```
<layout>

    <data>

    </data>

    <com.palline.merit.frame.widget.ReboundScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/white">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="@dimen/dp5"
                android:gravity="center"
                android:textColor="@color/black"
                android:text="成本结构" />

            <include layout="@layout/view_line" />

            <com.palline.merit.test.MyPieChart
                android:id="@+id/mpc"
                android:layout_width="match_parent"
                android:layout_height="@dimen/dp200"
                android:layout_gravity="center"
                android:visibility="invisible" />
        </LinearLayout>
    </com.palline.merit.frame.widget.ReboundScrollView>

</layout>
```

```
package com.palline.merit.frame.widget;

import android.content.Context;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
import android.view.animation.TranslateAnimation;
import android.widget.ScrollView;

/**
 * Created by palline-106 on 2019/3/12.
 */

//仿ios可上提下拉的ScrollView
public class ReboundScrollView extends ScrollView {

    private static final String TAG = "ElasticScrollView";

    //移动因子, 是一个百分比, 比如手指移动了100px, 那么View就只移动50px
    //目的是达到一个延迟的效果
      private static final float MOVE_FACTOR = 0.5f;

    //松开手指后, 界面回到正常位置需要的动画时间
      private static final int ANIM_TIME = 100;

    //ScrollView的子View，  也是ScrollView的唯一一个子View
    private View contentView;

    //手指按下时的Y值, 用于在移动时计算移动距离
      //如果按下时不能上拉和下拉，  会在手指移动时更新为当前手指的Y值
      private float startY;

    //用于记录正常的布局位置
      private Rect originalRect = new Rect();

    //手指按下时记录是否可以继续下拉
      private boolean canPullDown = false;

    //手指按下时记录是否可以继续上拉
      private boolean canPullUp = false;

    //在手指滑动的过程中记录是否移动了布局
      private boolean isMoved = false;

    public ReboundScrollView(Context context) {
        super(context);
    }

    public ReboundScrollView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    @Override
    protected void onFinishInflate() {
        if (getChildCount() > 0) {
            contentView = getChildAt(0);
        }
    }

    @Override
    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        super.onLayout(changed, l, t, r, b);

        if (contentView == null) return;

        //ScrollView中的唯一子控件的位置信息, 这个位置信息在整个控件的生命周期中保持不变
            originalRect.set(contentView.getLeft(), contentView.getTop(), contentView
              .getRight(), contentView.getBottom());
    }
```

```
    //在触摸事件中, 处理上拉和下拉的逻辑
     @Override
    public boolean dispatchTouchEvent(MotionEvent ev) {

        if (contentView == null) {
            return super.dispatchTouchEvent(ev);
        }

        int action = ev.getAction();

        switch (action) {
            case MotionEvent.ACTION_DOWN:

                //判断是否可以上拉和下拉
                        canPullDown = isCanPullDown();
                canPullUp = isCanPullUp();

                //记录按下时的Y值
                        startY = ev.getY();
                break;

            case MotionEvent.ACTION_UP:

                if (!isMoved) break;  //如果没有移动布局，  则跳过执行

                        // 开启动画
                        TranslateAnimation anim = new TranslateAnimation(0, 0,
contentView.getTop(),
                      originalRect.top);
                anim.setDuration(ANIM_TIME);

                contentView.startAnimation(anim);

                // 设置回到正常的布局位置
                        contentView.layout(originalRect.left, originalRect.top,
        originalRect.right, originalRect.bottom);

    //将标志位设回false
    canPullDown = false;
    canPullUp = false;
    isMoved = false;

    break;
  case MotionEvent.ACTION_MOVE:

    //在移动的过程中， 既没有滚动到可以上拉的程度， 也没有滚动到可以下拉的程度
          if (!canPullDown && !canPullUp) {
      startY = ev.getY();
      canPullDown = isCanPullDown();
      canPullUp = isCanPullUp();

      break;
    }

    //计算手指移动的距离
          float nowY = ev.getY();
    int deltaY = (int) (nowY - startY);

    //是否应该移动布局
          boolean shouldMove =
        (canPullDown && deltaY > 0)   //可以下拉， 并且手指向下移动
                        || (canPullUp && deltaY < 0)   //可以上拉， 并且手指向上
移动
                        || (canPullUp && canPullDown); //既可以上拉也可以下拉
（这种情况出现在ScrollView包裹的控件比ScrollView还小）

          if (shouldMove) {
      //计算偏移量
            int offset = (int) (deltaY * MOVE_FACTOR);

      //随着手指的移动而移动布局
```

```
                    contentView.layout(originalRect.left, originalRect.top + offset,
                originalRect.right, originalRect.bottom + offset);

            isMoved = true;  //记录移动了布局
                }

            break;
        default:
            break;
    }

    return super.dispatchTouchEvent(ev);
}



//判断是否滚动到顶部
  private boolean isCanPullDown() {
   return getScrollY() == 0 ||
         contentView.getHeight() < getHeight() + getScrollY();
}

//判断是否滚动到底部
  private boolean isCanPullUp() {
   return contentView.getHeight() <= getHeight() + getScrollY();
}

}
```