Problem I Met

大家的评价：

> 多学习多实践
>
> 实现方案要有长远计划
>
> 说话不要太直接

初始化很重要

以前做PTA的时候，一个算法要被运行好多次，所以每次开始时都需要把之前那一次的变量初始化。在leetcode做题就完全没有问题。

直到我遇到个需求，有个方法initProgress，一次是进入这个页面，一次是拿到服务端消息，然而我在第一次储存的list被第二次接着用了，所以导致结果不对。

comparator崩溃之谜

mood有一个排序，自己的在最前面，之后的按照未读数量和更新时间排序。

```
private var  isShowShareView = false
private var lastMood:IMMood? = null

companion object {
    val moodComparator: Comparator<IMMoodConversation> = Comparator { o1, o
        //自己的排在最前面
        if (o1.isSelf) return@Comparator -1
        if (o2.isSelf) return@Comparator 1
        //未读排在前面
        if (o1.unreadCountByTime > 0 && o2.unreadCountByTime <= 0) return@
        if (o2.unreadCountByTime > 0 && o1.unreadCountByTime <= 0) return@
        //其余按时间排序
        return@Comparator (o2.updatedTime - o1.updatedTime).toInt()
    }
}
```

```
private var  isShowShareView = false
private var lastMood:IMMood? = null

companion object {
    val moodComparator: Comparator<IMMoodConversation> = Comparator { o1, o2
        //自己的排在最前面
        if (o1.isSelf && !o2.isSelf) return@Comparator -1
        if (o2.isSelf && !o1.isSelf) return@Comparator 1
        //未读排在前面
        if (o1.unreadCountByTime > 0 && o2.unreadCountByTime <= 0) return@Co
        if (o2.unreadCountByTime > 0 && o1.unreadCountByTime <= 0) return@Co
        //其余按时间排序
        return@Comparator (o2.updatedTime - o1.updatedTime).toInt()
    }
}
```

这段代码我看了下，应该是第一个版本实现有问题，后来改过后应该没有问题了

左边的有问题，右边的没有问题

sgn(compare(x, y)) == -sgn(compare(y, x))
((compare(x, y)>0) && (compare(y, z)>0)) implies compare(x, z)>0
compare(x, y)==0 implies that sgn(compare(x, z))==sgn(compare(y, z)) for all z

要符合这个原则

它已经满足了自反性 传递性 对称性三个原则，为什么还会报TimSort崩溃呢，因为线上大使有非常多的mood，排序的时候，正好有个回调在子线程改了updatedTime，然后比较就错误了。

务必好好写注释

三个月前写的代码看不懂了，没有注释，commit写了一句无关痛痒的洋文。

所以业务相关代码一定要注释（不仅仅翻译变量的英文意思，写清楚为什么要这么做，比如为了解决什么问题）

Touch事件冲突

之前重写了个mood键盘单个emoji点击和一直按住的自定义事件，发现和键盘滚动冲突了，采用了内部拦截parent.requestDisallowInterceptTouchEvent(true)解决。

这期有个需求是viewpager里仅有一个fragment，左划右划有不同的退出效果，OnPageChange-Callback没有能做到这件事的回调，viewpager添加onTouchListener是没有用的，估计是内部的recyclerView消化了事件。

最终决定FrameLayout嵌套viewpager，让FL截获左右滑的事件（其他事件比如上下滑都得传给vp），问题是onInterceptTouchEvent是否返回true很难判断，不像这篇文章。解决方案是改写dispatchTouch-Event，索性不拦截事件，内部判断一下，提前把左右滑的事情做了。

不要相信服务端

有个需求是服务端传过来一个emoji，我根据emoji找drawable的id，然后服务端传过来一个“\n<0x04>😊”，我没找到drawable，crash了。所以要时刻怀疑服务端传过来的数据的可靠

性。

为什么SingleUserFragment耦合严重

2021.4.6 今天起倒叙记录

Mood浏览架构是很不合理的，缺少了Presenter层，每次需求都往SingleUserFragment里面塞，经过2.3、2.4、2.5三个迭代，这个fragment已经有了一千六百行。原因是，第一版它只是作为一个viewpager中的一个播放模块，里面塞个MoodLayout就能用了，之后的需求owner往里面塞代码的时候也因为代码量不大，直接写进了fragment里面。所以我mentor的评价（above）还是很正确的😭。

这一期quick reaction赶紧悬崖勒马，写了个Presenter。以后的需求这个fragment就当IoC容器，传几个View，其他的逻辑都在presenter里处理。

顺便看表情键盘的data manager，解耦得很优雅，接口+代理。

Kotlin

```kotlin
object ExpressionDataManager :
    IEmojiExpressionDataManager by emojiExpressionDataManger,
    IHotExpressionDataManager by hotExpressionDataManager,
    IAudioExpressionDataManager by audioExpressionDataManager,
    IFavorExpressionDataManager by favorExpressionDataManager
```

写代码除了把逻辑考虑全，更要注重SOLID原则。

SharedPreference ANR Modification

Reference:

https://mp.weixin.qq.com/s/IFgXvPdiEYDs5cDriApkxQ

https://phantomvk.github.io/2018/11/05/QueuedWork/

Why apply() would cause ANR:

File writing just at the time of a lifecycle (service stop/startcommand，activity pause/stop, sleeping) because of the following strategy.

Code analysis:

Java

```
apply() {
    ...
    SharedPreferencesImpl.this.enqueueDiskWrite(mcr, postWriteRunnable);
    ...
}

enqueueDiskWrite(...) {
    ...
    final Runnable writeToDiskRunnable = new Runnable() {
        public void run() {
            // there is a lock
            synchronized (mWritingToDiskLock) {
                writeToFile(mcr);
        }
    ...
    };
    ...
    // if is apply() rather than commit()
    QueuedWork.singleThreadExecutor().execute(writeToDiskRunnable);
}
```

And *QueuedWork.waitToFinish()* is called in the lifecycle mentioned above.

Java

```
#ActivityThread
@Override
public void handleStopActivity(IBinder token, boolean show, int
configChanges,
        PendingTransactionActions pendingActions, boolean
finalStateRequest, String reason) {
    ...
    // Make sure any pending writing is now committed.
    if (!r.isPreHoneycomb()) {
        QueuedWork.waitToFinish();
    }
    ....
}
```

Though the process of file writing isn't running in the main thread, there is a considerable possibility that it will block at the same time of a lifecycle period. What's more, it is serial due to the lock: `mWritingToDiskLock`, thus causing an ANR.

算了我用中文了

QueuedWork里面的runnable是串行的，这个runnable是调用apply()的时候带上的，runnable内容如下：

Java

```java
Runnable postWriteRunnable = new Runnable() {
        @Override
        public void run() {
            awaitCommit.run(); // awaitCommit也是个runnable:
                               //mcr.writtenToDiskLatch.await();
            QueuedWork.removeFinisher(awaitCommit);
        }
    };
```

所以流程是sp写入文件 -> 释放 `mWritingToDiskLock` -> await执行 -> 移出QueuedWork队列

Queue本身就是个串行锁收集器

How to fix:

1. Find a balance between the number of SP files and the number of values in the SP file, which means we should reduce the number of SP files as well as make sure reading/writing speed is fast enough.
2. Clear work queue, like reference 1. This Code in RA written by zhaofenghui：

Java

```java
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.O) {
    SpBlockHelper.tryHackActivityThreadH()
}
```

Kotlin

```kotlin
public class SpBlockHelper {
    static final String TAG = "SpBlockHelper";
    static boolean init = false;
    static String CLASS_QUEUED_WORK = "android.app.QueuedWork";
    static String FIELD_PENDING_FINISHERS = "sPendingWorkFinishers";
    static ConcurrentLinkedQueue<Runnable> sPendingWorkFinishers = null;

    public static void beforeSPBlock(String tag) {
        if (!init) {
            getPendingWorkFinishers();
            init = true;
        }
```

```
13          RALog.d(TAG, "beforeSPBlock " + tag);
14          RALog.d(TAG, "sPendingWorkFinishers  " + sPendingWorkFinishers);
15          if (sPendingWorkFinishers != null) {
16              sPendingWorkFinishers.clear();
17          }
18      }
19
20      static void getPendingWorkFinishers() {
21          try {
22              Class clazz = Class.forName(CLASS_QUEUED_WORK);
23              Field field =
    clazz.getDeclaredField(FIELD_PENDING_FINISHERS);
24              if (field != null) {
25                  field.setAccessible(true);
26                  sPendingWorkFinishers =
    (ConcurrentLinkedQueue<Runnable>) field.get(null);
27              }
28          } catch (Exception e) {
29              RALog.e(TAG, "getPendingWorkFinishers", e);
30          }
31      }
32
33      public static void tryHackActivityThreadH() {
34          try {
35              Reflect activityThreadRef =
    Reflect.on(Class.forName("android.app.ActivityThread")).
36                      call("currentActivityThread");
37              if (activityThreadRef != null) {
38                  Handler h = activityThreadRef.field("mH",
    Class.forName("android.app.ActivityThread$H")).get();
39                  if (h != null) {
40                      Reflect hRef = Reflect.on(h);
41                      Handler.Callback hCallBack = hRef.field("mCallback",
    Handler.Callback.class).get();
42                      ActivityThreadHCallBack activityThreadHCallBack =
    new ActivityThreadHCallBack();
43                      hRef.set("mCallback", activityThreadHCallBack);
44                  }
45              }
46          } catch (Throwable t) {
47              t.printStackTrace();
48              // ignore
49          }
50
51      }
52  }
```

Kotlin

```
public class ActivityThreadHCallBack implements Handler.Callback {
    private static final int SERVICE_ARGS = 115;
    private static final int STOP_SERVICE = 116;
    private static final int SLEEPING = 137;
    private static final int STOP_ACTIVITY_SHOW = 103;
    private static final int STOP_ACTIVITY_HIDE = 104;
    private static final int ENTER_ANIMATION_COMPLETE = 149;

    @Override
    public boolean handleMessage(Message msg) {
        final int what = msg.what;
        switch (what) {
            case SERVICE_ARGS:
                SpBlockHelper.beforeSPBlock("SERVICE_ARGS");
                break;
            case STOP_SERVICE:
                SpBlockHelper.beforeSPBlock("STOP_SERVICE");
                break;
            case SLEEPING:
                SpBlockHelper.beforeSPBlock("SLEEPING");
                break;
            case STOP_ACTIVITY_SHOW:
            case STOP_ACTIVITY_HIDE:
                SpBlockHelper.beforeSPBlock("STOP_ACTIVITY");
                break;
            case ENTER_ANIMATION_COMPLETE:
                SpBlockHelper.beforeSPBlock("ENTER_ANIMATION_COMPLETE");
                break;
        }
        return false;
    }
}
```

Unicode combination

Reference https://www.cnblogs.com/crazylqy/p/10184291.html

The capacity of unicode character set is `2^21` currently.

Therefore, using utf-16 we can represent a character with 2 bytes or 4 bytes. Thus some character's length will be more than 1: "𡃁".length = 2, "😧🟫". length = 4. Use following way to cut:

Kotlin

```kotlin
val end = prefix.offsetByCodePoints(0, prefix.codePointCount(0, 10))
prefix = prefix.substring(0, end)
```

VideoLayout多次设置视频拉伸/透明问题

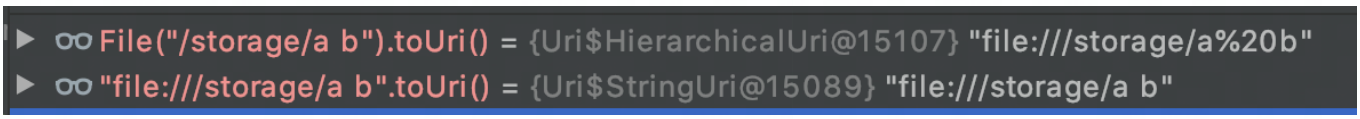
videolayout里面的video播放用的是surfaceView，mood浏览有个需求需要更换播放的video，在两个视频切换的间隙，video背景会透明（看到后面的activity）,原因是browseActivity是透明的。

1、用来描述SurfaceView的Layer或者LayerBuffer的Z轴位置是小于用来其宿主Activity窗口的Layer的Z轴位置的，但是前者会在后者的上面挖一个“洞”出来，以便它的UI可以对用户可见。实际上，**SurfaceView在其宿主Activity窗口上所挖的“洞”只不过是在其宿主Activity窗口上设置了一块透明区域。**

一个有趣的解决思路是把surfaceView的背景设置为黑色，再surfaceView.setZOrderOnTop(true)，但是遮挡了其他view。最后是用两个videoView，第一个在播放了再remove另一个。

Uri的转义

做图片裁剪的时候需要一个map储存裁剪前后图片的Uri，结果存了之后用原来的Uri作为key拿不到value。发现问题出在这儿：



调用 File.toUri(): Uri = Uri.fromFile(this) 得到的是HierarchicalUri的对象，而String.toUri()得到的是StringUri对象(HierarchicalUri和StringUri都是Uri的子类)，所以俩key不同。

而且这两个对象的toString()方法得到的结果是不同的（一个保留了空格的转义结果"%20"，一个空格转回了空格）发现HierarchicalUri的toString()方法最终调用了encode：

Java

```java
/**
 * Encodes characters in the given string as '%'-escaped octets
 * using the UTF-8 scheme. Leaves letters ("A-Z", "a-z"), numbers
 * ("0-9"), and unreserved characters ("_-!.~'()*") intact. Encodes
 * all other characters with the exception of those specified in the
 * allow argument.
 *
 * @param s string to encode
 * @param allow set of additional characters to allow in the encoded form,
 *  null if no characters should be skipped
 * @return an encoded version of s suitable for use as a URI component,
 *  or null if s is null
 */
public static String encode(String s, String allow) {
...
String toEncode = s.substring(current, nextAllowed);
        try {
byte[] bytes = toEncode.getBytes(DEFAULT_ENCODING);
            int bytesLength = bytes.length;
            for (int i = 0; i < bytesLength; i++) {
            // 看这儿
                encoded.append('%');
                encoded.append(HEX_DIGITS[(bytes[i] & 0xf0) >> 4]);
                encoded.append(HEX_DIGITS[bytes[i] & 0xf]);
}
        } catch (UnsupportedEncodingException e) {
throw new AssertionError(e);
}
...
// Encoded could still be null at this point if s is empty.
return encoded == null ? s : encoded.toString();
}
```

为什么要encode呢，猜测HierarchicalUri不需要与String互相转化，转义能保留其完整性

最终的解决方案是直接用string转uri 😁

表情键盘删除按钮消失了？？

中午接到一个bug说表情键盘的删除按钮消失了，debug的时候发现每次setVisibility都是生效的。

问了mentor，会不会设置的不是同一个对象，一看id，果然不是同一个Orz，原来是新功能在原有表情面板上加了一层，所以有俩面板了，每次设置都对另一个对象在用。

所以一定要看看对象的id！！！！！！！一下午时间啊啊啊啊啊

View.post为什么没有用

遇到个切换下一个动态、前一个视频动态无法暂停的bug，发现videoLayout暂停调用View的post方法没有被执行：

```
post {
    mVideoController?.pause()
}
```

查看post方法，发现attachInfo != null的时候runnable进入主线程的handler是必被执行的，说明是attachInfo == null时出的问题，看到“Assume that the runnable will be successfully placed after attach.”这个注释，猜测是切换动态（viewpager切下一个）时view被detach了。

```
public boolean post(Runnable action) {
    final AttachInfo attachInfo = mAttachInfo;
    if (attachInfo != null) {
        return attachInfo.mHandler.post(action);
    }

    // Postpone the runnable until we know on which thread it needs to run.
    // Assume that the runnable will be successfully placed after attach.
    getRunQueue().post(action);
    return true;
}
```

换成我们自己的Post就解决这个bug了。

```
PostUtil.post {
    mVideoController?.pause()
}
```

PS. RunQueue是ThreadLocal，它的调用时机是performTraversals( ) {

...

getRunQueue().executeActions(mAttachInfo.mHandler,0)

...

}

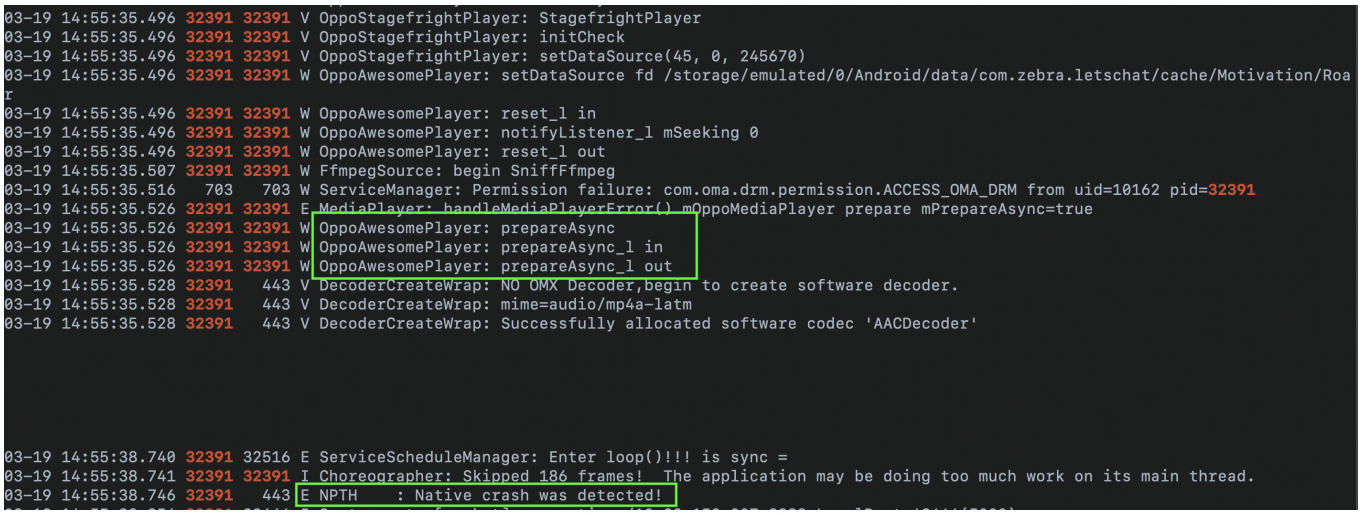
所以这个View得等下一次在页面布局才能调用post。

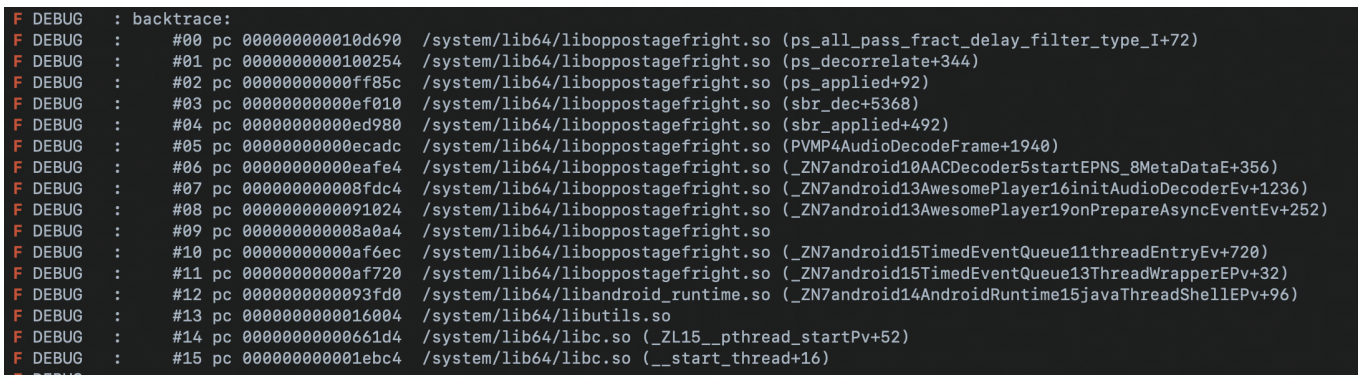
mood播放crash问题追踪

遇到一个crash，切换到一条特定mood之后app闪退重启。

一跑到断点停不到一秒就crash，Android Studio的log也自动消失。

用terminal的adb logcat | grep {进程号}或adb logcat | grep " F " （`V`：详细（最低优先级）`D`：调试 `I`：信息 `W`：警告`E`：错误 `F`：严重错误 `S`：静默（最高优先级，绝不会输出任何内容））可以保持log不消失。特别注意E和F。





发现native crash，有个和媒体播放器有关的段错误。（stagefright.so is the name given to a group of software bugs that affect versions 2.2 "Froyo" of the Android operating system. The name is taken from the affected library, which among other things, is used to unpack MMS

（Multimedia Messaging Service）messages. Exploitation of the bug allows an attacker to perform arbitrary operations on the victim's device through remote code execution and privilege escalation.）

之后测试这个mood对应的music，发现没什么问题。

仔细看log，prepare后立刻crash了，意识到这个mood的music还没被set，是前一个music切换出的问题，猜测是音频编码错误。

解决：让服务端把前一个music换成正确编码的music，再升级一下端上的settings就可以了。

思考：发现PM提供的东西（比如emoji列表的json中英文混淆）总能出点问题。差点就把这个问题归结为Android 6的系统bug QAQ

fragment需要无参构造函数

- 【必须】fragment需要无参构造函数，用newInstance创建，数据传递用bundle。
  - Unable to start activity ComponentInfo{...}: androidx.fragment.app.Fragment$InstantiationException: Unable to instantiate fragment ...: could not find Fragment constructor 这个exception是因为系统在恢复fragment时无法调用无参构造函数

Activity的onRestoreInstanceState（在resume前）和onSaveInstanceState（在pause后）：

restore一般不参与生命周期

save在按下home键时必调用，restore看是否被回收

Fragment的onViewStateRestored和onSaveInstanceState：

onViewStateRestored参与生命周期（onstart后必调用）

save在按下home键时必调用

附一个自己写的优化实践

表情键盘优化实践