前端+测试+运维+算法综合宝典 2020 版

前端部分面试题

1. 判断第二个日期比第一个日期大

如何用脚本判断用户输入的的字符串是下面的时间格式 2004-11-21 必须要保证用户的输入是此格式，并且是时间，比如说月份不大于 12 等等，另外我需要用户输入两个，并且后一个要比前一个晚，只允许用 JAVASCRIPT，请详细帮助作答，，

```
//这里可用正则表达式判断提前判断一下格式，然后按下提取各时间字段内容
    <script type="text/javascript">
      window.onload = function()
      {
         //这么写是为了实现 js 代码与 html 代码的分离，当我修改 js 时，不能影响 html 代码。
         document.getElementById("frm1").onsubmit =
              function(){
                   var d1 = this.d1.value;
                   var d2 = this.d2.value;
                   if(!verifyDate (d1)) {alert("第一个日期格式不对");return false;}
                   if(!verifyDate (d2)) {alert("第二个日期格式不对");return false;}
                   if(!compareDate(d1,d2)) {alert("第二个日期比第一日期小");return false;}

              };
         }

         function compareDate(d1,d2)
         {
              var arrayD1 = d1.split("-");
              var date1 = new Date(arrayD1[0],arrayD1[1],arrayD1[2]);
              var arrayD2 = d2.split("-");
              var date2 = new Date(arrayD2[0],arrayD2[1],arrayD2[2]);
              if(date1 > date2) return false;
              return true;
         }

         function verifyDate(d)
         {
              var datePattern = /^\d{4}-(0?[1-9]|1[0-2])-(0?[1-9]|[1-2]\d|3[0-1])$/;
```

```
            return datePattern.test(d);
        }
    </script>

    <form id="frm1" action="xxx.html">
    <input type="text" name="d1" />
    <input type="text" name="d2" />
    <input type="submit"/>
    </form>
```

2. 用 table 显示 n 条记录，每 3 行换一次颜色，即 1，2，3 用红色字体，4，5，6 用绿色字体，7，8，9 用红颜色字体。

```
    <body>
    <table id="tbl">
      <tr><td>1</td></tr>
      <tr><td>2</td></tr>
      <tr><td>3</td></tr>
      <tr><td>4</td></tr>
      <tr><td>5</td></tr>
      <tr><td>6</td></tr>
      <tr><td>7</td></tr>
      <tr><td>8</td></tr>
      <tr><td>9</td></tr>
      <tr><td>10</td></tr>
    </table>
    </body>
    <script type="text/javascript">
      window.onload=function()
        {
            var tbl = document.getElementById("tbl");
            rows = tbl.getElementsByTagName("tr");
            for(i=0;i<rows.length;i++)
            {
                var j = parseInt(i/3);
                if(j%2==0) rows[i].style.backgroundColor="#f00";
                else    rows[i].style.backgroundColor="#0f0";
            }
        }
    </script>
```

3、HTML 的 form 提交之前如何验证数值文本框的内容全部为数字？否则的话提示用户并终止提交?

```
<form onsubmit=’return chkForm(this)’>
<input type="text" name="d1"/>
<input type="submit"/>
</form>
<script type=”text/javascript” />
function chkForm(this)
    {
        var value = thist.d1.value;
        var len = value.length;
        for(var i=0;i<len;i++)
        {
            if(value.charAt(i)>"9" || value.charAt(i)<"0")
            {
                alert("含有非数字字符");
                return false;
            }
        }
        return true;
    }
</script>
```

4、请写出用于校验 HTML 文本框中输入的内容全部为数字的 javascript 代码

```
    <input type="text" id="d1" onblur=" chkNumber (this)"/>
<script type=”text/javascript” />
function chkNumber(eleText)

    {
        var value = eleText.value;
        var len = value.length;
        for(var i=0;i<len;i++)
        {
            if(value.charAt(i)>"9" || value.charAt(i)<"0")
            {
                alert("含有非数字字符");

                eleText.focus();
                break;
            }
```

```
        }
    }
</script>
```

除了写完代码，还应该在网页上写出实验步骤和在代码中加入实现思路，让面试官一看就明白你的意图和检查你的结果。

5、px 和 em 的区别

相同点：px 和 em 都是长度单位；
异同点：px 的值是固定的，指定是多少就是多少，计算比较容易。em 得值不是固定的，并且 em 会继承父级元素的字体大小。
浏览器的默认字体高都是 16px。所以未经调整的浏览器都符合: 1em=16px。那么 12px=0.75em, 10px=0.625em。

6、$(document).ready()方法和 window.onload 有什么区别？

(1)、window.onload 方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到浏览器后才执行的。
(2)、$(document).ready() 方法可以在 DOM 载入就绪时就对其进行操纵，并调用执行绑定的函数。

7、jquery 中$.get()提交和$.post()提交有区别吗？

相同点：都是异步请求的方式来获取服务端的数据；
异同点：
1、请求方式不同：$.get() 方法使用 GET 方法来进行异步请求的。$.post() 方法使用 POST 方法来进行异步请求的。
2、参数传递方式不同：get 请求会将参数跟在 URL 后进行传递，而 POST 请求则是作为 HTTP 消息的实体内容发送给 Web 服务器的，这种传递是对用户不可见的。
3、数据传输大小不同：get 方式传输的数据大小不能超过 2KB 而 POST 要大的多
4、安全问题： GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

8、谈谈你对 webpack 的看法

WebPack 是一个模块打包工具，你可以使用 WebPack 管理你的模块依赖，并编绎输出模块们所需的静态文件。它能够很好地管理、打包 Web 开发中所用到的 HTML、**JavaScript**、CSS 以及各种静态文件（图片、字体等），让开发过程更加高效。对于不同类型的资源，webpack 有对应的模块加载器。webpack 模块打包器会分析模块间的依赖关系，最后 生成了优化且合并后的静态资源。
webpack 的两大特色：
1.code splitting（可以自动完成）
2.loader 可以处理各种类型的静态文件，并且支持串联操作
webpack 是以 commonJS 的形式来书写脚本滴，但对 AMD/CMD 的支持也很全面，方便旧

项目进行代码迁移。

webpack 具有 requireJs 和 browserify 的功能，但仍有很多自己的新特性：

1. 对 CommonJS 、 AMD、ES6 的语法做了兼容
2. 对 js、css、图片等资源文件都支持打包
3. 串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性，例如提供对 CoffeeScript、ES6 的支持
4. 有独立的配置文件 webpack.config.js
5. 可以将代码切割成不同的 chunk，实现按需加载，降低了初始化时间
6. 支持 SourceUrls 和 SourceMaps，易于调试
7. 具有强大的 Plugin 接口，大多是内部插件，使用起来比较灵活
8.webpack 使用异步 IO 并具有多级缓存。这使得 webpack 很快且在增量编译上更加快

9、创建 ajax 过程

(1)创建 XMLHttpRequest 对象,也就是创建一个异步调用对象.
(2)创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息.
(3)设置响应 HTTP 请求状态变化的函数.
(4)发送 HTTP 请求.
(5)获取异步调用返回的数据.
(6)使用 JavaScript 和 DOM 实现局部刷新.

10、快速 排序的思想并实现一个快速排序？

“快速排序”的思想很简单，整个排序过程只需要三步：

（1）在数据集之中，找一个基准点
（2）建立两个数组，分别存储左边和右边的数组
（3）利用递归进行下次比较

```
<scripttype="text/javascript">
    functionquickSort(arr){
      if(arr.length<=1){
        return arr;//如果数组只有一个数，就直接返回；
      }

      var num = Math.floor(arr.length/2);//找到中间数的索引值，如果是浮点数，则向下取整
      var numValue =arr.splice(num,1);//找到中间数的值
      var left = [];
      var right = [];
      for(var i=0;i<arr.length;i++){
        if(arr[i]<numValue){
          left.push(arr[i]);//基准点的左边的数传到左边数组
        }
        else{
         right.push(arr[i]);//基准点的右边的数传到右边数组
```

```
            }
        }
        returnquickSort(left).concat([numValue],quickSort(right));//递归不断重复比较
    }

    alert(quickSort([32,45,37,16,2,87]));//弹出“2,16,32,37,45,87”
</script>
```

11、cookie 和 session 的区别：

1、cookie 数据存放在客户的浏览器上，session 数据放在服务器上。
2、cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗
考虑到安全应当使用 session。
3、session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能
考虑到减轻服务器性能方面，应当使用 COOKIE。
4、单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。
5、所以个人建议：
将登陆信息等重要信息存放为 SESSION
其他信息如果需要保留，可以放在 COOKIE 中

12、CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3 新增伪类有那些？

1.id 选择器（ # myid）
2.类选择器（.myclassname）
3.标签选择器（div, h1, p）
4.相邻选择器（h1 + p）
5.子选择器（ul > li）
6.后代选择器（lia）
7.通配符选择器（ * ）
8.属性选择器（a[rel = "external"]）
9.伪类选择器（a: hover, li:nth-child）

优先级为:
!important > id > class > tag
important 比 内联优先级高,但内联比 id 要高
CSS3 新增伪类举例：
p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。
p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。
:enabled :disabled 控制表单控件的禁用状态。

:checked　　单选框或复选框被选中。

13、浏览器输入 url 到页面呈现出来发生了什么？

1.进行地址解析

解析出字符串地址中的主机，域名，端口号，参数等

2.根据解析出的域名进行 DNS 解析

首先在浏览器中查找 DNS 缓存中是否有对应的 ip 地址，如果有就直接使用，没有机执行第二步

在操作系统中查找 DNS 缓存是否有对应的 ip 地址，如果有就直接使用，没有就执行第三步

向本地 DNS 服务商发送请求查找时候有 DNS 对应的 ip 地址。如果仍然没有最后向 Root Server 服务商查询。

3.根据查询到的 ip 地址寻找目标服务器

与服务器建立连接

进入服务器，寻找对应的请求

4.浏览器接收到响应码开始处理。

5.浏览器开始渲染 dom，下载 css、图片等一些资源。直到这次请求完成

软件测试面试题

1. 软件的生命周期：

计划阶段（planning）-〉需求分析（requirement）-〉设计阶段（design）-〉编码（coding）->测试（testing）->运行与维护（running maintrnacne）

2. 什么是软件测试，软件测试的原则和目的

在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

软件测试的目的：

测试是程序的执行过程，目的在于发现错误

一个成功的测试用例在于发现至今未发现的错误

一个成功的测试是发现了至今未发现的错误的测试

确保产品完成了它所承诺或公布的功能，并且用户可以访问到的功能都有明确的书面说明。

确保产品满足性能和效率的要求

确保产品是健壮的和适应用户环境的

软件测试的原则：

测试用例中一个必须部分是对预期输出或接过进行定义

程序员应避免测试自己编写的程序

编写软件的组织不应当测试自己编写的软件

应当彻底检查每个测试的执行结果

测试用例的编写不仅应当根据有效和预料到的输入情况，而且也应当根据无效和未预料到的输入情况

检擦程序是否"未做其应该做的"仅是测试的一半，测试的另一半是检查程序是否"做了其不应该做的"

应避免测试用例用后即弃，除非软件本身就是个一次性的软件

计划测试工作时不应默许假定不会发现错误

程序某部分存在更多错误的可能性，与该部分已经发现错误的数量成正比

软件测试是一项极富创造性，极具智力的挑战性的工作

3. **目前主要的测试用例设计方法是什么？**

白盒测试：逻辑覆盖、循环覆盖、基本路径覆盖

黑盒测试：边界值分析法、等价类划分、错误猜测法、因果图法、状态图法、测试大纲法、随机测试、场景法

4. **软件的安全性应从哪几个方面去测试？**

软件安全性测试包括程序、数据库安全性测试。根据系统安全指标不同测试策略也不同。

用户认证安全的测试要考虑问题： 明确区分系统中不同用户权限 、系统中会不会出现用户冲突 、系统会不会因用户的权限的改变造成混乱 、用户登陆密码是否是可见、可复制 、是否可以通过绝对途径登陆系统（拷贝用户登陆后的链接直接进入系统）、用户退出系统后是否删除了所有鉴权标记，是否可以使用后退键而不通过输入口令进入系统 、系统网络安全的测试要考虑问题 、测试采取的防护措施是否正确装配好，有关系统的补丁是否打上 、模拟非授权攻击，看防护系统是否坚固 、采用成熟的网络漏洞检查工具检查系统相关漏洞（即用最专业的黑客攻击工具攻击试一下，现在最常用的是 NBSI 系列和 IPhacker IP ） 、采用各种木马检查工具检查系统木马情况 、采用各种防外挂工具检查系统各组程序的外挂漏洞

数据库安全考虑问题： 系统数据是否机密（比如对银行系统，这一点就特别重要，一般的网站就没有太高要求）、系统数据的完整性（我刚刚结束的企业实名核查服务系统中就曾存在数据 的不完整，对于这个系统的功能实现有了障碍） 、系统数据可管理性

、系统数据的独立性 、系统数据可备份和恢复能力（数据备份是否完整，可否恢复，恢复是否可以完整）

5. **简述什么是静态测试、动态测试、黑盒测试、白盒测试、α测试 β测试**

静态测试是不运行程序本身而寻找程序代码中可能存在的错误或评估程序代码的过程。

动态测试是实际运行被测程序，输入相应的测试实例，检查运行结果与预期结果的差异，判定执行结果是否符合要求，从而检验程序的正确性、可靠性和有效性，并分析系统运行效率和健壮性等性能。

黑盒测试一般用来确认软件功能的正确性和可操作性,目的是检测软件的各个功能是否能得以实现,把被测试的程序当作一个黑盒,不考虑其内部结构,在知道该程序的输入和输出之间的关系或程序功能的情况下,依靠软件规格说明书来确定测试用例和推断测试结果的正确性。

白盒测试根据软件内部的逻辑结构分析来进行测试,是基于代码的测试，测试人员通过阅读程序代码或者通过使用开发工具中的单步调试来判断软件的质量，一般黑盒测试由项目经理在程序员开发中来实现。

α测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，Alpha 测试不能由程序员或测试员完成。

β测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，Beta 测试不能由程序员或测试员完成。

6. **软件测试分为几个阶段 各阶段的测试策略和要求是什么?**

和开发过程相对应，测试过程会依次经历单元测试、集成测试、系统测试、验收测试四个主要阶段：

单元测试：单元测试是针对软件设计的最小单位—程序模块甚至代码段进行正确性检验的测试工作，通常由开发人员进行。

集成测试：集成测试是将模块按照设计要求组装起来进行测试，主要目的是发现与接口有关的问题。由于在产品提交到测试部门前，产品开发小组都要进行联合调试，因此在大部分企业中集成测试是由开发人员来完成的。

系统测试：系统测试是在集成测试通过后进行的，目的是充分运行系统，验证各子系统是否都能正常工作并完成设计的要求。它主要由测试部门进行，是测试部门最大最重要的一个测试，对产品的质量有重大的影响。

验收测试：验收测试以需求阶段的《需求规格说明书》为验收标准，测试时要求模拟实际用户的运行环境。对于实际项目可以和客户共同进行，对于产品来说就是最后一次的系统测试。测试内容为对功能模块的全面测试，尤其要进行文档测试。

单元测试测试策略：

自顶向下的单元测试策略：比孤立单元测试的成本高很多，不是单元测试的一个好的选择。

自底向上的单元测试策略：比较合理的单元测试策略，但测试周期较长。

孤立单元测试策略：最好的单元测试策略。

集成测试的测试策略：

大爆炸集成：适应于一个维护型项目或被测试系统较小

自顶向下集成：适应于产品控制结构比较清晰和稳定；高层接口变化较小；底层接口未定义或经常可能被修改；产口控制组件具有较大的技术风险，需要尽早被验证；希望尽早能看到产品的系统功能行为。

自底向上集成：适应于底层接口比较稳定；高层接口变化比较频繁；底层组件较早被完成。

基于进度的集成
优点：具有较高的并行度；能够有效缩短项目的开发进度。
缺点：桩和驱动工作量较大；有些接口测试不充分；有些测试重复和浪费。

系统测试的测试策略：

数据和数据库完整性测试；功能测试；用户界面测试；性能评测；负载测试；强度测试；容量测试；安全性和访问控制测试；故障转移和恢复测试；配置测试；安装测试；加密测试；可用性测试；版本验证测试；文档测试

7. 在您以往的工作中，一条软件缺陷（或者叫 Bug）记录都包含了哪些内容？如何提交高质量的软件缺陷（Bug）记录？

一条 Bug 记录最基本应包含：

bug 编号；
bug 严重级别，优先级；
bug 产生的模块；
首先要有 bug 摘要，阐述 bug 大体的内容；
bug 对应的版本；
bug 详细现象描述，包括一些截图、录像....等等；
bug 出现时的测试环境，产生的条件即对应操作步骤；

高质量的 Bug 记录：

1) 通用 UI 要统一、准确
缺陷报告的 UI 要与测试的软件 UI 保持一致，便于查找定位。
2) 尽量使用业界惯用的表达术语和表达方法
使用业界惯用的表达术语和表达方法，保证表达准确，体现专业化。
3) 每条缺陷报告只包括一个缺陷
每条缺陷报告只包括一个缺陷，可以使缺陷修正者迅速定位一个缺陷，集中精力每次只修正一个缺陷。校验者每次只校验一个缺陷是否已经正确修正。
4) 不可重现的缺陷也要报告
首先缺陷报告必须展示重现缺陷的能力。不可重现的缺陷要尽力重现，若尽

力之后仍不能重现，仍然要报告此缺陷，但在报告中要注明无法再现，缺陷出现的频率。

5）明确指明缺陷类型

根据缺陷的现象，总结判断缺陷的类型。例如，即功能缺陷、界面缺陷、数据缺陷，合理化建议这是最常见的缺陷或缺陷类型，其他形式的缺陷或缺陷也从属于其中某种形式。

6）明确指明缺陷严重等级和优先等级

时刻明确严重等级和优先等级之间的差别。高严重问题可能不值得解决，小装饰性问题可能被当作高优先级。

7）描述（Description），简洁、准确，完整，揭示缺陷实质，记录缺陷或缺陷出现的位置

描述要准确反映缺陷的本质内容，简短明了。为了便于在软件缺陷管理数据库中寻找制定的测试缺陷，包含缺陷发生时的用户界面（UI）是个良好的习惯。例如记录对话框的标题、菜单、按钮等控件的名称。

8）短行之间使用自动数字序号，使用相同的字体、字号、行间距

短行之间使用自动数字序号，使用相同的字体、字号、行间距，可以保证各条记录格式一致，做到规范专业。

9）每一个步骤尽量只记录一个操作

保证简洁、条理井然，容易重复操作步骤。

10）确认步骤完整，准确，简短

保证快速准确的重复缺陷，“完整”即没有缺漏，“准确”即步骤正确，“简短”即没有多余的步骤。

11）根据缺陷，可选择是否进行图象捕捉

为了直观的观察缺陷或缺陷现象，通常需要附加缺陷或缺陷出现的界面，以图片的形式作为附件附着在记录的“附件”部分。为了节省空间，又能真实反映缺陷或缺陷本质，可以捕捉缺陷或缺陷产生时的全屏幕，活动窗口和局部区域。为了迅速定位、修正缺陷或缺陷位置，通常要求附加中文对照图。

附加必要的特殊文档和个人建议和注解

如果打开某个特殊的文档而产生的缺陷或缺陷，则必须附加该文档，从而可以迅速再现缺陷或缺陷。有时，为了使缺陷或缺陷修正者进一步明确缺陷或缺陷的表现，可以附加个人的修改建议或注解。

12）检查拼写和语法缺陷

在提交每条缺陷或缺陷之前，检查拼写和语法，确保内容正确，正确的描述缺陷。

13）尽量使用短语和短句，避免复杂句型句式

软件缺陷管理数据库的目的是便于定位缺陷，因此，要求客观的描述操作步骤，不需要修饰性的词汇和复杂的句型，增强可读性。

以上概括了报告测试缺陷的规范要求，随着软件的测试要求不同，测试者经过长期测试，积累了相应的测试经验，将会逐渐养成良好的专业习惯，不断补充新的规范书写要求。此外，经常阅读、学习其他测试工程师的测试缺陷报告，结合自己以前的测试缺陷报告进行对比和思考，可以不断提高技巧。

14）缺陷描述内容

缺陷描述的内容可以包含缺陷操作步骤，实际结果和期望结果。操作步骤可以方便开发人员再现缺陷进行修正，有些开发的再现缺陷能力很差，虽然他

明白你所指的缺陷，但就是无法再现特别是对系统不熟悉的新加入开发人员，介绍步骤可以方便他们再现。实际结果可以让开发明白错误是什么，期望结果可以让开发了解正确的结果应该是如何。

8. **黑盒测试和白盒测试是软件测试的两种基本方法，请分别说明各自的优点和缺点！**

黑盒测试的优点有：比较简单，不需要了解程序内部的代码及实现；与软件的内部实现无关； 从用户角度出发，能很容易的知道用户会用到哪些功能，会遇到哪些问题；基于软件开发文档，所以也能知道软件实现了文档中的哪些功能；在做软件自动化测试时较为方便。

黑盒测试的缺点有：不可能覆盖所有的代码，覆盖率较低，大概只能达到总代码量的 30%；自动化测试的复用性较低。

白盒测试的优点有：帮助软件测试人员增大代码的覆盖率，提高代码的质量，发现代码中隐 藏的问题。

白盒测试的缺点有：程序运行会有很多不同的路径，不可能测试所有的运行路径；测试基于代码，只能测试开发人员做的对不对，而不能知道设计的正确与否，可能会漏掉一些功能需求；系统庞大时，测试开销会非常大。

9. **详细的描述一个测试活动完整的过程**。（供参考，本答案主要是瀑布模型的做法）

项目经理通过和客户的交流，完成需求文档，由开发人员和测试人员共同完成需求文档的评审，评审的内容包括：需求描述不清楚的地方和可能有明显冲突或者无法实现的功能的地方。项目经理通过综合开发人员，测试人员以及客户的意见，完成项目计划。然后 SQA 进入项目，开始进行统计和跟踪

开发人员根据需求文档完成需求分析文档，测试人员进行评审，评审的主要内容包括是否有遗漏或双方理解不同的地方。测试人员完成测试计划文档，测试计划包括的内容上面有描述。

测试人员根据修改好的需求分析文档开始写测试用例，同时开发人员完成概要设计文档，详细设计文档。此两份文档成为测试人员撰写测试用例的补充材料。

测试用例完成后，测试和开发需要进行评审。

测试人员搭建环境

开发人员提交第一个版本，可能存在未完成功能，需要说明。测试人员进行测试，发现 BUG 后提交给 BugZilla。

开发提交第二个版本，包括 Bug Fix 以及增加了部分功能，测试人员进行测试。

重复上面的工作，一般是 3-4 个版本后 BUG 数量减少，达到出货的要求。

如果有客户反馈的问题，需要测试人员协助重现并重新测试。

10. **软件验收测试包括什么？**

软件验收测试包括：正式验收测试、alpha 测试、beta 测试三种测试。

11. **系统测试的策略**

系统测试的策略有很多种的，有性能测试、负载测试、强度测试、易用性测试、安全测试、配置测试、安装测试、文档测试、故障恢复测试、用户界面测试、恢复测试、分布测试、可用性测试。

12. **软件测试项目从什么时候开始，？为什么？**

软件测试应该在需求分析阶段就介入,因为测试的对象不仅仅是程序编码,应该对软件开发过程中产生的所有产品都测试,并且软件缺陷存在放大趋势.缺陷发现的越晚,修复它所花费的成本就越大.

13. **什么是回归测试?**

回归测试: (regression testing): 回归测试有两类：用例回归和错误回归；用例回归是过一段时间以后再回头对以前使用过的用例在重新进行测试,看看会重新发现问题。错误回归，就是在新版本中，对以前版本中出现并修复的缺陷进行再次验证，并以缺陷为核心，对相关修改的部分进行测试的方法。

14. **单元测试、集成测试、系统测试的侧重点是什么？**

单元测试针对的是软件设计的最小单元--程序模块（面向过程中是函数、过程；面向对象中是类。）,进行正确性检验的测试工作,在于发现每个程序模块内部可能存在的差错.一般有两个步骤:人工静态检查\动态执行跟踪

集成测试针对的是通过了单元测试的各个模块所集成起来的组件进行检验,其主要内容是各个单元模块之间的接口,以及各个模块集成后所实现的功能.

系统测试针对的是集成好的软件系统，作为整个计算机系统的一个元素,与计算机硬件\外设\某些支持软件\数据和人员等其他系统元素结合在一起,要在实际的运行环境中,对计算机系统进行一系列的集成测试和确认测试.

15. **测试结束的标准是什么？**

从微观上来说，在测试计划中定义，比如系统在一定性能下平稳运行 72 小时，目前 Bug Tracking System 中，本版本中没有一般严重的 BUG，普通 BUG 的数量在 3 以下，BUG 修复率 90%以上等等参数，然后由开发经理，测试经理，项目经理共同签字认同版本 Release。

如果说宏观的，则是当这个软件彻底的消失以后，测试就结束了。

16. **LoadRunner 分为哪三个模块？请简述各模块的主要功能。**

Virtual User Generator：用于录制脚步

Mercury LoadRunner Controller：用于创建、运行和监控场景

17. **你对测试最大的兴趣在哪里？为什么？**

最大的兴趣就是测试有难度，有挑战性！做测试越久越能感觉到做好测试有多难。曾经在无忧测试网上看到一篇文章，是关于如何做好一名测试工程师。一共罗列了 11，12 点，有部分是和人的性格有关，有部分需要后天的努力。但除了性格有关的 1，2 点我没有把握，其他点我都很有信心做好它。

刚开始进入测试行业时，对测试的认识是从无忧测试网上了解到的一些资料，当时是冲着做测试需要很多技能才能做的好，虽然入门容易，但做好很难，比开发更难，虽然当时我很想

做开发（学校专业课我基本上不缺席，因为我喜欢我的专业），但看到测试比开发更难更有挑战性，想做好测试的意志就更坚定了。

我觉得做测试整个过程中有 2 点让我觉得很有难度（对我来说，有难度的东西我就非常感兴趣），第一是测试用例的设计，因为测试的精华就在测试用例的设计上了，要在版本出来之前，把用例写好，用什么测试方法写？（也就是测试计划或测试策略），如果你刚测试一个新任务时，你得花一定的时间去消化业务需求和技术基础，业务需求很好理解（多和产品经理和开发人员沟通就能达到目的），而技术基础可就没那么简单了，这需要你自觉的学习能力，比如说网站吧，最基本的技术知识你要知道网站内部是怎么运作的的，后台是怎么响应用户请求的？测试环境如何搭建？这些都需要最早的学好。至少在开始测试之前能做好基本的准备，可能会遇到什么难题？需求细节是不是没有确定好？这些问题都能在设计用例的时候发现。

第二是发现 BUG 的时候了，这应该是测试人员最基本的任务了，一般按测试用例开始测试就能发现大部分的 bug，还有一部分 bug 需要测试的过程中更了解所测版本的情况获得更多信息，补充测试用例，测试出 bug。还有如何发现 bug？这就需要在测试用例有效的情况下，通过细心和耐心去发现 bug 了，每个用例都有可能发现 bug，每个地方都有可能出错，所以测试过程中思维要清晰（测试过程数据流及结果都得看仔细了，bug 都在里面发现的）。如何描述 bug 也很有讲究，bug 在什么情况下会产生，如果条件变化一点点，就不会有这个 bug，以哪些最少的操作步骤就能重现这个 bug，这个 bug 产生的规律是什么？如果你够厉害的话，可以帮开发人员初步定位问题。

Mercury LoadRunner Analysis：用于分析测试结果

18. 当开发人员说不是 BUG 时，你如何应付？

开发人员说不是 bug，有 2 种情况，一是需求没有确定，所以我可以这么做，这个时候可以找来产品经理进行确认，需不需要改动，3 方商量确定好后再看要不要改。二是这种情况不可能发生，所以不需要修改，这个时候，我可以先尽可能的说出是 BUG 的依据是什么？如果被用户发现或出了问题，会有什么不良结果？程序员可能会给你很多理由，你可以对他的解释进行反驳。如果还是不行，那我可以给这个问题提出来,跟开发经理和测试经理进行确认,如果要修改就改,如果不要修改就不改。其实有些真的不是 bug，我也只是建议的方式写进 TD 中，如果开发人员不修改也没有大问题。如果确定是 bug 的话，一定要坚持自己的立场，让问题得到最后的确认。

19. 简述 bug 的生命周期?

1， 有效地记录 BUG 2， 使用 BUG 模板 3， 评价 BUG 优先级和严重性 4， BUG 的生命 5， 维护 BUG 数据库

运维面试题

1.什么是运维？什么是游戏运维？

1）运维是指大型组织已经建立好的网络软硬件的维护，就是要保证业务的上线与运作的正常，

在他运转的过程中，对他进行维护，他集合了网络、系统、数据库、开发、安全、监控于一身的技术
运维又包括很多种，有 DBA 运维、网站运维、虚拟化运维、监控运维、游戏运维等等

2）游戏运维又有分工，分为开发运维、应用运维（业务运维）和系统运维
开发运维：是给应用运维开发运维工具和运维平台的
应用运维：是给业务上线、维护和做故障排除的，用开发运维开发出来的工具给业务上线、维护、做故障排查
系统运维：是给应用运维提供业务上的基础设施，比如：系统、网络、监控、硬件等等

总结：开发运维和系统运维给应用运维提供了"工具"和"基础设施"上的支撑
开发运维、应用运维和系统运维他们的工作是环环相扣的

2. 工作中，运维人员经常需要跟运营人员打交道，请问运营人员是做什么工作的？
游戏运营要做的一个事情除了协调工作以外
还需要与各平台沟通，做好开服的时间、开服数、用户导量、活动等计划

3. raid0 raid1 raid5 三种工作模式的工作原理及特点

RAID，可以把硬盘整合成一个大磁盘，还可以在大磁盘上再分区，放数据
还有一个大功能，多块盘放在一起可以有冗余（备份）
RAID 整合方式有很多，常用的：0 1 5 10

RAID 0，可以是一块盘和 N 个盘组合

其优点读写快，是 RAID 中最好的

缺点：没有冗余，一块坏了数据就全没有了
RAID 1，只能 2 块盘，盘的大小可以不一样，以小的为准

10G+10G 只有 10G，另一个做备份。它有 100%的冗余，缺点：浪费资源，成本高
RAID 5 ，3 块盘，容量计算 10*（n-1）,损失一块盘

特点，读写性能一般，读还好一点，写不好
冗余从好到坏：RAID1 RAID10 RAID 5 RAID0
性能从好到坏：RAID0 RAID10 RAID5 RAID1
成本从低到高：RAID0 RAID5 RAID1 RAID10
单台服务器：很重要盘不多，系统盘，RAID1
数据库服务器：主库：RAID10 从库 RAID5RAID0（为了维护成本，RAID10）
WEB 服务器，如果没有太多的数据的话，RAID5,RAID0（单盘）
有多台，监控、应用服务器，RAID0 RAID5
我们会根据数据的存储和访问的需求，去匹配对应的 RAID 级别

4. squid、Varinsh 和 Nginx 有什么区别，工作中你怎么选择？

Squid、Varinsh 和 Nginx 都是代理服务器
什么是代理服务器：
能当替用户去访问公网，并且能把访问到的数据缓存到服务器本地，等用户下次再访问相同

的资
源的时候，代理服务器直接从本地回应给用户，当本地没有的时候，我代替你去访问公网，我接
收你的请求，我先在我自己的本地缓存找，如果我本地缓存有，我直接从我本地的缓存里回复你
如果我在我本地没有找到你要访问的缓存的数据，那么代理服务器就会代替你去访问公网

区别：
1）Nginx 本来是反向代理/web 服务器，用了插件可以做做这个副业

但是本身不支持特性挺多，只能缓存静态文件
2）从这些功能上。varnish 和 squid 是专业的 cache 服务，而 nginx 这些是第三方模块完成
3）varnish 本身的技术上优势要高于 squid，它采用了可视化页面缓存技术
在内存的利用上，Varnish 比 Squid 具有优势，性能要比 Squid 高。
还有强大的通过 Varnish 管理端口，可以使用正则表达式快速、批量地清除部分缓存
它是内存缓存，速度一流，但是内存缓存也限制了其容量，缓存页面和图片一般是挺好的
4）squid 的优势在于完整的庞大的 cache 技术资料，和很多的应用生产环境
工作中选择：
要做 cache 服务的话，我们肯定是要选择专业的 cache 服务，优先选择 squid 或者 varnish。
5.Tomcat 和 Resin 有什么区别，工作中你怎么选择？
区别：Tomcat 用户数多，可参考文档多，Resin 用户数少，可考虑文档少
最主要区别则是 Tomcat 是标准的 java 容器，不过性能方面比 resin 的要差一些
但稳定性和 java 程序的兼容性，应该是比 resin 的要好

工作中选择：现在大公司都是用 resin，追求性能；而中小型公司都是用 Tomcat，追求稳定和程序的兼容

5.什么是中间件？什么是 jdk？

中间件介绍：
中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源
中间件位于客户机/ 服务器的操作系统之上，管理计算机资源和网络通讯
是连接两个独立应用程序或独立系统的软件。相连接的系统，即使它们具有不同的接口

但通过中间件相互之间仍能交换信息。执行中间件的一个关键途径是信息传递
通过中间件，应用程序可以工作于多平台或 OS 环境。

jdk：jdk 是 Java 的开发工具包
它是一种用于构建在 Java 平台上发布的应用程序、applet 和组件的开发环境

6.讲述一下 Tomcat8005、8009、8080 三个端口的含义？

8005==》 关闭时使用
8009==》 为 AJP 端口，即容器使用，如 Apache 能通过 AJP 协议访问 Tomcat 的 8009 端口
8080==》 一般应用使用

7. 简述 DNS 进行域名解析的过程？

用户要访问 www.baidu.com，会先找本机的 host 文件，再找本地设置的 DNS 服务器，如果也没有的话，就去网络中找根服务器，根服务器反馈结果，说只能提供一级域名服务器.cn，就去找一级域名服务器，一级域名服务器说只能提供二级域名服务器.com.cn,就去找二级域名服务器，二级域服务器只能提供三级域名服务器.baidu.com.cn，就去找三级域名服务器，三级域名服务器正好有这个网站 www.baidu.com，然后发给请求的服务器，保存一份之后，再发给客户端。

8. 讲一下 Keepalived 的工作原理？

在一个虚拟路由器中，只有作为 MASTER 的 VRRP 路由器会一直发送 VRRP 通告信息, BACKUP 不会抢占 MASTER，除非它的优先级更高。当 MASTER 不可用时(BACKUP 收不到通告信息)
多台 BACKUP 中优先级最高的这台会被抢占为 MASTER。这种抢占是非常快速的(<1s)，以保证服务的连续性
由于安全性考虑，VRRP 包使用了加密协议进行加密。BACKUP 不会发送通告信息，只会接收通告信息

9. 讲述一下 LVS 三种模式的工作过程？

LVS 有三种负载均衡的模式，分别是 VS/NAT（nat 模式） VS/DR(路由模式) VS/TUN（隧道模式）

1）NAT 模式（VS-NAT）

原理：就是把客户端发来的数据包的 IP 头的目的地址，在负载均衡器上换成其中一台 RS 的 IP 地址

并发至此 RS 来处理,RS 处理完后把数据交给负载均衡器,负载均衡器再把数据包原 IP 地址改为自己的 IP

将目的地址改为客户端 IP 地址即可期间,无论是进来的流量,还是出去的流量,都必须经过负载均衡器

优点：集群中的物理服务器可以使用任何支持 TCP/IP 操作系统，只有负载均衡器需要一个合法的 IP 地址

缺点：扩展性有限。当服务器节点（普通 PC 服务器）增长过多时,负载均衡器将成为整个系统的瓶颈

因为所有的请求包和应答包的流向都经过负载均衡器。当服务器节点过多时

大量的数据包都交汇在负载均衡器那，速度就会变慢！2)IP 隧道模式（VS-TUN）

原理：首先要知道，互联网上的大多 Internet 服务的请求包很短小，而应答包通常很大

那么隧道模式就是，把客户端发来的数据包，封装一个新的 IP 头标记(仅目的 IP)发给 RS

RS 收到后,先把数据包的头解开,还原数据包,处理后,直接返回给客户端,不需要再经过

负载均衡器。注意,由于 RS 需要对负载均衡器发过来的数据包进行还原,所以说必须支持

IPTUNNEL 协议，所以,在 RS 的内核中,必须编译支持 IPTUNNEL 这个选项

优点：负载均衡器只负责将请求包分发给后端节点服务器，而 RS 将应答包直接发给用户

所以，减少了负载均衡器的大量数据流动，负载均衡器不再是系统的瓶颈，就能处理很巨大的请求量

这种方式，一台负载均衡器能够为很多 RS 进行分发。而且跑在公网上就能进行不同地域的分发。

缺点：隧道模式的 RS 节点需要合法 IP，这种方式需要所有的服务器支持"IP Tunneling"

(IP Encapsulation)协议，服务器可能只局限在部分 Linux 系统上

3)直接路由模式（VS-DR）

原理：负载均衡器和 RS 都使用同一个 IP 对外服务但只有 DR 对 ARP 请求进行响应

所有 RS 对本身这个 IP 的 ARP 请求保持静默也就是说,网关会把对这个服务 IP 的请求全部定向给 DR

而 DR 收到数据包后根据调度算法,找出对应的 RS,把目的 MAC 地址改为 RS 的 MAC（因为 IP 一致）

并将请求分发给这台 RS 这时 RS 收到这个数据包,处理完成之后，由于 IP 一致，可以直接将数据返给客户则等于直接从客户端收到这个数据包无异,处理后直接返回给客户端

由于负载均衡器要对二层包头进行改换,所以负载均衡器和 RS 之间必须在一个广播域

也可以简单的理解为在同一台交换机上

优点：和 TUN（隧道模式）一样，负载均衡器也只是分发请求，应答包通过单独的路由方法返回给客户端

与 VS-TUN 相比，VS-DR 这种实现方式不需要隧道结构，因此可以使用大多数操作系统做为物理服务器。

缺点：（不能说缺点，只能说是不足）要求负载均衡器的网卡必须与物理网卡在一个物理段上

10. 如何重置 mysql root 密码？

1) 在已知 MYSQL 数据库的 ROOT 用户密码的情况下，修改密码的方法：

A. 在 SHELL 环境下，使用 mysqladmin 命令设置：

mysqladmin –u root –p password “新密码” 回车后要求输入旧密码

b.在 mysql>环境中,使用 update 命令，直接更新 mysql 库 user 表的数据：

update mysql.user set password=password(‘新密码’)where user=’root’;

mysql> flush rivileges;

注意：mysql 语句要以分号”；”结束

c.在 mysql>环境中，使用 grant 命令，修改 root 用户的授权权限。 grant all on *.* to root@’localhost’identified by‘新密码’;

2)如查忘记了 mysql 数据库的 ROOT 用户的密码，又如何做呢？方法如下：

a.当前运行的 mysqld 服务程序：service mysqld stop（要先将 mysqld 添加为系统服务）

b.mysqld_safe 脚本以安全模式（不加载授权表）启动 mysqld 服务

/usr/local/mysql/bin/mysqld_safe --skip-grant-table

c.空密码的 root 用户登录数据库，重新设置 ROOT 用户的密码

mysql -uroot

mysql>update mysql.user set password=new password(‘新密码’) where user=’root’;

mysql>flush privileges;

11.lvs/nginx/haproxy 优缺点

Nginx 的优点是：

1)工作在网络的 7 层之上，可以针对 http 应用做一些分流的策略，比如针对域名、目录结构

它的正则规则比 HAProxy 更为强大和灵活，这也是它目前广泛流行的主要原因之一

Nginx 单凭这点可利用的场合就远多于 LVS 了。

2)Nginx 对网络稳定性的依赖非常小，理论上能 ping 通就就能进行负载功能，这个也是它的优势之一

相反 LVS 对网络稳定性依赖比较大，这点本人深有体会；

3)Nginx 安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出来

LVS 的配置、测试就要花比较长的时间了，LVS 对网络依赖比较大。

4)可以承担高负载压力且稳定，在硬件不差的情况下一般能支撑几万次的并发量，负载度比 LVS 相对小些。

5)Nginx 可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持 url

来检测。比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障，Nginx 会把上传切到另一台服务器重新处理，而 LVS 就直接断掉了

如果是上传一个很大的文件或者很重要的文件的话，用户可能会因此而不满。

6)Nginx 不仅仅是一款优秀的负载均衡器/反向代理软件，它同时也是功能强大的 Web 应用服务器

LNMP 也是近几年非常流行的 web 架构，在高流量的环境中稳定性也很好。

7)Nginx 现在作为 Web 反向加速缓存越来越成熟了，速度比传统的 Squid 服务器更快，可考虑用其作为反向代理加速器

8)Nginx 可作为中层反向代理使用，这一层面 Nginx 基本上无对手，唯一可以对比 Nginx 的就只有 lighttpd 了

不过 lighttpd 目前还没有做到 Nginx 完全的功能，配置也不那么清晰易读，社区资料也远远没 Nginx 活跃

9)Nginx 也可作为静态网页和图片服务器，这方面的性能也无对手。还有 Nginx 社区非常活跃，第三方模块也很多

Nginx 的缺点是：

1)Nginx 仅能支持 http、https 和 Email 协议，这样就在适用范围上面小些，这个是它的缺点

2)对后端服务器的健康检查，只支持通过端口来检测，不支持通过 url 来检测

不支持 Session 的直接保持，但能通过 ip_hash 来解决

LVS：使用 Linux 内核集群实现一个高性能、高可用的负载均衡服务器

它具有很好的可伸缩性（Scalability)、可靠性（Reliability)和可管理性（Manageability)

LVS 的优点是：

1)抗负载能力强、是工作在网络 4 层之上仅作分发之用，没有流量的产生

这个特点也决定了它在负载均衡软件里的性能最强的，对内存和 cpu 资源消耗比较低

2)配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西

所以并不需要太多接触，大大减少了人为出错的几率

3)工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案

如 LVS+Keepalived，不过我们在项目实施中用得最多的 LVS/DR+Keepalived

4)无流量，LVS 只分发请求，而流量并不从它本身出去，这点保证了均衡器 IO 的性能不会收到大流量的影响。

5)应用范围较广，因为 LVS 工作在 4 层，所以它几乎可对所有应用做负载均衡，包括 http、数据库、在线聊天室等

LVS 的缺点是：

1)软件本身不支持正则表达式处理，不能做动静分离 而现在许多网站在这方面都有较强的需求，这个是 Nginx/HAProxy+Keepalived 的优势所在

2)如果是网站应用比较庞大的话，LVS/DR+Keepalived 实施起来就比较复杂了

特别后面有 Windows Server 的机器的话，如果实施及配置还有维护过程就比较复杂了相对而言，Nginx/HAProxy+Keepalived 就简单多了。

HAProxy 的特点是：

1)HAProxy 也是支持虚拟主机的。

2)HAProxy 的优点能够补充 Nginx 的一些缺点，比如支持 Session 的保持，Cookie 的引导同时支持通过获取指定的 url 来检测后端服务器的状态

3)HAProxy 跟 LVS 类似，本身就只是一款负载均衡软件单纯从效率上来讲 HAProxy 会比 Nginx 有更出色的负载均衡速度，在并发处理上也是优于 Nginx

4)HAProxy 支持 TCP 协议的负载均衡转发，可以对 MySQL 读进行负载均衡对后端的 MySQL 节点进行检测和负载均衡，大家可以用 LVS+Keepalived 对 MySQL 主从做负载均衡

5)HAProxy 负载均衡策略非常多，HAProxy 的负载均衡算法现在具体有如下 8 种：

①roundrobin，表示简单的轮询，这个不多说，这个是负载均衡基本都具备的；

② static-rr，表示根据权重，建议关注；

③leastconn，表示最少连接者先处理，建议关注；

④ source，表示根据请求源 IP，这个跟 Nginx 的 IP_hash 机制类似我们用其作为解决 session 问题的一种方法，建议关注；

⑤ri，表示根据请求的 URI；

⑥rl_param，表示根据请求的 URl 参数'balance url_param' requires an URL parameter name；

⑦hdr(name)，表示根据 HTTP 请求头来锁定每一次 HTTP 请求；

⑧rdp-cookie(name)，表示根据据 cookie(name)来锁定并哈希每一次 TCP 请求。

12. 你对现在运维工程师的理解和以及对其工作的认识

运维工程师在公司当中责任重大，需要保证时刻为公司及客户提供最高、最快、最稳定、最安全的服务

运维工程师的一个小小的失误，很有可能会对公司及客户造成重大损失

因此运维工程师的工作需要严谨及富有创新精神

13. Linux 系统中病毒怎么解决

1）最简单有效的方法就是重装系统

2）要查的话就是找到病毒文件然后删除

中毒之后一般机器 cpu、内存使用率会比较高

机器向外发包等异常情况，排查方法简单介绍下

top 命令找到 cpu 使用率最高的进程

一般病毒文件命名都比较乱，可以用 ps aux 找到病毒文件位置

rm -f 命令删除病毒文件

检查计划任务、开机启动项和病毒文件目录有无其他可以文件等

3）由于即使删除病毒文件不排除有潜伏病毒，所以最好是把机器备份数据之后重装一下

14. 发现一个病毒文件你删了他又自动创建怎么解决

公司的内网某台 linux 服务器流量莫名其妙的剧增,用 iftop 查看有连接外网的情况

针对这种情况一般重点查看 netstat 连接的外网 ip 和端口。

用 lsof -p pid 可以查看到具体是那些进程，哪些文件

经查勘发现/root 下有相关的配置 conf.n hhe 两个可疑文件，rm -rf 后不到一分钟就自动生成了

由此推断是某个母进程产生的这些文件。所以找到母进程就是找到罪魁祸首

查杀病毒最好断掉外网访问，还好是内网服务器，可以通过内网访问

断了内网，病毒就失去外联的能力，杀掉它就容易的多

怎么找到呢，找了半天也没有看到蛛丝马迹，没办法只有 ps axu 一个个排查

方法是查看可以的用户和和系统相似而又不是的冒牌货，果然，看到了如下进程可疑

看不到图片就是/usr/bin/.sshd

于是我杀掉所有.sshd 相关的进程，然后直接删掉.sshd 这个可执行文件

然后才删掉了文章开头提到的自动复活的文件

总结一下，遇到这种问题，如果不是太严重，尽量不要重装系统

一般就是先断外网，然后利用 iftop,ps,netstat,chattr,lsof,pstree 这些工具顺藤摸瓜

一般都能找到元凶。但是如果遇到诸如此类的问题

/boot/efi/EFI/redhat/grub.efi: Heuristics.Broken.Executable FOUND，个人觉得就要重装系统了

15. 说说 TCP/IP 的七层模型

应用层 (Application)：

网络服务与最终用户的一个接口。

协议有：HTTP FTP TFTP SMTP SNMP DNS TELNET HTTPS POP3 DHCP

表示层（Presentation Layer）：

数据的表示、安全、压缩。（在五层模型里面已经合并到了应用层）

格式有，JPEG、ASCll、DECOIC、加密格式等

会话层（Session Layer）：

建立、管理、终止会话。（在五层模型里面已经合并到了应用层）

对应主机进程，指本地主机与远程主机正在进行的会话

传输层 (Transport)：

定义传输数据的协议端口号，以及流控和差错校验。

协议有：TCP UDP，数据包一旦离开网卡即进入网络传输层

网络层 (Network)：

进行逻辑地址寻址，实现不同网络之间的路径选择。

协议有：ICMP IGMP IP（IPV4 IPV6） ARP RARP

数据链路层 (Link)：

建立逻辑连接、进行硬件地址寻址、差错校验等功能。（由底层网络定义协议）

将比特组合成字节进而组合成帧，用 MAC 地址访问介质，错误发现但不能纠正

物理层（Physical Layer）：是计算机网络 OSI 模型中最低的一层

物理层规定:为传输数据所需要的物理链路创建、维持、拆除而提供具有机械的，电子的，功能的和规范的特性

简单的说，物理层确保原始的数据可在各种物理媒体上传输。局域网与广域网皆属第 1、2 层

物理层是 OSI 的第一层，它虽然处于最底层，却是整个开放系统的基础

物理层为设备之间的数据通信提供传输媒体及互连设备，为数据传输提供可靠的环境

如果您想要用尽量少的词来记住这个第一层，那就是"信号和介质"

16. 请列出你了解的 web 服务器负载架构

Nginx

Haproxy

Keepalived

LVS

17. 如何优化 Linux 系统（可以不说太具体）？

不用 root，添加普通用户，通过 sudo 授权管理

更改默认的远程连接 SSH 服务端口及禁止 root 用户远程连接

定时自动更新服务器时间

配置国内 yum 源

关闭 selinux 及 iptables（iptables 工作场景如果有外网 IP 一定要打开，高并发除外）

调整文件描述符的数量

精简开机启动服务（crond rsyslog network sshd）

内核参数优化（/etc/sysctl.conf）

更改字符集，支持中文，但建议还是用英文字符集，防止乱码

锁定关键系统文件

清空/etc/issue，去除系统及内核版本登录前的屏幕显示

18.给你一套环境，你会如何设计高可用、高并发的架构？

如果这套环境是部署在云端(比如阿里云)，你就不用去考虑硬件设计的问题。可直接上阿里云的 SLB+ECS+RDS 这套标准的高可用、高并发的架构。对外服务直接上 SLB 负载均衡技术，由阿里的 SLB 分发到后端的 ECS 主机；ECS 主机部署多台，应用拆分在不同的 ECS 主机上，尽量细分服务。数据库用 RDS 高可用版本（一主一备的经典高可用架构）、或者用 RDS 金融版（一主两备的三节点架构）。在结合阿里其它的服务就完全 OK，业务量上来了，主机不够用了，直横向扩容 ECS 主机搞定。

如果这套环境托管在 IDC，那么你就要从硬件、软件（应用服务）双面去考虑了。硬件要达到高可用、高并发公司必须买多套网络硬件设备（比如负载设备 F5、防火墙、核心层交换、接入层交换）都必须要冗余，由其是在网络设计上，设备之间都必须有双线连接。设备如果都是跑的单机，其中一个设备挂了，你整个网络都瘫痪了，就谈不上高可用、高并发了。其次在是考虑应用服务了，对外服务我会采用成熟的开源方案 LVS+Keepalived 或者 Nginx+Keepalived，缓存层可以考虑 redis 集群及 Mongodb 集群，中间件等其它服务可以用 kafka、zookeeper，图片存储可以用 fastDFS 或 MFS，如果数据量大、又非常多，那么可采用 hadoop 这一套方案。后端数据库可采用“主从+MHA”。这样一套环境下来是绝对满足高可用、高并发的架构。

19.cpu 单核和多核有啥区别？

双核 CPU 就是能处理多份任务，顺序排成队列来处理。单核 CPU 一次处理一份任务，轮流处理每个程序任务。双核的优势不是频率，而是对付同时处理多件事情。单核同时只能干一件事，比如你同时在后台 BT 下载，前台一边看电影一边拷贝文件一边 QQ。

20. 机械磁盘和固态硬盘有啥区别？

HDD 代表机械硬盘，SSD 代表固态硬盘。首先，从性能方面来说，固态硬盘几乎完胜机械硬盘，固态硬盘的读写速度肯定要快机械硬盘，因为固态硬盘和机械硬盘的构造是完全不同的（具体的构造就没必要解释了）。其次，固态盘几乎没有噪音、而机械盘噪音比较大。还有就是，以目前的市场情况来看，一般机械盘容量大，价格低；固态盘容量小，价格偏高。但是企业还是首选固态盘

21. Tomcat 工作模式？

笔者回答：Tomcat 是一个 JSP/Servlet 容器。其作为 Servlet 容器，有三种工作模式：独立的 Servlet 容器、进程内的 Servlet 容器和进程外的 Servlet 容器。

进入 Tomcat 的请求可以根据 Tomcat 的工作模式分为如下两类：

Tomcat 作为应用程序服务器：请求来自于前端的 web 服务器，这可能是 Apache, IIS, Nginx 等；

Tomcat 作为独立服务器：请求来自于 web 浏览器；

实施面试题

1. 两电脑都在同一个网络环境中，A 电脑访问不到 B 电脑的共享文件。此现象可能是哪些 方面所导致？怎样处理？

首先你要确定是不是在一个工作组内，只有在一个工作组内才可以共享文件，然后看一个看一看有没有防火墙之类的，然后确定文件是不是已经共享。

2. 电脑开机时风扇转， 但是屏幕没有任何显示， 此现象可能是哪些方面所导致？怎样处理？

a.查看是否有报警声？如果没有接着看第二条。如果有可能是内存或者是显卡问题，建议都取下清理干净，看看是否 ok！在经手指部分用橡皮擦擦拭

b.主板问题，建议先去下 主板上的那块电池，放电主板！ 前提是要把全部的电源关掉！

c.电源问题，建议换个电源试试！

3. **在同一个网络环境中 A 电脑 IP： 192.168.1.100 ，电脑 IP： B 192.168.0.100， 路由器的 IP：192.168.1.1 。请问有那些方法可以让 B 电脑即可以上公网也可 以访问到 A 电脑？**

如果你的公网是需要拨号的，那么这需要两个路由器来进行转换，首先要使一个路由能上公网。再使用另一台路由器进行两个网段进行路由转换，在路由器中设置静态路由转换，一个为 192.168.1.*，一个为 192.168.0*,这样这两台电脑就能进行互访。把这台路由器和一台电脑分别连接到原公网的路由上就能使两个不同网段的电脑都能上网。如果你的公网不需要拨号，你可以把原连接公网的那台路由器更换为交换机就 OK，其他连接同上。

4. 你熟悉的远程有哪些方法？各种方法应该怎么配置？

a.最简单的 QQ 上有，打开对话框 上边有个 “应用”图标 点击“远程协助”

b.在要远程的主机安装 “客户机”（一个小软件）端 自己 安装“主机”（软件） 就可以

远程控制了 （网络上有下载）

c.还有就是一些大单位的专业通信系统 即时通 OA 之类的 有些远程协助功能（这种要花钱买）

5. **如果有一个不太懂电脑的客户，你应该采取什么样的方法去教他用公司的软件产品？**

我觉得这是一个面试的题目，提问题的人想从其中看出你的为人处事能力。这个可以灵活回答的，如果软件产品比较难懂，你就可以先教一些简单的功能。

再告诉他需要再了解哪些知识来掌握这个软件。如果软件产品比较简单，就可以直接一步一步的教他怎么操作，一直操作熟练就行了。

6. **在你进行实施的过程中，公司制作的一款软件系统缺少某一项功能，而且公司也明确表示不会再为系统做任何的修改或添加任何的功能，而客户也坚决要求需要这一项功能！对于实施人员来说，应该怎么去合理妥善处理这个问题？**

说清楚你实施的是一个项目，不是定制软件。比如超市里卖的皮鞋和鞋匠做的皮鞋，这都是鞋子，但前者是商品，很多一样的商品，你可以买到差不多可以穿的舒服的鞋子；后者是定制，不仅是商品，更是一种劳动，是已完全针对客户需求而生产的，每个细节都是要求完美的。软件也是，没有客户需求的功能，公司既然表明无法实现，那么肯定是功能研发所耗费的时间,财力和利益冲突，你可以对客户说明增加功能要他支付更多的费用，甚至比买软件的价格还贵。（这个主要考察看个人口才）

7. **在你进行实施的过程中，公司制作的一款软件系统缺少某一项功能，且公司也明确表示不会再为系统做任何的修改或者添加任何功能，而客户也坚决要求需要这一项功能！对于实施人员来说，应该怎样妥善处理这个问题？**

先看用户要求合不合理，不合理就可以坚决退还需求，如果需求合理的话，可以申请做二次开发，并且收取一定的费用，这个两边要沟通好。如果上述方法不奏效，使用第三方软件做补助。

8. **在项目实施过程中，使用者对产品提出了适合自己习惯的修改意见，但多个使用者意见互相矛盾，应该如何去处理？**

对于客户的意见，我们实施人员应该有自己的实施方案。当使用者意见出现不一致时候，我们应当引导他们内部意见达到统一和用户经过沟通确认后，找到切实可行的方案，双方认可并达成共识。

9. 同一个网络环境中，A 电脑访问不到 B 电脑的共享文件，此现象是哪些方面所导致？该怎样处理？

首先检查网络是否有问题，在确定是不是早同一个工作组内，只有在同一个工作组内才可以共享文件，燃火看有没有被防火墙阻止，最后确定文件是不是已经被共享。

10. 什么是 DHCP？如何快速为多台电脑安装操作系统？多台电脑如何组网？

1、DHCP 动态主机设置协议，是一个局域网的网络协议,使用 UDP 协议工作，主要有两个用途：给内部网络或网络服务自动分配 IP 地址，给用户以及内部网络管理员作为对所有计算机作中央管理的手段。

2、可以通过网络硬盘克隆，过程为：在装有软驱的工作站上，用一张引导盘来启动机器，连接到服务器，使用 Ghost 多播服务（Multicast Server）将硬盘或分区的映像克隆岛工作站，这样就实现了不拆机、安全、快速地网络硬盘克隆。

3.多台电脑组网可以分为两个类型：

少于 250 台：可以采用用户接入层和核心'接入层这二层网络结构，通过普通二层交换机与核心交换机的堆叠连接组层单位局域网，以满足单位各种上网访问需求。普通电脑通过双绞线，连接到普通百兆二层交换机。

超过 250 台：我们就需要通过交换机的 VLAN 功能，将他们划分到不同的子网中。为了让两网段中的所有电脑都能够实现共享上网的目的，我们还需要在核心路由交换机或者双 WAN 端口路由器设备中对两个网关参数进行合适的配置，确保各个子网中的电脑能通过局域网路由功能访问 Internert 网络。

11. 局域网内，一台机器不能上网，而其他机器可以。所有的机器都安装 WinXP 系统，且电脑可以访问局域网内电脑，试分析原因？

1、检查有无 Microosoft 网络客户端，Microsoft 网络的文件和打印机共享、Internet 协议（TCP）。

2、检查 IP 地址、网关、DNS、网络是否连接上等。

3、木马、病毒。

12. 如果有一个不太懂电脑的客户，你应该采用什么的方法去叫他使用公司的软件产品？

1、如果软件产品比较难懂，你就可以先教一些简单的。再告诉他需要了解哪些知识来掌握这个软件。

2、如果软件产品比较简单、就可以直接一步一步的教他怎样操作，一直操作熟练就可以了。

13. 当你的工作的付出和你的收入不成正比时你会怎样想？

无论做什么工作，必须做一行爱一行，脚踏实地、用心去钻研，只要真正有能力，只要有思想和技术，终会出头。钻石总会发光的。接受你不能接受的，改变你能改变的。会争取到更高的薪水的。如果当初进来的时候公司有加薪的承诺，那就看你的表现是否达到了要求，可以主动和相关领导沟通。如果实在无法忍受可以选择跳槽，毕竟工作是一个双向选择的事情。

14. 一旦数据库若出现日志满了，会出现什么情况，是否还能继续使用？

数据库满了就不能使用数据库年，数据库满是指文件达到设置的最大文件大小，没设置的时候就是最大可用磁盘空间只能执行查询、读取等操作，不能执行更改、备份等写操作，原因是任何读写操作都要记录日志。

15. 触发器的作用是什么？

触发器是针对数据表（库）的特殊存储过程，当这个表发生了 Insert、Update 或 Delete 操作时，数据库就会自动执行触发器所定义的 SQL 语句，从而确保对数据的处理必须符合这些 SQL 语句所定义的规则。

16. 系统启动后，不能连接数据库，可能是哪些方面的原因？

1、数据库有关的服务没有启动

2、防火墙可能阻止了数据库的端口

3、如数据库可以启动，而登录不了，可能是密码错误或连接参数配置错误

4、文件已经破坏或不存在

17. 你认为客户服务的重点是什么？

随着市场的竞争进一步加剧，服务已经成为企业核心竞争力的要素之一，服务的重点是沟通，沟通可以消除客户的误会和不满，沟通可以提高客户的感知度。因此，我认为我们客户服务管理工作就应该从做好沟通的管理开始。

自己一定要理解服务，理解服务能干什么，能做到什么，结合公司的业务能给客户提供什么服务。服务过程中是否能给客户提供优秀的服务，倾听客户的意见，持续改进服务方式。尽量在事件发生之前，避免或杜绝客户的投诉，投诉发生后，认真处理。

18. 说明静态路由和动态路由的区别？

静态路由：就是由管理员在路由器中手工设置的固定路由信息，静态路由不能对网络的改变做出反应，一般用于规模不大、拓扑结构固定的网络中，其优点是设置简单、高效，在所有路由中，静态路由优先级最高，当动态路由与静态路由发生冲突时，以静态路由为准。

动态路由：就是由网络中的路由器之间互相通信，传递路由信息，利用收到的路由信息更新路由表的过程，它能适应网络结构的变化。主要用于规模大、拓扑结构复杂的网络。

大数据和数据结构算法

1.海量日志数据，提取出某日访问阿里次数最多的那个 IP。

首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。注意到 ip 是 32 位的，最多有个 2^32 个 ip。同样可以采用映射的方法，比如模 1000，把整个大文件映射为 1000 个小文件，在找出每个小文件中出现频率最大的 ip（可以采用 hash_map 进行频率统计，然后再找出频率最大的几个）及相应的频率。然后再在这 1000 个最大的 ip 中，找出那个频率最大的 ip，即为所求。

算法思想：分而治之+Hash

1. IP 地址最多有 2^32=4G 种取值情况，所以不能完全加载到内存中处理；
2. 可以考虑采用“分而治之”的思想，按照 IP 地址的 hash(ip)%1024,把海量 IP 日志分别存储到 1024 个小文件中。这样，每个小文件最多包含 4MB 个 IP 地址；
3. 对于每个小文件，可以构建一个 ip 为 key，出现次数为 value 的 Hash map，同时记录当前出现次数最多的那个 ip 地址；
4. 可以得到 1024 个小文件中的出现次数最多的 ip，再依据常规的排序算法得到总体上出现次数最多的 ip；
5. 搜索引擎会通过日志文件把用户每次检索使用的所有的检索串都记录下来，每个查询串的长度是 1-255 字节；

2.假设目前有一千万个记录（这些查询串的重复读比较高，虽然总数是一千万，但如果出去重复后，不超过 3 百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门。）请你统计最热门的 10 个查询串，要求使用的内存不能超过 1g。

第 1 步：先对这批海量数据预处理，在 O（N）的时间内用 Hash 表完成统计。

第 2 步：借助堆这个数据结构，找出 Top K，时间复杂度为 Nlogk。

借助堆结构，我么可以在 log 量级的时间内查找和调整/移动。因此，维护一个 K（该题目中是 10）大小的小根堆，然后遍历 300 万的 query，分别和根元素进行对比所以，我们最终的时间复杂度是：o(N)+n`*o(logk),(N 为 1000 万，N‵为 300 万)。

或者，采用 trie 树，关键字域存该查询串出现的次数，没有出现为 0.最后用 10 个元素的最大堆来对出现的频率进行排序。

3.有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 字节，内存限制大小是 1M，返回频数最高的 100 个词。

顺序读文件中，对于每个词 X，取 hash(X)%5000,然后按照该值存到 5000 个小文件（记为 x0,x1,x2...x4999）中.这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过 1M。

对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树/hash_map 等），并取出出现频率最大的 100 个词（可以用含 100 个结点的最小堆），并把 100 个词及相应的频率存入文件,这样又得到了 5000 个文件,下一步就是把这 5000 个文件进行归并(类似于归并排序）的过程了。

4.有 10 个文件,每个文件 1G,每个文件的每一行存放的都是用户的 query,每个文件的 query 都可能重复。要求你按照 query 的频度排序。

还是典型的 TOP K 算法，解决方案如下：

方案 1：顺序读取 10 个文件，按照 hash（query）%10 的结果将 query 写入到另外 10 个文件（记为）中。这样新生成的文件每个的大小大约也 1G（假设 hash 函数是随机的）。

找一台内存在 2G 左右的机器，依次对用 hash_map(query,query_count)来统计每个 query 出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对应的 query_count 输出到文件中。这样得到了 10 个排序好的文件（记为）。对这 10 个文件进行归并排序（内排序与外排序相结合）。

方案 2：

一般 query 的总量是有限的，只是重复的次数比较多而已，可以对于所有的 query，一次性就可以加入到内存了。这样，我们就可以采用 trie 树/hash_map 等直接来统计每个 query 出现的次数，然后按出现次数做快速/堆/归并排序就可以了。

方案 3：

与方案 1 类似，但在昨晚 hash，分成多个文件后，可以交给多个文件来处理，采用分布式的架构来处理（比如 mapreduce），最后在进行合并。

5.给定 a、b 两个文件，各存放 50 亿个 url，每个 url 各占 64 字节，内存限制是 4G，让你找出 a、b 文件共同的 url？

方案 1：可以估计每个文件的大小为 5G*64=320G，远远大于内存限制的 4G。所以不可能将其完全加载到内存中处理，考虑采取分而治之的方法。

遍历文件 A，对每个 url 求取 hash（url）%1000,然后根据所取得的值将 url 分别存储到 1000 个小文件（记为 a0,a1,...,a999）中，这样每个小文件大约为 300M。

遍历文件 b,采取和 a 相同的方式将 url 分别存储到 1000 小文件（记为 a0vsb0,a1vsb1,...,a999vsb999）中，不对应的小文件不可能相同的 url.然后我们只要求出 1000 对小文件中相同的 url 即可。

求每队小文件中相同的 url 时，可以把其中一个小文件的 url 存储到 hash_set 中。然后遍历另一个小文件的每个 url,看其是否在刚才构建的 hash_set 中，如果是，那么就是共同的 url,存到文件里面就可以了。

方案 2：如果允许有一定的错误率，可以使用 Bloom filter,4G 内存大概可以表示 340 亿 bit。将其中一个文件中的 url 使用 bloom filter 映射为这 340 亿 bit，然后挨个读取另外一个文件的 url，检查是否与 Bloom filter,如果是，那么该 url 应该是共同的 url（注意会有一定的错误率）。

Bloom filter 日后会在本 BLOG 内详细阐述。

6.在 2.5 亿个整数中找到不重复的整数，注，内存不足以容纳这 2.5 亿个整数。

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表示多次，11 无意义）进行，共需内存 2^32*2bit=1GB 内存，还可以接受。然后扫描这 2.5 亿个整数，查看 Bitmap 中相对应位，如果是 00 变 01,01 变 10,10 保持不变。扫描完事后，查看 bitmap,把对应位是 01 的整数输出即可。

方案 2：也可以用与第 1 题类似的方法，进行划分小文件的方法，然后在小文件中找出不重复的整数，并排序，然后再进行归并，注意去除重复的元素。

7.腾讯面试题：给 40 亿个不重复的 unsigned int 的整数，没排过序的，然后再给一个整数，如何快速判断这个数是否在那 40 亿个数中？

方法 1：申请 512M 的内存，一个 bit 位代表一个 unsigned int 值。读入 40 亿个数，设置相应的 bit 位，读入要查询的数，查看相应的 bit 是否为 1，为 1 表示存在，为 0 表示不存在。

方法 2：参考思路：

又因为 2^32 为 40 亿多，所以给定一个数可能在，也可能不在其中；

这里我们把 40 亿个数中的每一个用 32 位的二进制来表示

假设这 40 亿个数开始放在一个文件中。

然后将这 40 亿个数分成二类：

1.最高位为 0

2.最高位为 1

并将这两类分别写入到两个文件中，其中一个文件中数的个数《=20 亿，而另一个》=20 亿（这相当于折半了）；

与要查找的数的最高位比较并接着进入相应的文件再查找

再然后把这个文件为又分成两类：

1.次最高位为 0

2.次最高位为 1

并将这两类分别写入到两个文件中，其中一个文件中数的个数《=10 亿，而另一个》=10 亿（这相当于折半了）

与要查找的数的次最高位比较并接着进入相应的文件再查找。

8.怎么在海量数据中找出重复次数最多的一个？

方案 1：先做 hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数，然后找出上一步求出的数据中重复次数最多的一个就是所求。

9.上千万或上亿数据（有重复），统计其中出现次数最多的前 n 个数据。

上千万或上亿的数据，现在的机器的内存应该能存下，所以考虑采用 hash_map/搜索二叉树/红黑树等来进行统计次数。然后就是去出前 N 个出现次数最多的数据了。可以使用堆机制。

10.一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，请给出思想，给出实践复杂度分析。

这题是考虑时间效率，用 trie 树统计每个词出现的次数，时间复杂度是 O（n*le）(le 表示单词的平均长度)。然后是找出出现最频繁的前 10 个词，可以用堆实现，时间复杂度是 O（n*lg10）,所以总的时间复杂度是 O（n*le）与 O(n*lg10)中较大的哪一个。

11.双层桶划分---其实本质上就是分而治之的思想，重在“分”的技巧上！

适用范围：第 K 大，中位数，不重复或重复的数字

基本原理及要点：因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步确定范围，然后最后在一个可以接受的范围内进行。可以通过多次缩小，双层只是一个例子。

扩展问题实例：

1）2.5 亿个整数中找出不重复的个数，内存空间不足以容纳这 2.5 亿个整数

有点像鸽巢原理，整数个数为2^32，也就是，我们可以将这2^32个数， 划分为2^8个区域（比如用单个文件代表一个区域），然后将数据分离到不同的区域，然后不同的区域在利用bitmap就可以解决了，也就是说只要有足够的磁盘空间，就可以方便的解决问题。

2).5亿个int找它们的中位数

这个例子比上面的那个更明显。首先我们把int划分为2^16个区域，然后读取数据统计落到各个区域中的数的个数，之后我们根据统计结果就可以判断中位数落在那个区域，同时知道这个区域的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那些数就可以了。

12.数据库索引

使用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对于海量的增删改查进行处理。

13.倒排索引（Inverted index）

使用范围：搜索引擎、关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全本搜索下某个单词在一个文档或者一组文档中的存储位置的映射。

以英文为例，下面是要被索引的文本：

T0="it is what it is"

T1="what is it"

T2="it is a banana"

我们就是得到下面的反向文件索引

"a":{2}

"banana":{2}

"is":{0,1,2}

"it":{0,1,2}

"what":{0,1}

检索的条件"what","is","it"将对应集合的交集

正向索引开发出来用来存储每个文档的单词的列表，正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在检验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

扩展问题实例：文档检索系统，查询那些文件包含了某个单词，比如常见的学术论文的关键词检索。

14.外排序

使用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择败者树原理，最有归并树

扩展问题实例：

1）.有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 个字节，内存限制大小是 1M。返回频数最高的 100 个词。

这个数据具有很明显的特点，词的大小为 16 个字节，但是内存只有 1m 做 hash 有些不够，所以可以用来排序，内存可以当输入缓冲区使用。

15.trie 树

使用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展问题实例：

1)有 10 个文件,每个文件 1G,每个文件的每一行都存放的是用户的 query,每个文件的 query 都是可能重复，要你按照 query 的频度排序

2）1000 万字符串，其中有些是相同的（重复），需要把重复的全部去掉，保留没有重复的字符串。请问怎么设计和实现？

3）寻找热门查询：查询串的重复读比较高，虽然总数是 1 千万，但如果去除重复后，不超过 3 百万个，每个不超过 255 字节。

16.分布式处理 mapreduce

使用范围:数据量大，但是数据类小可以放入内存

基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约

扩展问题实例：

1).海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10.

2).一共有 N 个机器，每个机器上有 N 个数。每个机器最多存 O（n）个数并对它们操作。如何找到 N^2 个数的中数（median）？

经典问题分析

上千万 or 亿数据（有重复），统计其中出现次数最多的前 N 个数据，分两种情况：可一次读入内存，不可一次读入。

可用思路：trie 树+堆，数据库索引，划分子集分别统计，hash,分布式计算，近似统计，外排序

所谓的是否能一次读入内存，实际上应该指取出重复的数据量。如果取出重复的数据量可以放入内存，我们可以为数据创建字典，比如通过 map,hashmap,trie,然后直接进行统计即可。当然在更新每天数据出现次数的时候，我们可以利用一个堆来维护出现次数最多的前 N 个数据，当然这样导致维护次数增加，不如完全统计后再求前 N 大效率高。

如果数据无法放入内存，一方面我们可以考虑上面的字典方法能否被改进以适应这种情形，可以做的改变就是把字典存放到硬盘上，而不是内存，这可以参考数据库的存储方法。

当然还有更好的方法，就是可以采用分布式计算，基本上就是 map-reduce 过程，首先可以根据数据值或者把数据 hash（md5）后的值，将数据按照范围划分到不同的机子，最好可以让数据划分后可以一次读入内存，这样不同的机子负责处理各种的数值范围，实际就是 map。得到结果后，各个机子只需拿出各自的出现次数最多的前 N 个数据，然后汇总，选出所有的数据中出现次数最多的前 N 个数据，这实际就是 reduce 过程。

实际上可能想直接将数据均分到不同的机子上进行处理，这样是无法得到正确的解的。因为一个数据可能被均分到不同的机子上，而另一个则可能 完全聚集到一个机子上，同时还可能存在具有相同数目的数据。比如，我们要找出次数最多的前 100 个，我们将 1000 万的数据分布到 10 台机器上，找到每台出现次数最多的前 100 个，归并之后这样不能保证找到真正的第 100 个，因为比如出现次数最多的第 100 个可能有 1 万个，但是它被分到了 10 台机器上，这样在每台上只有 1 千个，假设这些机子排名在 1000 个之前的那些都是单独分布在一台机器上的，比如有 1001 个，这样本来具 1 万个的这个就会被淘汰，即使我们让每台机器选出出现次数最多的 1000 个再归并，仍然会出错，因为可能存在大量个数为 1001 个的发生聚集。因此不能将数据随便均分到不同机器上，而是要根据 hash 后的值将它们映射到不同的机子上处理，让不同的机器处理一个数值范围。

16、假设淘宝一天有 5 亿条成交数据，求出销量最高的 100 个商品并给出算法的时间复杂度。

先用哈希，统计每个商品的成交次数，然后再用在 N 个数中找出前 K 大个数的方法找出成交次数最多的前 100 个商品。
优化方法：可以把 5 亿个数据分组存放，比如放在 5000 个文件中。这样就可以分别在每个文件的 10^6 个数据中，用哈希+堆统计每个区域内前 100 个频率最高的商品，最后求出所有记录中出现频率最高的前 100 个商品。

17、有 10 亿个杂乱无章的数，怎样最快地求出其中前 1000 大的数。

方法 1：
建一个 1000 个数的最小堆，然后依次添加剩余元素，如果大于堆顶的数（堆中最小的），将这个数替换堆顶，并调整结构使之仍然是一个最小堆，这样，遍历完后，堆中的 1000 个数就是所需的最大的 1000 个。算法的时间复杂度为 O（nlogk）=n*log1000=10n（n 为 10 亿，k 为 1000）。

优化的方法：分治法。可以把所有 10 亿个数据分组存放，比如分别放在 1000 个文件中。这样处理就可以分别在每个文件的 10^6 个数据中找出最大的 10000 个数，合并到一起再找出最终的结果。

优化的方法：如果这 10 亿个数里面有很多重复的数，先通过 Hash 法，把这 10 亿个数字去重复，这样如果重复率很高的话，会减少很大的内存用量，从而缩小运算空间，然后通过分治法或最小堆法查找最大的 1000 个数。

方法 2：

1.用每一个 BIT 标识一个整数的存在与否，这样一个字节可以标识 8 个整数的存在与否，对于所有 32 位的整数，需要 512Mb，所以开辟一个 512Mb 的字符数组 A，初始全 0

2.依次读取每个数 n，对于数 n，n 存放在 A[N>>3]中的某个 bit，因此，将 A[n>>3]设置为 **A[n>>3]|(1<<(n%8))**(这里是 1 左移几位),相当于将每个数的对应位设置为 1

3.在 A 中，从数组尾开始向前遍历，从大到小读取 1000 个值为 1 的数，就是最大的 1000 个数了。

这样读文件就只需要 1 遍，在不考虑内存开销的情况下，应该是速度最快的方法了。

20、给一列无序数组，求出中位数并给出算法的时间复杂度。

若数组有奇数个元素，中位数是 a[(n-1)/2]；若数组有偶数个元素，中位数为 a[n/2-1]和 a[n/2]两个数的平均值。这里为方便起见，假设数组为奇数个元素。

思路一：把无序数组排好序，取出中间的元素。时间复杂度取决于排序算法，最快是快速排序，O(nlogn)，或者是非比较的基数排序，时间为 O(n),空间为 O(n)。这明显不是我们想要的。

思路二：采用快速排序的分治 partition 过程。任意挑一个元素，以该元素为支点，将数组分成两部分，左边是小于等于支点的，右边是大于支点的。如果左侧长度正好是(n－1)／2，那么支点恰为中位数。如果左侧长度<(n-1)/2, 那么中位数在右侧，反之，中位数在左侧。 进

```
//快速排序的分治过程找无序数组的中位数
int partition(int a[], int low, int high) //快排的一次排序过程
{
        int q = a[low];
        while (low < high)
        {
                while (low < high && a[high] >= q)
                        high--;
                a[low] = a[high];
                while (low < high && a[low] <= q)
                        low++;
                a[high] = a[low];
        }
        a[low] = q;
        return low;
}
```

```
int findMidium(int a[], int n)
{
        int index = n / 2;
        int left = 0;
        int right = n - 1;
        int q = -1;
        while (index != q)
        {
                q = partition(a, left, right);
                if (q < index)
                        left = q + 1;
                else if (q>index)
                        right = q - 1;
        }
        return a[index];
}
```

思路三：将数组的前（n+1）/ 2 个元素建立一个最小堆。然后，对于下一个元素，和堆顶的元素比较，如果小于等于，丢弃之，如果大于，则用该元素取代堆顶，再调整堆，接着看下一个元素。重复这个步骤，直到数组为空。当数组都遍历完了，（堆中元素为最大的（n+1）/ 2 个元素，）堆顶的元素即是中位数。

```
//构建最小堆找无序数组的中位数
void nswap(int& i, int& j)
{
        i = i^j;
        j = i^j;
        i = i^j;
}
void minHeapify(int a[], int i, int len)
{
        int temp;
        int least = i;
        int l = i * 2 + 1;
        int r = i * 2 + 2;
        if (l < len && a[l] < a[least])
                least = l;
        if (r < len && a[r] < a[least])
                least = r;
        if (least != i)
        {
                nswap(a[i], a[least]);
                minHeapify(a, least, len);
        }
}
```

```
void buildMinHeap(int a[], int len)
{
        for (int i = (len-2) / 2; i >= 0; i--)
        {
                minHeapify(a, i, len);
        }
}
int findMidium2(int a[], int n)
{
        buildMinHeap(a, (n + 1) / 2);
        for (int i = (n + 1) / 2; i < n; i++)
        {
                if (a[i] > a[0])
                {
                        nswap(a[i], a[0]);
                        minHeapify(a, 0,(n + 1) / 2);
                }
        }
        return a[0];
}
```

引申一：
查找 N 个元素中的第 K 个小的元素

该方案改编自快速排序。经过快排的一次划分
1）如果左半部份的长度>K-1，那么这个元素就肯定在左半部份了
2）如果左半部份的长度==K-1，那么当前划分元素就是结果了。
3）如果。。。。。。<K-1,那么这个元素就肯定在右半部分了。
并且，该方法可以用尾递归实现。效率更高。

也可以用来查找 N 个元素中的前 K 个小的元素，前 K 个大的元素。。。。等等。
引申二：
查找 N 个元素中的第 K 个小的元素，假设内存受限，仅能容下 K/4 个元素。
分趟查找，
第一趟，用堆方法查找最小的 K/4 个小的元素，同时记录剩下的 N-K/4 个元素到外部文件。
第二趟，用堆方法从第一趟筛选出的 N-K/4 个元素中查找 K/4 个小的元素，同时记录剩下的 N-K/2 个元素到外部文件。
。。。
第四趟，用堆方法从第一趟筛选出的 N-K/3 个元素中查找 K/4 个小的元素，这是的第 K/4 小的元素即使所求。

21、输入一个整型数组，求出子数组和的最大值，并给出算法的时间复杂度。

设 b[i]表示 a[0...i]的子数组和的最大值，且 b[i]一定包含 a[i]，即：

sum 为子问题的最优解，

1. 包含 a[i],即求 b[i]的最大值,在计算 b[i]时,可以考虑以下两种情况,因为 a[i]要求一定包含在内，所以

1) 当 b[i-1]>0, b[i] = b[i-1]+a[i]

2) 当 b[i-1]<=0, b[i] = a[i], 当 b[i-1]<=0，这时候以 a[i]重新作为 b[i]的起点。

2. 不包含 a[i],即 a[0]~a[i-1]的最大值（即 0~i-1 局部问题的最优解),设为 sum

最后比较 b[i]和 sum，即,如果 b[i] >sum ,即 b[i]为最优解,然后更新 sum 的值.

在实现时，bMax 代表 b[k], sum 更新前代表前一步子问题的最优解，更新后代表当前问题的最优解。实现如下：

```cpp
//求数组的子数组和的最大值，时间复杂度为O(n)
int maxSumArr(int a[], int n,int* start, int* end)
{
	int s, e;
	int sum = a[0];
	int bMax=a[0];
	*start = *end = 0;
	for (int i = 1; i < n; i++)
	{
		if (bMax > 0) //情况一，子数组包含a[i]，且b[i-1]>0（上一次的最优解大于0），b[i] = b[i-1]+
		{
			bMax += a[i];
			e = i;
		}
		else     //情况二，子数组包含a[i]，且b[i-1]<=0（上一次的最优解小于0），这时候以a[i]重新作为
		{
			bMax = a[i];
			s = i;
			e = i;
		}            //情况三，子数组不包含a[i],即b[i]=sum
		if (bMax > sum)   //三种情况相比较，最大值作为更新后的最优解，存在sum
		{
			sum = bMax;
			*start = s;
			*end = e;
		}
	}
	return sum;
}
```

引申：求子数组和的最小值，同理。

```
int minSumArr(int a[], int n, int* start, int* end)
{
        int s, e;
        int bMin = a[0];
        int sum = a[0];
        *start = *end = 0;

        for (int i = 0; i < n; i++)
        {
                if (bMin < 0) //情况一，子数组包含a[i]，且b[i-1]<0，b[i] = b[i-1]+a[i]
                {
                        bMin += a[i];
                        e = i;
                }
                else  //情况二，子数组包含a[i]，且b[i-1] > 0，这时候以a[i]重新作为b[i]的起点
                {
                        bMin = a[i];
                        s = e = i;
                }      //情况三，子数组不包含a[i],即b[i]=sum
                if (bMin < sum)  //三种情况相比较，最小值作为更新后的最优解，存在sum
                {
                        sum = bMin;
                        *start = s;
                        *end = e;
                }
        }
        return sum;
}
```

22、给出 10W 条人和人之间的朋友关系，求出这些朋友关系中有多少个朋友圈（如 A-B、B-C、D-E、E-F，这 4 对关系中存在两个朋友圈），并给出算法的时间复杂度。

解题思路：并查集，而题目只需要求出朋友圈数量，并不需要求出各朋友圈，所以该并查集的实现也可以非常简单。

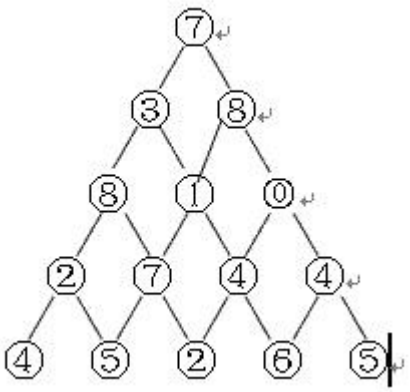
A-B,就把 father[B] = A，处理每条朋友关系即可得到结果。

而关于并查集的介绍，已有很多博文有所阐述，这里就不啰嗦了。

```c
//朋友圈-并查集
int set[10001];
int find(int x)
{
        int i, j, r;
        r = x;
        while (set[r] != r) //寻找此集合的代表
                r = set[r];
        i = x;
        while (i != r) //使得r代表的集合中，所有结点直接指向r，即路径压缩
        {
                j = set[i];
                set[i] = r;
                i = j;
        }
        return r;
}
void merge(int x, int y)
{
        int t = find(x);
        int h = find(y);
        if (t < h)
                set[h] = t;
        else
                set[t] = h;
}
```

```c
int friends(int n, int m, int (*r)[2])  //n个人，m对好友关系，存放在二维数组r[m][2]中
{
        int i, count;
        for (i = 1; i <= n; i++)
                set[i] = i;
        for (i = 0; i < m; i++)
                merge(r[i][0], r[i][1]);
        count = 0;
        for (i = 1; i <= n; i++)
        {
                if (set[i] == i)
                        count++;
        }
        return count;
}
```

23、如图所示的数字三角形，从顶部出发，在每一结点可以选择向左走或得向右走，一直走到底层，要求找出一条路径，使路径上的值的和最大。给出算法的时间复杂度。



定义状态为：dp[i][j]表示，从第 i 行第 j 个数字到最后一行的某个数字的权值最大的和。
那么我们最后只需要输出 dp[1][1]就是答案了.
状态转移方程为：dp[i][j] += max(dp[i+1][j+1],dp[i+1][j]);好了， 从第 n-1 行往上面倒退就好了。