网易2021校招笔试-前端开发工程师（正式...

1、请说明数据库事务的特性，并描述脏读、不可重复读和幻读的现象

> **参考答案1**
> 事务特性有：原子性、一致性、持久性、隔离性
>
> 脏读：就是没有提交的数据，举个例子：比如某个事务在对数据库中某条记录进行修改，修改后没有提交也没有回滚，也就是让其处于一个待定的状态，这个时候如果有其他的事务来先一步对这条记录进行读取或者处理了的现象。
>
> 不可重复读取：一个事务先后读取某条记录，但在两次读取之间该条记录杯其他另一个事务给修改了，就造成了两次读取的数据不同的现象。
>
> 幻读：幻读就是一个事务按照查询条件查询以前检索过的数据可是发现该数据被其他事务插入了满足其查询条件的新数据的现象。
>
> 不可重复读喝脏读的区别是一个是读取了前一事务提交的数据，而一个是读取了另一个事务未提交的数据

> **参考答案2**
>
> 数据库事务特性：
>
> A atomicity 原子性 指的是事务包含的所有操作，要么全部成功，要么全部回滚失败
>
> C consistency 一致性 指的是事务必须使数据库从一个一致性状态到另一个一致性状态，即事务执行前后全部处于一致性状态
>
> I isolation 隔离性 指的是多个并发事物间互相隔离
>
> D durability 持久性 指的是事务一旦提交了，那么对数据库中数据的修改就是永久性的，即便在数据库系统遇到故障的情况下也不会丢失提交事务的操作
>
> 脏读：指的是 一个事物读取了另一个事务还没有提交的事务 。事务隔离级别：READ_COMMITED
>
> 不可重复读：指的是同一事务内，两次相同的查询返回了不同的结果 。事务隔离级别：REPEATABLE_READ

> 幻读：一个事务进行读取，分别读取到了不同的数据，与不可重复读不同的是，幻读针对的是数据条数的变化。 事务隔离级别：SERIALIZABLE_READ

2、请描述下Http2.0的特性

> **参考答案**
>
> 1.二进制分帧, 在应用层和传输层之间, 使其更接近1/0的传输, 提高了传输性能
>
> 2.头部压缩: 对于头部某些经常传递的字段会在浏览器和服务器端各自cache了一份header fields表, 比如cookie很长就可以通过该手段减少每次传递的cookie
>
> 3.多路复用: 实现了真正的复用, http1.1是长连接, 而http2.0中是并行的多个连接。能够在一个TCP上进行任意数量HTTP请求
>
> 4.服务器推送: 举个例子，比如浏览器像服务器请求html资源，服务器首先会向其发送html资源。然后不会等浏览器请求css和js资源，而是在发送完html后主动再向浏览器发送css和js资源

> **参考答案2**
>
> 1.首部压缩，HTTP/2为首部压缩设计了HPACK算法。
>
> 2.二进制分帧：在应用层HTTP/2和传输层TCP/UDP之间加了一个二进制分帧层，在二进制分帧层中，HTTP/2会将所有传输的信息分割为更小的消息和帧，并对他们进行二进制格式的编码。其中首部信息会被封装到HEADER frame，请求体部分会被封装到DATA frame中
>
> 3.多路复用:允许同时通过单一的HTTP/2连接发起多重的请求–响应消息，主要解决浏览器请求同一域名有一定数量限制的问题。HTTP/2可以很容易的去实现多流并行而不用依赖建立多个TCP连接，HTTP/2把HTTP协议通信的基本单位缩小到一个一个的帧，这些帧对应着逻辑流中的信息，并行地在同一个TCP连接上双向交换信息。

3、给定两个字符串 s和 t ，判断 s是否为 t 的子序列。

你可以认为 s 和 t 中仅包含英文小写字母。字符串 t 可能会很长（长度 ~= 500,000），而 s 是个短字符串（长度 <=100）。

字符串的一个子序列是原始字符串删除一些（也可以不删除）字符而不改变剩余字符相对位置形成的新字符串。（例如，”ace”是”abcde”的一个子序列，而”aec”不是）。

> 参考答案1
>
> `
>
> ```
> var boolen = function(str1,str2) {
>   if (str1.length > str2.length) {
> ```

```
    return false
  }
  var first = str1.split('')[0]
  for (let a = 0; a <= str2.length; a++) {
    if (str2[a] === first) {
      if (str1.length > 0) {
        var strNew1 = str1.slice(1)
        var strNew2 = str2.slice(a)
        return boolen(strNew1,strNew2)
      } else {
        return true
      }
    }
  }
  return false
}
```

参考答案2

```
let s = readline();
let t = readline();
function a(s,t){
    let count = 0;
    for(var i = 0;i < s.length;i++){
        for(var j = 0;j < t.length;j++){
            if(si){
                count++;
                i++;
            }
        }
    }
    return count >= s.length;
}
print(a(s,t));
```

参考答案3

```
function decide(str1, str2) {
  str2 = str2.split('')
  str1 = str1.split('')
  if (str2.length < str1.length) return false;
  let num = str1.length
```

```
  let first = str1.shift()
  let counter = 0
  for (let i = 0; i < str2.length; i++) {
    if (first === str2[i]) {
      counter++;
      first = str1.shift()

    }
  }
  return num === counter
}
`
```

4、疫情逐步缓和后，电影院终于开业了，但是由于当前仍处于疫情期间，应尽量保持人群不聚集的原则。

所以当小易来电影院选定一排后，尽量需要选择一个远离人群的位置。
已知由0和1组成的数组表示当前排的座位情况,其中1表示已被选座，0表示空座
请问小易所选座位和最近人的距离座位数最大是多少?
有如下假设：至少有一个人已选座，至少有一个空座位，且座位数限制为

$$2 \leq length \leq 1000$$

参考答案

假设数组有n个元素，则有两种情况：

第一种是买边上的座位，如果边上的座位都是空的，买最左边或者最右边的座位，那最大的距离就是索引0和n−1分别与离自己最近的1的距离的最大值。

第二种是买中间的座位，即从第一个1到最后一个1之间的某个座位，这时候需要寻找两个隔得最远的1，这时候这两个1距离的一半就是中间位置的最大距离。

综上两种情况，选择更大的那个距离，就是我们要求的最大距离

参考答案2

```
`
let _str=readline();
```

```
_str = _str.replace(/ /g, '');
let _arr = _str.split('1');
_arr = _arr.map((v, i) => {
  if (i === 0 || i === _arr.length - 1) {
    return v.length;
  }
  return Math.ceil(v.length / 2);
})
console.log(Math.max.apply(null, _arr));
`
```

5、和谐连续序列是指一个连续序列中元素的最大值和最小值之间的差值正好是1。现在，给定一个整数数组，你需要在所有可能的连续子序列中找到最长的和谐连续子序列的长度。

参考答案1

```
`
//滑动窗口
let arr = readline();
arr = arr.split(' ');
let maxLen=0, queue=;
arr.forEach(val => {
    queueval ? queue[val]+1 : 1;
    curr.push(val);

    let max=val, min=null;
    queue.forEach((v, index) => {
        if (min==null) min = index;
        if (max < index) max = index;
    })

    while (max - min > 1 && curr.length>1) {
        let queueVal = curr.shift();
        if (queuequeueVal;
        else queue[queueVal]--;

       max=val, min=null;
        queue.forEach((v, index) => {
            if (min==null) min = index;
            if (max < index) max = index;
        })
    }
    if (maxLen < curr.length && max-min==1) {
        maxLen = curr.length;
    }
})
```

```
console.log(maxLen);
`
```

参考答案2

1.穷举所有的连续子序列

2.用线性复杂度求取各个连续子序列的最大最小值

3.遇到和谐序列时更新和谐序列的最大长度