按照请求的类别分类

所有标黄的地方才是需要真正请求的地方

之前没有理解清楚Id1---> Id2 的边是单向边

1. Id1 ---> Id2
   1. 1-hop

请求 Id1, （比较Rid

* 1. 2-hop
* Id1 --- {C.Cid, J.JId, F.Fid, AA.AuId} --- Id2

请求Id2 和 {1.1 1-hop} 中已经得到了数据，其中C.Cid J.JId都是唯一的 F.Fid AA.AuId 是多个

* Id1 ---> Id3 ---> Id2

请求 {Id1.Rid} 也就是请求{Id3}

* 1. 3-hop
* Id1 ---> Id3 --- ~Id --- Id2

~Id 表示 除了Id的其它类，在这个地方只有可能是 {C.Cid, J.JId, F.Fid, AA.AuId}

这一类不用再次请求 2-hop请求中的已经足够了

* Id1 --- ~Id --- Id3 ---> Id2

请求 Rid=Id2 的所有{Id}

* Id1 ---> Id3 ---> Id4 ---> Id2

在1-hop中已经有了所有的Id3，在3-hop第2类中有所有的Id4

1. AA.AuId1 ---> AA.AuId2
   1. 1-hop

无

* 1. 2-hop

AuId1 --- {AA.AfId, Id} --- AuId2

请求AuId1 AuId2

* 1. 3-hop

AuId1 --- Id2 ---> Id3 --- AuId2

上边的{2.2 2-hop} 已经得到了足够的数据

1. Id1 ---> AuId2
   1. 1-hop

Id1 ---> AuId2

请求Id1，匹配一下AuId2就OK

* 1. 2-hop

Id1 ---> Id3 --- AuId2

请求一次AuId2就能判断了

* 1. 3-hop
* Id1 --- {C.Cid, J.JId , F.Fid } --- Id3 --- AuId2
* Id1 ---> Id3 ---> Id4 --- AuId2

需要请求{Id3}的Rid 或者 请求Rid=Id4的集合

* Id1 --- AuId3 --- Id4 ---- AuId2
* Id1 --- AuId3 --- AfId4 --- AuId2