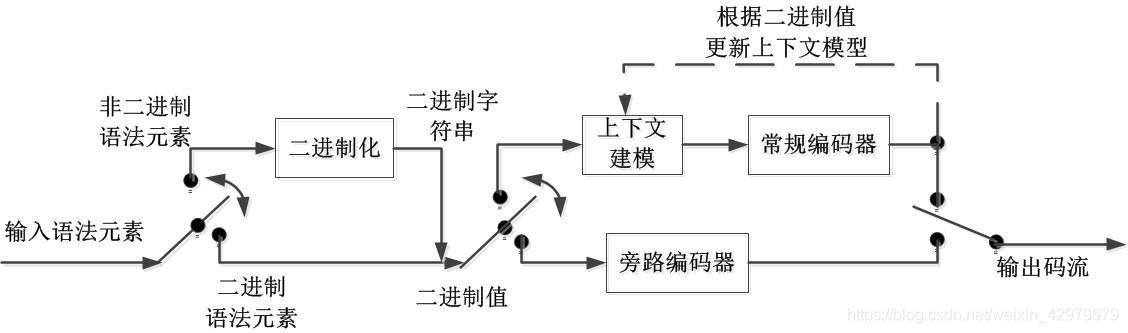
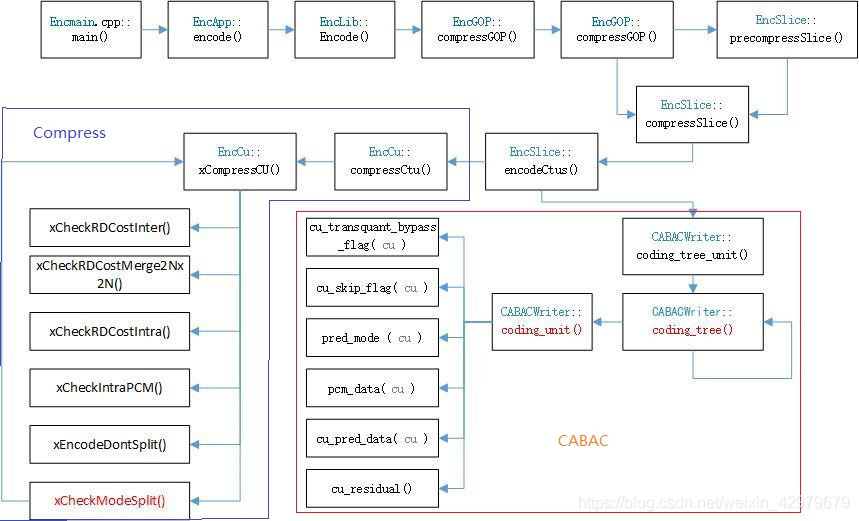
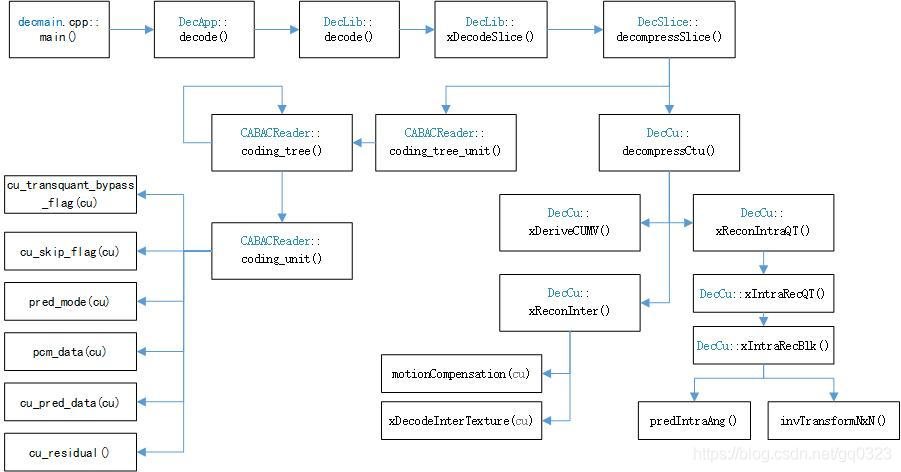
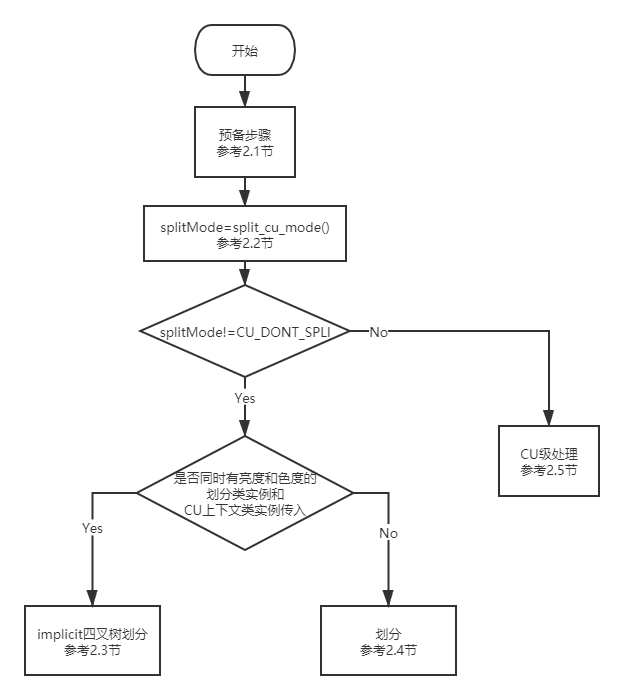
## 熵编码原理：

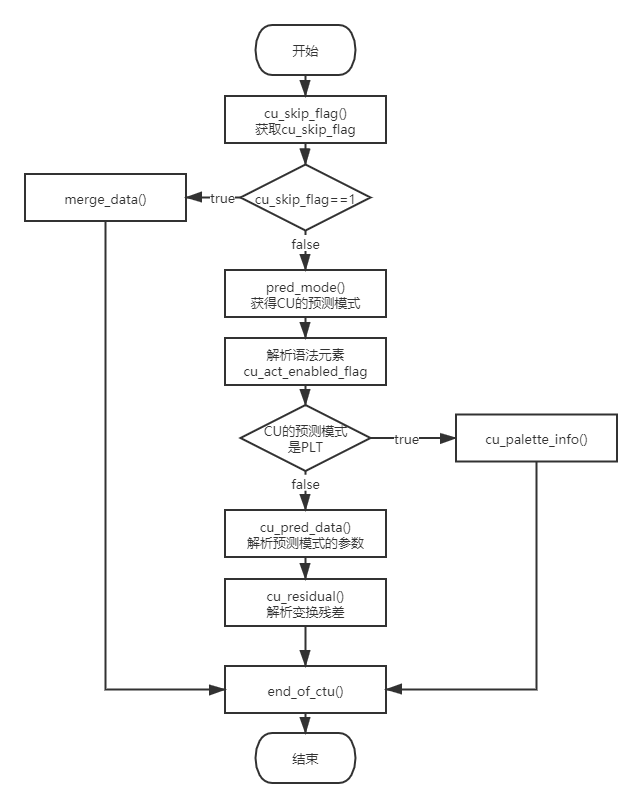


## 熵编码出现的位置、次数：







****

1. 第一次出现在xCompressCU这个大函数里，但是各个参数的熵编码函数是分散在各个小函数中的，比如在xCompressCU里有

encodeBTSplitMode()

encodeSplitFlag();

在xCheckRDCostIntra里面有：

encodeCUTransQuantByPassFlag(),

encodeSkipFlag(),

encodePredMode

等等函数，来完成熵编码的工作；

2.第二次出现是在cmpressSlice函数里面的encodeCtu函数里面，里面整合了所有在1中出现的所有encodeXXXX（）函数；

3.第三次出现是在encodeSlice函数里面的encodeCtu函数里面，同样里面整合了所有在1中出现的所有encodeXXXX()函数；

这三次熵编码的功能是不一样的，分别是

1）.第一次熵编码是因为在xCompressCU里面要确定最优划分和各种预测参数，在比较RDCOST的时候需要用到比特数，而这个比特数是需要经过熵编码来计算的；

2）第二次熵编码是为了在compressCtu完后再进行下一个compressCtu时你要确保下一个CTU的熵编码的状态是上一个CTU熵编码完后的状态；

3）第三次是真正的进行熵编码的过程，其实他跟第二次熵编码的流程几乎是一样的，除了第三次熵编码是要把比特写到码流中，而前两次都是不用输出比特的，而只是计算比特数。

那么这三次的熵编码是否输出比特流是如何区分的呢？

在代码中，

compressSlice()

{

m\_pcEnctropyCoder->setBitstream(&tempBitCouter);

m\_pcEntropyCoder->compressCtu();

m\_pcEnctropyCoder->setBitstream(&tempBitCouter);

m\_pcEntropyCoder->encodeCtu();

}

encodeSlice()

{

m\_pcEnctropyCoder->setBitstream(&pcSubstream[uiSubStream]);

m\_pcEntropyCoder->encodeCtu();

}

其中，pcSubstream是TComOutputBitstream类，tempBitCouter是TComBitCounter类，前者是输出比特流，后者是比特计数器，所以，通过这两个之间的区分就能确定熵编码是只是计算比特数/保持状态，还是进行真正的熵编码会有输出。