[问答题]

1. 如果线上某台虚机CPU Load过高，该如何快速排查原因？只介绍思路和涉及的Linux命令即可

参考答案：造成cpu load过高的原因： Full gc次数的增大、代码中存在Bug（例如死循环、正则的不恰当使用等）都有可能造成cpu load 增高。   
1. jps -v：查看java进程号   
2. top -Hp [java进程号]：查看当前进程下最耗费CPU的线程   
3. printf "%x\n" [步骤2中的线程号]：得到线程的16进制表示   
4. jstack [java进程号] | grep -A100 [步骤3的结果]：查看线程堆栈，定位代码行。参考：如何使用JStack分析线程状态

## 2. 请简要描述MySQL数据库联合索引的命中规则，可举例说明

参考答案：1) MySQL联合索引遵循最左前缀匹配规则，即从联合索引的最左列开始向右匹配，直到遇到匹配终止条件。例如联合索引(col1, col2, col3), where条件为col1=`a` AND col2=`b`可命中该联合索引的(col1,col2)前缀部分, where条件为col2=`b` AND col3=`c`不符合最左前缀匹配，不能命中该联合索引。

2) 匹配终止条件为范围操作符(如>, <, between, like等)或函数等不能应用索引的情况。例如联合索引(col1, col2, col3), where条件为col1=`a` AND col2>1 AND col3=`c`, 在col2列上为范围查询，匹配即终止，只会匹配到col1，不能匹配到(col1, col2, col3).

3) where条件中的顺序不影响索引命中。例如联合索引(col1, col2, col3), where条件为col3=`c` AND col2=b AND col1=`a`, MySQL优化器会自行进行优化，可命中联合索引(col1, col2, col3).

1. 什么是分布式事务，分布式事务产生的原因是什么？分布式事务的解决方案有哪些？分别有哪些优缺点？

参考答案：答案参考InfoQ的相关文章：

分布式系统事务一致性

[https://mp.weixin.qq.com/s?\_\_biz=MzI5ODQ2MzI3NQ==&mid=2247486113&idx=1&sn=bde545f4f5efacb6a68df3ae9a8f1ec1&chksm=eca433e5dbd3baf344af2dd70e46233dd64047a4dd6ac487ba353fb5546da3aa68c65580c3e4&scene=27#wechat\_redirect](https://www.nowcoder.com/test/question/done?tid=33548175&qid=894499#wechat_redirect)

5种分布式事务解决方案优缺点对比

[https://mp.weixin.qq.com/s?\_\_biz=MzI5ODQ2MzI3NQ==&mid=2247487531&idx=1&sn=b3fbc4dee7cea4a78db062a4a656afdf&chksm=eca4296fdbd3a079a8e328ec7946ced7d1f94c0f105463743a8bee569bae6da00bf2133c3e1a&scene=27#wechat\_redirect](https://www.nowcoder.com/test/question/done?tid=33548175&qid=894499#wechat_redirect)

1. 请描述https的请求过程。

参考答案：1) 客户端向服务器发起HTTPS请求，连接到服务器的443端口；

2) 服务器端有一个密钥对，即公钥（即数字证书）和私钥，是用来进行非对称加密使用的，服务器端保存着私钥，不能将其泄露，公钥可以发送给任何人；

3) 服务器将自己的公钥发送给客户端；

4) 客户端收到服务器端的公钥之后，检查其合法性，如果发现发现公钥有问题，那么HTTPS传输就无法继续，如果公钥合格，则客户端会生成一个客户端密钥，然后用服务器的公钥对客户端密钥进行非对称加密成密文，至此，HTTPS中的第一次HTTP请求结束；

5) 客户端发起HTTPS中的第二个HTTP请求，将加密之后的客户端密钥发送给服务器；

6) 服务器接收到客户端发来的密文之后，会用自己的私钥对其进行非对称解密，解密之后的明文就是客户端密钥，然后用客户端密钥对数据进行对称加密，这样数据就变成了密文；

7) 然后服务器将加密后的密文发送给客户端；

8) 客户端收到服务器发送来的密文，用客户端密钥对其进行对称解密，得到服务器发送的数据。这样HTTPS中的第二个HTTP请求结束，整个HTTPS传输完成。

5.什么是事务传播行为？你知道Spring事务中都有哪些传播类型吗？如何使用/指定传播类型？

参考答案：1） 事务传播用于描述当一个由事务传播行为修饰的方法被嵌套入另外一个方法时，事务如何传播。常用于定义发生事务嵌套时，如何继续执行。

2） Spring 中共定义了7中事务传播类型，明细如下表， 需答出3~4种常见类型即可：

    a) PROPAGATION\_REQUIRED： 当前没有事务时开启新事务，如果有则加入；

    b) PROPAGATION\_REQUIRES\_NEW： 强制开启新事务，挂起已有事务（如有）；

    c) PROPAGATION\_SUPPORTS： 当前有事务时加入， 没有则以非事务方式执行；

    d) PROPAGATION\_NOT\_SUPPORTED： 以非事务方式执行， 挂起当前事务（如有）；

3） 可以在注解或者XML中指定传播类型， 如 “@Transactional(Propagation=xxx)”

6.  IO设计中Reactor 和 Proactor 区别。

参考答案：1、 Reactor被动的等待指示事件的到来并作出反应，有一个等待的过程，做什么都要先放入到监听事件集合中等待handler可用时再进行操作，实现相对简单，对于耗时短的处理场景比较高效，但Reactor处理耗时长的操作会造成事件分发的阻塞，影响到后续事件的处理。

2、Proactor直接调用异步读写操作，调用完后立刻返回，实现了一个主动的事件分离和分发模型；这种设计允许多个任务并发的执行，从而提高吞吐量；并可执行耗时长的任务（各个任务间互不影响），Proactor性能更高，能够处理耗时长的并发场景，但Proactor实现逻辑复杂；依赖操作系统对异步的支持，目前实现了纯异步操作的操作系统少，实现优秀的如windows IOCP，但由于其windows系统用于服务器的局限性，目前应用范围较小；而Unix/Linux系统对纯异步的支持有限，应用事件驱动的主流还是通过select/epoll来实现。

7. [编程题]大数加法

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

以字符串的形式读入两个数字，再以字符串的形式输出两个数字的和。

##### 输入描述:

输入两行，表示两个数字a和b，-109 <= a , b <= 109 ，用双引号括起。

##### 输出描述:

输出a+b的值，用双引号括起。

##### 输入例子1:

"-26"

"100"

##### 输出例子1:

"74"

8. [编程题]回文子串

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

给定一个字符串，你的任务是计算这个字符串中有多少个回文子串（回文串是一个正读和反读都一样的字符串）。

具有不同开始位置或结束位置的回文串，即使是由相同的字符组成，也会被计为是不同的子串。

##### 输入描述:

输入仅包含一个字符串，长度不会超过 1000。

##### 输出描述:

输出仅包含一个非负整数， 代表输入字符串有多少个回文子串。

##### 输入例子1:

abc

##### 输出例子1:

3

##### 输入例子2:

aaa

##### 输出例子2:

6

9.[编程题]合并金币

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

 有 N 堆金币排成一排，第 i 堆中有 C[i] 块金币。每次合并都会将相邻的两堆金币合并为一堆，成本为这两堆金币块数之和。经过N-1次合并，最终将所有金币合并为一堆。请找出将金币合并为一堆的最低成本。

其中，1 <= N <= 30，1 <= C[i] <= 100

##### 输入描述:

第一行输入一个数字 N 表示有 N 堆金币

第二行输入 N 个数字表示每堆金币的数量 C[i]

##### 输出描述:

输出一个数字 S 表示最小的合并成一堆的成本

##### 输入例子1:

4

3 2 4 1

##### 输出例子1:

20

##### 输入例子2:

30

10 20 30 40 50 60 70 80 90 100 99 89 79 69 59 49 39 29 19 9 2 12 22 32 42 52 62 72 82 92

##### 输出例子2:

7307

10. [编程题]最小唯一前缀

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

给定一组个字符串，为每个字符串找出能够唯一识别该字符串的最小前缀。

##### 输入描述:

第一行输入一个整数 n 表示字符串个数

后面n行，每行一个字符串，一共n串互不相同的字符串。（2 <= n <= 100，字符串长度不超过100）

##### 输出描述:

输出n行，每行一个字符串，依次是每个字符串的最小可唯一识别前缀

##### 输入例子1:

5

meituanapp

meituanwaimai

dianpingliren

dianpingjiehun

mt

##### 输出例子1:

meituana

meituanw

dianpingl

dianpingj

mt

[问答题]

1.RESTful调用和 RPC调用有什么区别？如果让你设计一个RPC服务治理框架你会设计那些模块？是否了解过Service Mesh，如果了解Service Mesh是用来解决什么问题的？

参考答案：1、RESTful是一种软件架构风格，用于约束客户端和服务器交互，满足这些约束条件和原则的应用程序或设计就是 RESTful。比如HTTP协议使用同一个URL地址，通过GET，POST，PUT，DELETE等方式实现查询、提交、删除数据。RPC是远程过程调用，是用于解决分布式系统服务间调用的一种方式。RPC采用客户端与服务端模式，双方通过约定的接口（常见为通过IDL定义或者是代码定义）以类似本地方法调用的方式来进行交互，客户端根据约定传输调用函数+参数给服务端（一般是网络传输TCP/UDP），服务端处理完按照约定将返回值返回给客户端。

重点为RESTful HTTP的约束风格，RPC调用模型。

1、可分为两大部分RPC +服务治理  
RPC部分 = IDL  +客户端/服务端实现层  +协议层 +数据传输层  
服务治理 =服务管理（注册中心） +服务监控 +服务容灾 +服务鉴权

2、 Service Mesh为了解决传统微服务框架"胖客户端"方式，引入的如下问题：  
与业务无关的服务治理逻辑与业务代码强耦合，框架、SDK的升级与业务代码强绑定，多语言的胖客户端支持起来性价比极低。

2. 请描述避免多线程竞争时有哪些手段？

参考答案：1) 不可变对象；

2) 互斥锁；

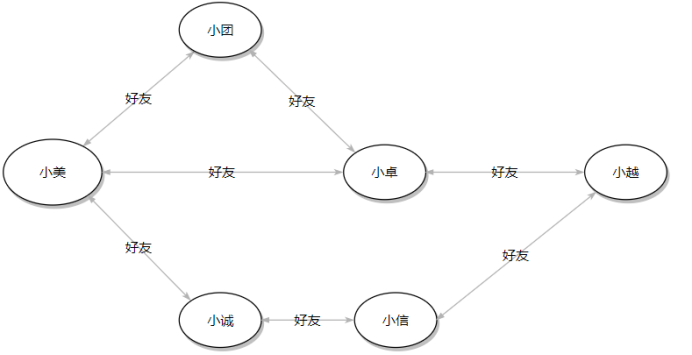
3)  ThreadLocal 对象；

4) CAS

## 3. 题目描述

1929年，匈牙利作家Frigyes Karinthy在短篇故事‘Chains’中首次提出的“六度人脉理论”，是指地球上所有的人都可以通过六层以内的熟人链和任何其他人联系起来。我们定义A的‘一度好友’为A直接相识的好友，A的‘二度好友’为A一度好友的好友且与A不是一度好友，A的‘三度好友’为A二度好友的好友且与A不是一度好友、二度好友，以此类推。

在美团点评，小美、小团、小卓、小越、小诚、小信的好友关系见下图。



小团、小卓、小诚是小美的一度好友。小越、小信是小美的二度好友。小诚、小越是小信的一度好友，小美、小卓是小信的二度好友，小团是小信的三度好友。

现在已知每个人的所有一度好友，需要为‘小点’推荐10个六度好友，请使用伪代码写出计算方法。

请简述HTTP的5个常用Method及其含义，以及5个常用Status Code及其含义？HTTP与HTTPS的区别是什么，简述一下HTTPS的实现原理。

[编程题]表达式求值

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

给出一个布尔表达式的字符串，比如：true or false and false，表达式只包含true，false，and和or，现在要对这个表达式进行布尔求值，计算结果为真时输出true、为假时输出false，不合法的表达时输出error（比如：true true）。表达式求值是注意and 的优先级比 or 要高，比如：true or false and false，等价于 true or (false and false)，计算结果是 true。

输入描述:

输入第一行包含布尔表达式字符串s，s只包含true、false、and、or几个单词（不会出现其它的任何单词），且单词之间用空格分隔。 (1 ≤ |s| ≤ 103).

输出描述:

输出true、false或error，true表示布尔表达式计算为真，false表示布尔表达式计算为假，error表示一个不合法的表达式。

输入例子1:

and

输出例子1:

error

输入例子2:

true and false

输出例子2:

False

输入例子3:

true or false and false

##### 输出例子3:

true

6.[编程题]字符串模式匹配

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

给出两个字符串，分别是模式串P和目标串T，判断模式串和目标串是否匹配，匹配输出 1，不匹配输出 0。模式串中‘？’可以匹配目标串中的任何字符，模式串中的 ’\*’可以匹配目标串中的任何长度的串，模式串的其它字符必须和目标串的字符匹配。例如P=a?b，T=acb，则P 和 T 匹配。

##### 输入描述:

输入第一行包含一个字符串p， (1 ≤ |p| ≤ 20).

输入第二行包含一个字符串t， (1 ≤ |t| ≤ 20).

##### 输出描述:

输出仅包含0和1的整数，0表示p和t不匹配，1表示p和t匹配。

##### 输入例子1:

a?b

ab

##### 输出例子1:

0

##### 输入例子2:

a\*b

ab

##### 输出例子2:

1

##### 输入例子3:

a\*b

a(cb

##### 输出例子3:

1

7.[编程题]订单分配

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

打车派单场景, 假定有N个订单， 待分配给N个司机。每个订单在匹配司机前，会对候选司机进行打分，打分的结果保存在N\*N的矩阵A， 其中Aij 代表订单i司机j匹配的分值。

假定每个订单只能派给一位司机，司机只能分配到一个订单。求最终的派单结果，使得匹配的订单和司机的分值累加起来最大，并且所有订单得到分配。

##### 输入描述:

第一行包含一个整数N，2≤N≤10。

第二行至第N+1行包含N\*N的矩阵。

##### 输出描述:

输出分值累加结果和匹配列表，结果四舍五入保留小数点后两位

（注意如果有多组派单方式得到的结果相同，则有限为编号小的司机分配编号小的订单，比如：司机1得到1号单，司机2得到2号单，就比司机1得到2号单，司机2得到1号单要好）

##### 输入例子1:

3

1.08 1.25 1.5

1.5 1.35 1.75

1.22 1.48 2.5

##### 输出例子1:

5.25

1 2

2 1

3 3

##### 例子说明1:

第一行代表得到的最大分值累加结果5.25，四舍五入保留两位小数；

第二行至第四行代表匹配的结果[i j],其中i按行递增：

订单1被派给司机2，订单2被派给司机1，订单3被派给司机3。使得A12+ A21+ A33= 1.25 + 1.5 + 2.5 = 5.25在所有的组合中最大。

8.[编程题]美团骑手包裹区间分组

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

2110年美团外卖火星第3000号配送站点有26名骑手，分别以大写字母A-Z命名，因此可以称呼这些骑手为黄家骑士特工A，黄家骑士特工B…黄家骑士特工Z，某美团黑珍珠餐厅的外卖流水线上会顺序产出一组包裹，美团配送调度引擎已经将包裹分配到骑手，并在包裹上粘贴好骑手名称，如RETTEBTAE代表一组流水线包裹共9个，同时分配给了名字为A B E R T的5名骑手。请在不打乱流水线产出顺序的情况下，把这组包裹划分为尽可能多的片段，同一个骑手只会出现在其中的一个片段，返回一个表示每个包裹片段的长度的列表。

##### 输入描述:

输入数据只有一行，为一个字符串(不包含引号)，长度不超过1000，只包含大写字母'A'到'Z'，字符之间无空格。

##### 输出描述:

输出每个分割成片段的包裹组的长度，每个长度之间通过空格隔开

##### 输入例子1:

MPMPCPMCMDEFEGDEHINHKLIN

##### 输出例子1:

9 7 8

##### 例子说明1:

划分结果为MPMPCPMCM,DEFEGDE,HINHKLIN。

每个骑手最多出现在一个片段中。

像MPMPCPMCMDEFEGDE,HINHKLIN的划分是错误的，因为划分的片段数较少。

9.[编程题]火星文字典

时间限制：C/C++ 1秒，其他语言2秒

空间限制：C/C++ 256M，其他语言512M

已知一种新的火星文的单词由英文字母（仅小写字母）组成，但是此火星文中的字母先后顺序未知。给出一组非空的火星文单词，且此组单词已经按火星文字典序进行好了排序（从小到大），请推断出此火星文中的字母先后顺序。

##### 输入描述:

一行文本，为一组按火星文字典序排序好的单词(单词两端无引号)，单词之间通过空格隔开

##### 输出描述:

按火星文字母顺序输出出现过的字母，字母之间无其他字符，如果无法确定顺序或者无合理的字母排序可能，请输出"invalid"(无需引号)

##### 输入例子1:

z x

##### 输出例子1:

zx

##### 输入例子2:

wrt wrf er ett rftt

##### 输出例子2:

wertf

##### 输入例子3:

z x z

##### 输出例子3:

invalid