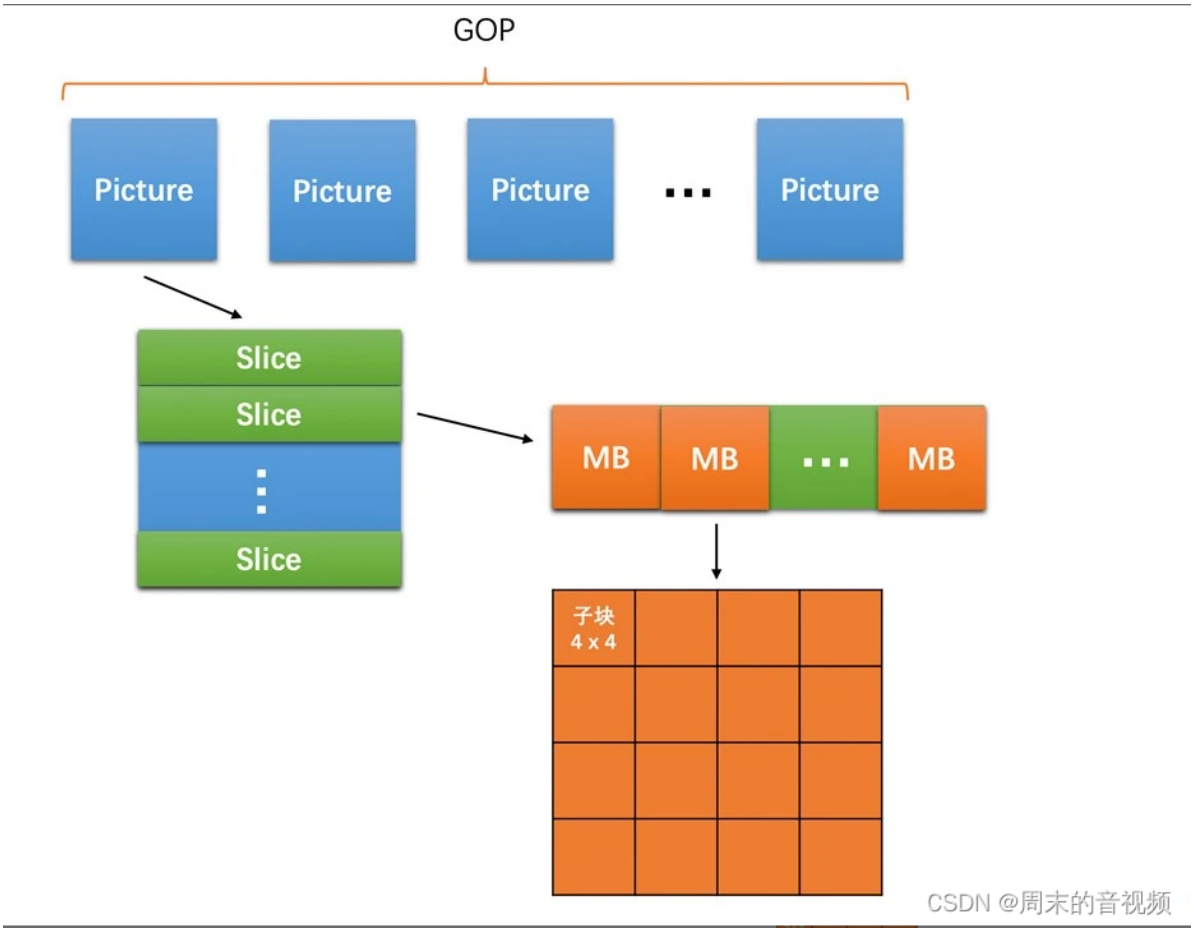
H.264码流结构

1.Slice

Slice其实就是“片”的概念，图像内的层次结构就是一帧图像可以划分成一个或多个 Slice，而一个 Slice 包含多个宏块，且一个宏块又可以划分成多个不同尺寸的子块。大概就是类似下面这样的结构图：



2.码流格式

H264 码流有两种格式：一种是 **Annexb** 格式；一种是 **MP4** 格式。

2.1 Annexb 格式

Annexb 格式使用起始码来表示一个编码数据的开始。起始码本身不是图像编码的内容，只是用来分隔用的。起始码有两种，**一种是 4 字节的“00 00 00 01”**，**一种是 3 字节的“00 00 01”**。由于图像编码出来的数据中也有可能出现“00 00 00 01”和“00 00 01”的数据。为了防止出现这种情况，H264 会将图像编码数据中的下面的几种字节串做如下处理：

“00 00 00”修改为“00 00 03 00”；

“00 00 01”修改为“00 00 03 01”；

“00 00 02”修改为“00 00 03 02”；

“00 00 03”修改为“00 00 03 03”。

其实也就是转义，同样地在解码端，我们在去掉起始码之后，也需要将对应的字节串转换回来。

Annexb格式

2.2 MP4 格式

MP4 格式没有起始码，而是在图像编码数据的开始使用了 **4 个字节**作为长度标识，用来表示编码数据的长度，这样我们每次**读取 4 个字节**，计算出编码数据长度，然后取出编码数据，再继续读取 4 个字节得到长度，一直继续下去就可以取出所有的编码数据了

MP4格式

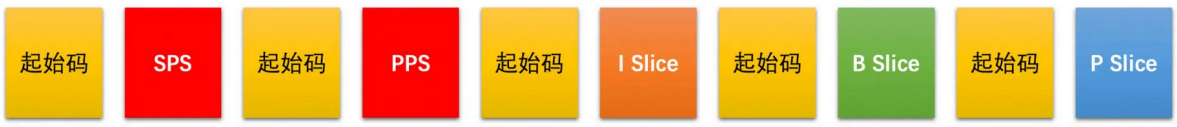
这两种格式差别不大，接下来我们主要使用 Annexb 格式来讲解 H264 码流结构

3.SPS和PPS

视频编码的时候还有一些编码参数数据的，为了能够将一些通用的编码参数提取出来，不在图像编码数据中重复，H264 设计了两个重要的参数集：一个是 SPS（序列参数集）；一个是 PPS（图像参数集）。 SPS 主要包含的是图像的宽、高、YUV 格式和位深等基本信息；PPS 则主要包含熵编码类型、基础 QP 和最大参考帧数量等基本编码信息。

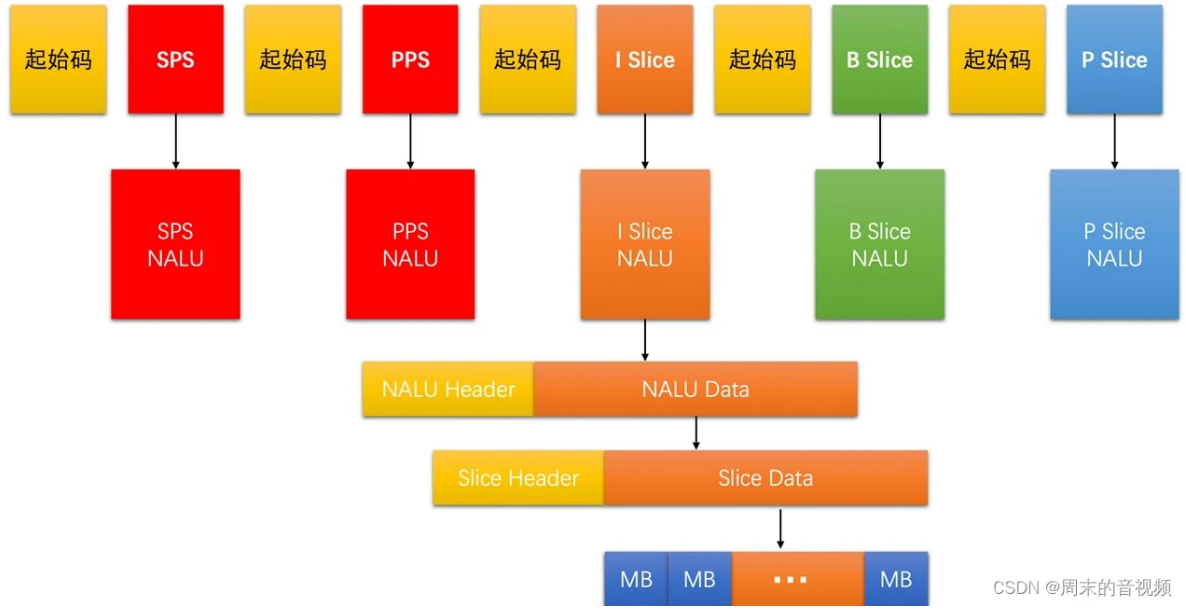
如果没有 SPS、PPS 里面的基础信息，之后的 I 帧、P 帧、B 帧就都没办法进行解码。因此 SPS 和 PPS 是至关重要的。

这样的话，H264码流主要包含了 SPS、PPS、I 帧、P 帧和 B 帧。由于帧又可以划分成一个或多个 Slice。因此，帧在码流中实际上是以 Slice 的形式呈现的。所以，H264 的码流主要是由 SPS、PPS、I Slice、P Slice和B Slice 组成的。如下图所示：

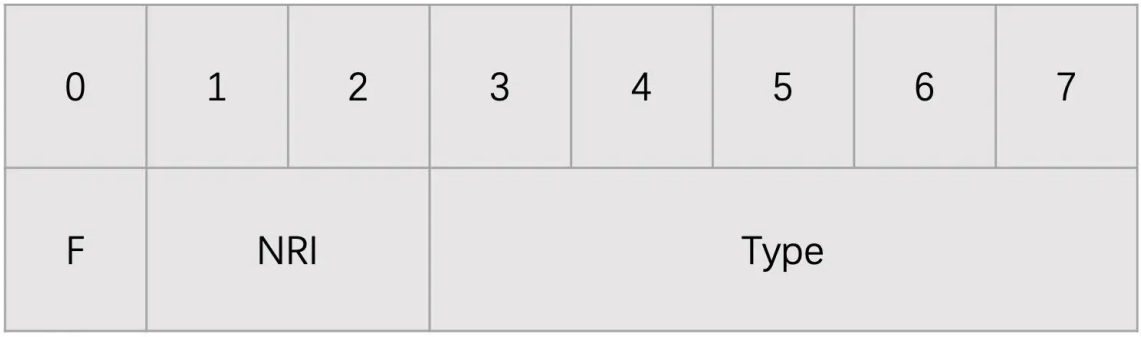


4.NALU

上面说到H264码流的组成部分，但是每个部分是如何区分开？为了解决这个问题，H264 设计了 NALU（网络抽象层单元）。SPS 是一个 NALU、PPS 是一个 NALU、每一个 Slice 也是一个 NALU。每一个 NALU 又都是由一个 1 字节的 NALU Header 和若干字节的 NALU Data 组成的。而对于每一个 Slice NALU，其 NALU Data 又是由 Slice Header 和 Slice Data 组成，并且 Slice Data 又是由一个个 MB Data 组成。其结构如下



其中，NALU Header总共占用 1 个字节，具体如下图所示



其中，

--> F：forbidden_zero_bit，占 1bit，禁止位，H264 码流必须为 0；

--> NRI： nal_ref_idc，占 2bits，可以取 00～11，表示当前 NALU 的重要性。参考帧、SPS 和 PPS 对应的 NALU 必须要大于 0；

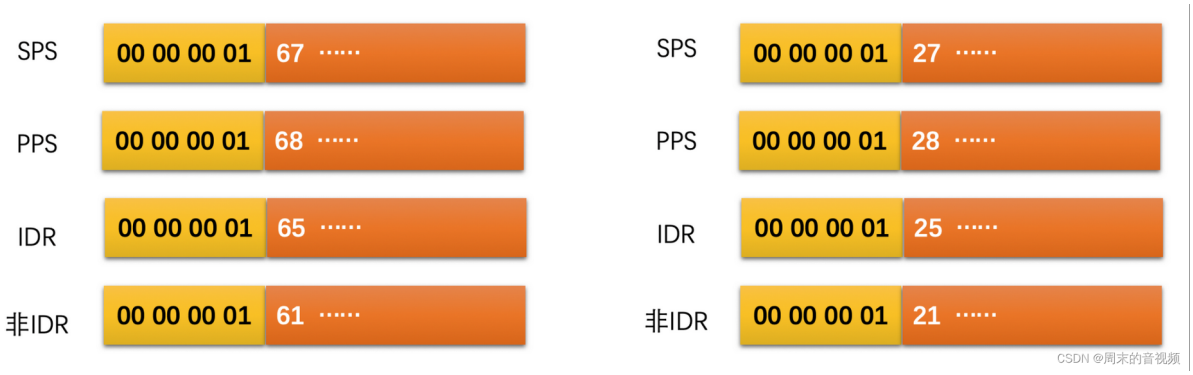
--> Type： nal_unit_type，占 5bits，表示 NALU 类型。其取值如下表所示

nal_unit_type	0	1	2	3	4	5	6	7	8	9	10	11	12	13···23	24···31
NAL类型	未使用	不分区，非IDR图像的片	片分区A	片分区B	片分区C	IDR图像中的片	补充增强信息单元（SEI）	序列参数集（SPS）	图像参数集（PPS）	分界符	序列结束	码流结束	填充	保留	未使用

有了这个，我们解析出 NALU Header 的 Type 字段，查询表格就可以得到哪个 NALU 是 SPS，哪个是 PPS，以及哪个是 IDR 帧了。不过NALU 类型只区分了 IDR Slice 和非 IDR Slice，至于非 IDR Slice 是普通 I Slice、P Slice 还是 B Slice，则需要继续解析 Slice Header 中的 Slice Type 字段得到。

有了这个，我们解析出 NALU Header 的 Type 字段，查询表格就可以得到哪个 NALU 是 SPS，哪个是 PPS，以及哪个是 IDR 帧了。

不过NALU 类型只区分了 IDR Slice 和非 IDR Slice，至于非 IDR Slice 是普通 I Slice、P Slice 还是 B Slice，则需要继续解析 Slice Header 中的 Slice Type 字段得到。 下面我们通过两个例子来看看常见的 NALU 里的 NALU Header 是怎样的：



下面我们再来看一个实际码流的例子，看看在实际编码出来的二进制数据中，各种 NALU 是怎么“放置”在数据中的。下图是用二进制查看工具打开实际编码后的码流数据。我们可以看到在码流的开始部分是一个起始码，之后紧接着是一个 SPS 的 NALU。在 SPS 后面是一个 PPS 的 NALU。然后就是一个 IDR Slice 的 NALU 和一个非 IDR Slice NALU。

```
0000000000  00 00 00 01 67 64 0c 16 ac 19 1a 80 a0 2f 79 60
0000000010  1e 11 08 d4 00 00 00 01 68 ee 3c 80 00 00 00 01
0000000020  65 b8 00 04 00 00 09 ff e5 7e e2 36 c3 b3 23 b5
0000000030  e1 c2 7a 79 80 09 1f 70 d0 fa 1f c9 85 92 90 8f
0000000040  1f ff 57 17 b0 88 35 4e 00 70 f7 07 cc 58 01 98
0000000050  b5 7e 6c dc 41 a0 d0 52 c0 bd ec 73 e9 5b 2d f9
0000000060  8b a0 89 5c 6f e2 e2 0c cf 0d 70 d2 ce 6f af 99
0000000070  b7 2e d4 fa dd 0f 16 2b d6 54 e8 7a 79 da 52 23
0000000080  02 92 af da 40 5e 58 d0 31 f8 36 26 0b b7 56 fc
0000000090  86 5e f5 28 ed 60 c0 98 fe 9e e5 bd 98 36 64 b3
00000000a0  f2 69 46 dc b4 82 29 06 9d 88 49 1b f0 e7 bb b5
00000000b0  af 9c a1 76 2c 7e d8 66 16 80 5a d5 25 e7 1f 68
00000000c0  94 62 b4 35 30 b7 c1 b5 69 4a 3a bf 8c 09 a5 4d
00000000d0  3d 85 57 75 1b 4e 0b ff cc 86 84 d4 4e 6c 87 a8
00000000e0  5b 5e ef 8c c3 7f fc 9e d5 26 35 a0 61 98 14 c6
00000000f0  86 29 24 30 a5 9c bd ee f4 4f e9 a5 9f e3 03 9b
0000000100  41 67 2f fb 7e ba 83 5b 4f 8e 1d 5b 93 c9 a6 cf
0000000110  fb 5e 5b 5d ea b2 af 68 3e 3f 9a 2f 67 da 90 c3
0000000120  25 8a 3c 48 10 4b e9 61 52 2a 59 69 3e 79 2c ce
0000000130  d9 ec 8a f0 28 93 db 32 13 a7 b8 d1 f3 b9 80 80
0000000140  4d a2 95 31 11 52 fe 66 8e 36 49 df 83 c7 96 f2
0000000150  f6 b1 ef 3d f2 d2 fa 4b 5d 21 69 04 19 9f 80 c0
0000000160  ce ad d9 a5 8d 05 1d 6a 0e 77 67 df 2a fe 16 0a
0000000170  33 d9 dc 35 ed b6 b7 8a aa 9e 3f 0c b3 15 58 ea
0000000180  36 b9 a7 87 ad 98 ab 77 69 25 08 ef 76 f0 b2 94
0000000190  d7 9c b0 d5 e9 b2 0c 1e 6e 37 a3 27 76 7c 06 4a
```



```
000000a010  20 0a 5c 81 cc ce b2 ac 6f c0 c0 d6 6b 58 3a 71
000000a020  f5 c7 30 29 a1 96 ba 55 76 56 00 00 00 01 61 e0
000000a030  00 40 00 9c 88 9f bb 36 09 06 dc 24 64 7d a8 c0
000000a040  49 40 97 75 a7 b0 e9 39 bb 65 f7 93 72 96 28 9f
000000a050  bf 39 c2 c4 be 6a d3 91 21 09 d1 a4 29 b2 c4 75
000000a060  80 fc c8 1a af e6 36 59 41 b9 00 78 e4 33 d9 24
000000a070  2e bc b4 59 cd 00 1a ac f7 4d b0 85 65 6c ae aa
000000a080  75 9e 61 73 3a a8 ea 38 aa f1 e4 96 0a 71 6c 1d
000000a090  23 5d de 3f 47 29 0f 77 41 59 f2 77 2e 47 a0 24
000000a0a0  84 77 ec be 88 3c 88 68 d7 ca 2a 35 6a 1b ed 7a
000000a0b0  98 08 42 b0 31 7c 26 33 56 7e 0b 2d 83 23 c8 d9
000000a0c0  6e 30 88 ae af 29 a3 db 50 86 f3 8d 41 34 9e 48
000000a0d0  a0 f9 39 c6 8a a5 bd 55 1d 8b b3 54 aa 3c 80 63
000000a0e0  67 fe 7c 25 b1 0f db 08 3c 54 67 a8 87 41 70 f9
000000a0f0  b9 a5 15 a7 82 d3 1c 58 04 1d 75 80 19 45 15 db
000000a100  60 0c 66 da a9 8f 4f e5 69 ad 35 a5 1b 44 a3 ed
000000a110  92 b0 81 c9 d0 a1 ea e8 71 89 45 22 49 d9 7c 57
000000a120  9b bc b4 75 f1 3c d0 35 96 0f 67 ac de 5f 53 2f
000000a130  7a 05 e3 9a 2c 01 7d e4 ab 00 3b db bd cb b7 17
000000a140  eb e8 09 5b 32 73 1d ae 70 9e dc 25 0d 5d 9c 61
000000a150  e5 24 8e 6f 72 f6 de ff 63 75 75 d9 19 d5 43 ab
000000a160  28 ff c0 66 41 5b 22 0d 32 01 ab e6 9b cf c8 10
000000a170  1d 37 3b 82 45 03 2d bd dc 29 d2 85 da 77 1e 77
000000a180  61 43 77 48 19 c6 41 65 74 f7 72 97 02 c9 91 85
000000a190  25 33 67 c0 77 e6 b2 65 68 b8 48 1b 22 70 63 81
000000a1a0  8d 1e 6f e8 19 5d 42 98 1d c0 90 48 d5 eb 50 56
```

起始码
SPS
PPS
IDR
非IDR

如何判断哪几个 Slice 是同一帧的

根据上面的分析，在H264 码流中，帧是以 Slice 的方式呈现的，或者可以说在 H264 码流里是没有“帧”这种数据的，只有 Slice。 那么有个问题，一帧有几个 Slice 是不知道的。也就是说码流中没有字段表示一帧包含几个 Slice。既然没有办法知道一帧有几个 Slice，那我们如何知道多 Slice 编码时一帧的开始和结束分别对应哪个 Slice 呢？ 其实，Slice NALU 由 NALU Header 和 NALU Data 组成，其中 NALU Data 里面就是 Slice 数据，而 Slice 数据又是由 Slice Header 和 Slice Data 组成。在 Slice Header 开始的地方有一个 first_mb_in_slice 的字段，表示当前 Slice 的第一个宏块 MB 在当前编码图像中的序号。我们只要解析出这个宏块的序号出来。

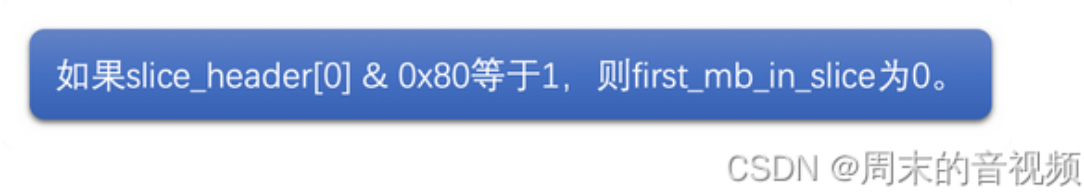
--> 如果 first_mb_in_slice 的值等于 0，就代表了当前 Slice 的第一个宏块是一帧的第一个宏块，也就是说当前 Slice 就是一帧的第一个 Slice。

--> 如果 first_mb_in_slice 的值不等于 0，就代表了当前 Slice 不是一帧的第一个 Slice。并且，使用同样的方式一直往下找，直到找到下一个 first_mb_in_slice 为 0 的 Slice，就代表新的一帧的开始，那么其前一个 Slice 就是前一帧的最后一个 Slice 了。

slice_header( ) {	C	描述符
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(nal_unit_type == 5)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type == 0) {		
pic_order_cnt_lsb	2	u(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type == 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0]	2	se(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1]	2	se(v)
}		

图片来源于H264标准文档 CSDN @周末的音视频

其中，first_mb_in_slice 是以无符号指数哥伦布编码的，需要使用对应的解码方式才能解码出来。但是有一个小技巧，如果只是需要判断 first_mb_in_slice 是不是等于 0，不需要计算出实际值的话，只需要通过下面的方式计算就可以了。



CSDN @周末的音视频

5.补充部分（重要）

H.264（也称为AVC，高级视频编码）码流中，起始码（Start Code）是用来指示码流的NAL单元（Network Abstraction Layer Units）的开始。在H.264中，一个帧（无论是I帧、P帧还是B帧）是由一个或多个slice组成的，而每个slice可能包含一个或多个宏块。

在解码过程中，起始码用于标识一个NAL单元的开始，而NAL单元是H.264码流的基本单元。以下是关于起始码使用的一些细节：

1. **每个NAL单元前都有一个起始码**：在H.264码流中，每个NAL单元前都会有一个起始码，而不是每个帧或每个slice前都有一个。这意味着无论是I帧、P帧还是B帧，它们的每个slice前都会有起始码，因为每个slice都被封装成一个NAL单元。
2. **帧内可能包含多个NAL单元**：一个帧可能由多个slice组成，每个slice被封装成一个NAL单元，因此一个帧可能对应多个NAL单元，每个NAL单元前都有自己的起始码。
3. **NAL单元的边界**：NAL单元的边界定义了解码过程中的处理单元。当解码器遇到一个起始码时，它知道一个新的NAL单元开始了，然后根据NAL单元的类型（如VCL NAL单元，包括I、P、B帧的slice数据，或者非VCL NAL单元，如序列参数集SPS或图像参数集PPS）进行相应的处理。

因此，回答您的问题：在解码时，是每个NAL单元（通常是每个slice）之间使用一个起始码，而不是每个帧之间。每个slice被视为一个独立的解码单元，并带有自己的起始码，以便在传输过程中可以进行错误检测和单元分割