中出错，会回滚到事务开始前的状态，所有的操作就像没有发生一样。也就是说事务是一个不可分割的整体，就像化学中学过的原子，是物质构成的基本单位。

2、一致性（Consistency）：事务开始前和结束后，数据库的完整性约束没有被破坏 。比如A向B转账，不可能A扣了钱，B却没收到。

3、隔离性（Isolation）：同一时间，只允许一个事务请求同一数据，不同的事务之间彼此没有任何干扰。比如A正在从一张银行卡中取钱，在A取钱的过程结束前，B不能向这张卡转账。

4、持久性（Durability）：事务完成后，事务对数据库的所有更新将被保存到数据库，不能回滚。

事务隔离级别

1）read uncommitted : 读取尚未提交的数据 ：哪个问题都不能解决

2）read committed：读取已经提交的数据 ：可以解决脏读 ---- oracle默认的

3）repeatable read：重读读取：可以解决脏读 和 不可重复读 ---mysql默认的

4）serializable：串行化：可以解决 脏读 不可重复读 和 幻读---相当于锁表

**1、脏读：事务A读取了事务B更新的数据，然后B回滚操作，那么A读取到的数据是脏数据**

**2、不可重复读：事务 A 多次读取同一数据，事务 B 在事务A多次读取的过程中，对数据作了更新并提交，导致事务A多次读取同一数据时，结果 不一致。**

**3、幻读：系统管理员A将数据库中所有学生的成绩从具体分数改为ABCDE等级，但是系统管理员B就在这个时候插入了一条具体分数的记录，当系统管理员A改结束后发现还有一条记录没有改过来，就好像发生了幻觉一样，这就叫幻读。**

**小结：不可重复读的和幻读很容易混淆，不可重复读侧重于修改，幻读侧重于新增或删除。解决不可重复读的问题只需锁住满足条件的行，解决幻读需要锁表**

Redis相关

数据类型：

字符串，链表，哈希，集合，有序集合

Redis单线程为什么快？

1、完全基于内存

Redis是纯内存数据库，相对于读写磁盘，读写内存的速度就不是几倍几十倍了，一般，hash查找可以达到每秒百万次的数量级。

2、多路复用IO

“多路”指的是多个网络连接，“复用”指的是复用同一个线程。采用多路 I/O 复用技术可以让单个线程高效的处理多个连接请求（尽量减少网络IO的时间消耗）。可以直接理解为：单线程的原子操作，避免上下文切换的时间和性能消耗；加上对内存中数据的处理速度，很自然的提高redis的吞吐量。

Redis怎么实现持久化

Redis主从同步

Php相关

Php内存回收机制

Php是怎么实现实时类型的判断

Php怎么启动

Php-fpm是做啥的

Nginx相关

介绍nginx的工作原理

Rabbitmq相关

Rabbitmq消息怎么实现持久化

Laravel相关

为什么用laravel

设计模式相关

了解两三个设计模式

单例模式

Linux相关

经常使用的命令（查看内存使用情况，ps，nohup）

进程间通信有哪些

Grep命令（不包含某某，查出来一个多少行）

Scp命令

网络通信相关

正向代理和反向代理了解

负载均衡的机制

网络协议

零碎知识点

empty(‘0’) 返回true