## 一、首先理清数据的相互之间的对应关系

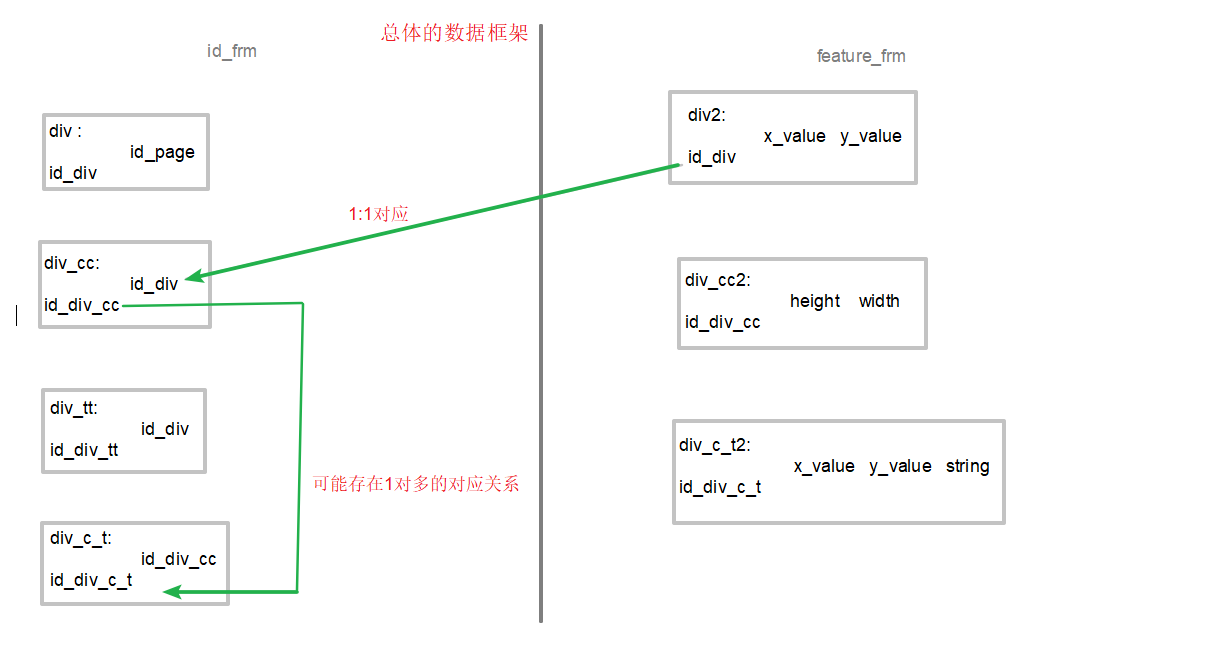
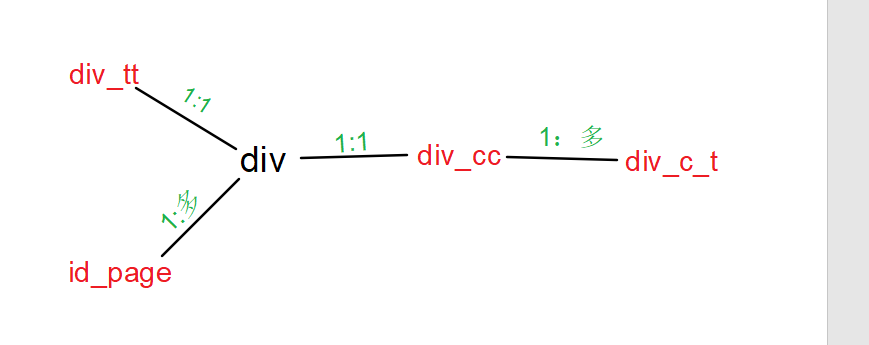


图1

1. 总体的数据分为两部分（id\_frm，feature\_frm），其中涉及三个个体（div\_tt , div\_cc , div\_c\_t）相互之间的关系。
2. 对于每一个div\_tt，都有唯一一个div与之对应，它（div\_tt）可以通过与之唯一对应的div来获取自己的x\_value , y\_value , id\_page等属性。
3. 对于每一个div\_cc, 都有唯一一个div与之对应，它（div\_cc）可以通过与之唯一对应的div来获取自己的x\_value , y\_value , id\_page等属性。但是一个div\_cc可以对应多个div\_c\_t。
4. 对于每一个div\_c\_t,都有唯一一个div\_cc与之对应，他可以通过div\_c\_t2取出自己的属性。



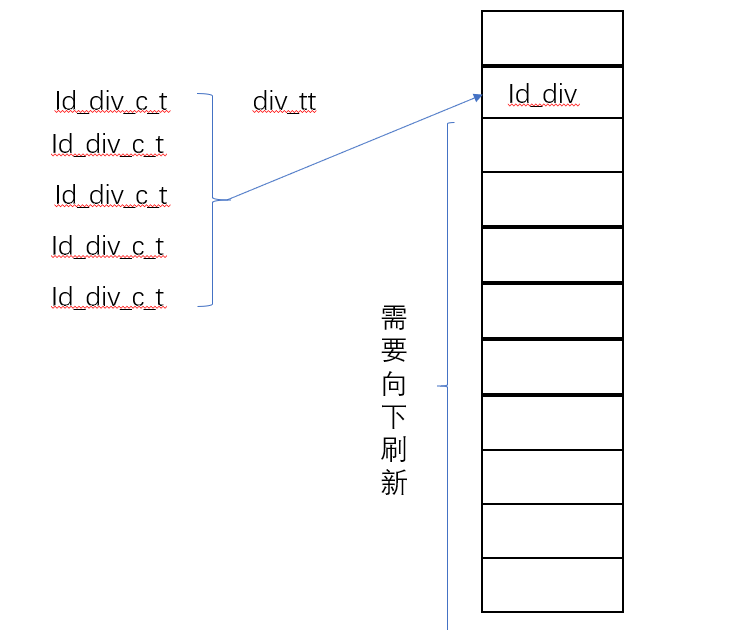
图二

## 二、解决思路

### Problem1

1. 问题分析

Problem1：所有的div标签全是div\_cc标签，也可以通过 root[‘id\_frm’][‘div\_tt’] 发现是空的，要求x\_value < 10的div\_cc标签全部删除，div\_cc里面的div\_c\_t就转为div\_tt标签，并刷新id\_div. 同时需要在div\_tt里面添加记录，在div里面添加（id\_div , id\_page）的对应关系



图三

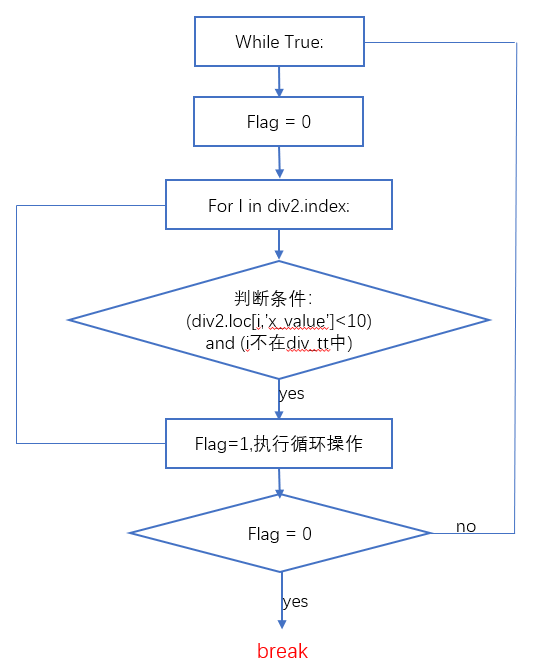
2. 解决思路

如上所述，我们需要选出x\_value < 10的div标签，当我们在选出一个目标div标签时，就需要进行如下操作，id\_div 🡪 id\_div\_cc 🡪 多个id\_div\_c\_t，

对于每一个符合条件的id\_div处理流程如下：

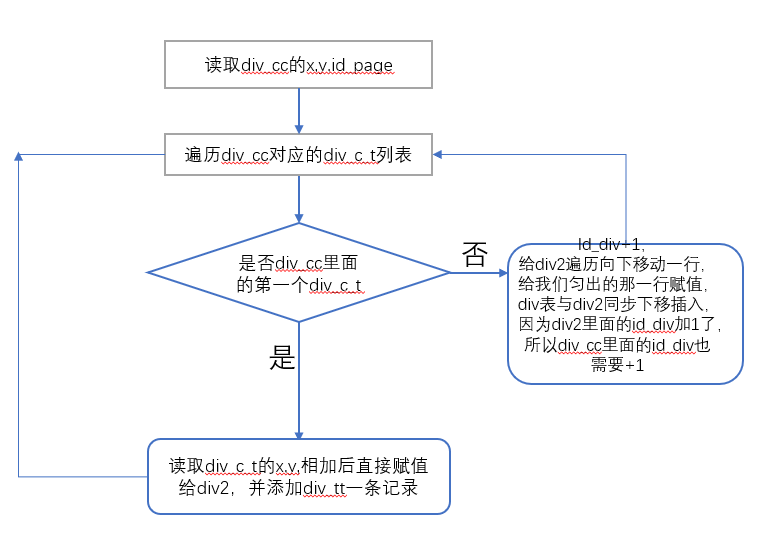
* 对于每一个符合条件的id\_div的处理之后都需要动态的刷新div2表，div表，给div\_tt增加行。
* 从div2表里面找出符合条件的id\_div, 我采取的办法是: 最外面用死循环包裹，设立一个标志flag = 0,用于检测，当进入过if里面执行（条件满足），flag=1,这样来判断是否需要跳出，当遍历div2一遍后，没有进入if语句。即flag=0,那么此时就可以判定div2里面已经没有满足条件的div\_tt标签，直接跳出即可。
* 总的概括就是：查找🡪处理🡪刷新🡪再查找……，如此循环往复。直到找不到符合条件的值

流程图如下：



图四

下图是上图的**执行循环操作**的详细流程图：



图五

### Problem2

1. 问题分析：

Problem2：要求将两个div\_tt标签之间的div\_cc全部拆掉，div\_cc里面的div\_c\_t全部转化为div\_tt标签，并插入到idv2中，通过分析发现第一个id\_div对应的是div\_tt，所以对于每一个id\_div需要满足两个条件：

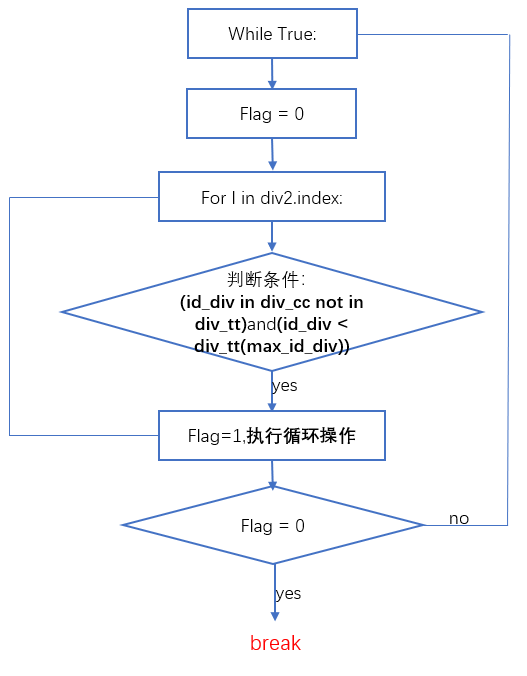
* **此id\_div必须在div\_cc里面找到，而不能存在于div\_tt里面**，因为我们最后一起删除的，所以会产生许多既存在于div\_tt中又存在于div\_cc中，因此在筛选时需要注意
* **id\_div必须小于此时的最后一个div\_tt标签的id\_div，**需要注意，随着div2行数的扩充，会导致最后一个div\_tt标签的id\_div会变大。

1. 解决思路

还是通过遍历div.index来挨个进行if判断，当判断成立的话通过id\_div 🡪 id\_div\_cc 🡪 多个id\_div\_c\_t，来获取div\_c\_t，通过之前的分析可知，div\_cc与div\_c\_t的比值为140：147，所以可以提前推断，每个div\_cc里面仅存在一个div\_c\_t占大多数。

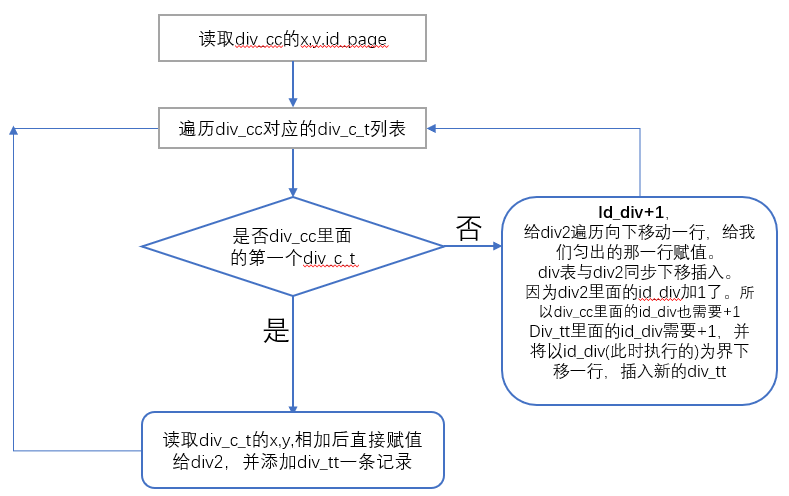
* 对于每一个符合条件的id\_div的处理之后都需要动态的刷新div2表，div表，给div\_cc里面的列值（列值>=id\_div）增加1，给div\_tt里面的列值（列值>=id\_div）增加1，然后向后移动一行在合适的位置插入该行，这样才可以保证互相连通的id\_div不会错乱。
* 从div2表里面找出符合条件的id\_div, 我采取的办法是: 最外面用死循环包裹，设立一个标志flag = 0,用于检测，当进入过if里面执行（条件满足），flag=1,这样来判断是否需要跳出，当遍历div2一遍后，没有进入if语句。即flag=0,那么此时就可以判定div2里面已经没有满足条件的div\_tt标签，直接跳出即可。

流程图如下：



图六

下图是上图的**执行循环操作**的详细流程图：



图七

## 三、代码结构

### 1. 公共部分抽取

前言：对两份代码的抽取操作，我并没有去高度的解耦，因为在循环查找，处理，刷新那一部分虽然相似之处较多，但抽取的话不好公用，故我仅仅对数据的读取赋值，处理完成后的删除，最后的写入文件做了抽取，因为他们是完全可以公用的。

### 2. 代码结构图解

* Find()函数的作用是根据遍历的id\_div，找到对应的id\_div\_cc，或者根据id\_div\_cc找到对应的id\_div\_c\_t，为了能共用一个函数，我设定的返回对象是一个数组，但通过的数据对应特性我们可以很轻松的取值。
* Judge()函数是用于if条件中其中一个条件的判断，在pro1中，我们要求该id\_div是不能存在于div\_tt中。在pro2中，我们要求该id\_div存在于div\_cc中，同时不能存在于div\_tt中。
* Print\_info()，就是输出处理信息的函数。



图八