缓存胡牌组合【key=手牌映射值（3N+1）】

性能优化相关文件：

bin\game\common\core\Base\CacheManager.lua 【缓存管理类】

bin\game\common\core\Base\mahjongUtil\cacheUtils.lua 【缓存方法集】

bin\game\common\core\Base\mahjongUtil\CheckUtils.lua 【性能优化方法集】

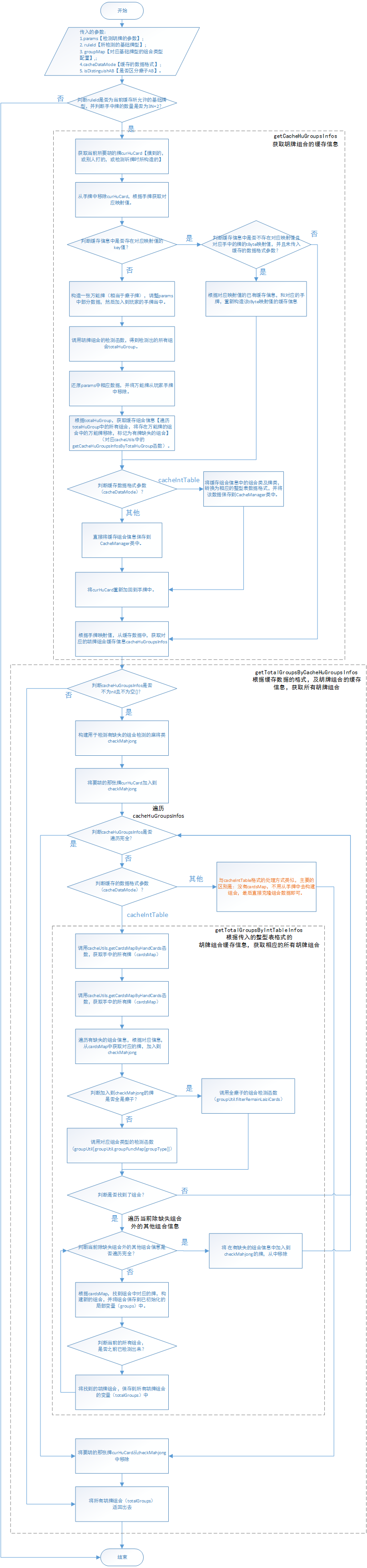
适用情况：

有/无癞子+听牌的麻将（注意：不是所有无癞子的麻将，使用这种缓存，都能提升性能！）

一、 使用方式

略。

二、 流程图



三、 流程解析

该接口函数与获取胡牌组合（huUtil.getTotalHuGroupByGroup）函数的所需参数相同，都是params和groupMap（所检测的组合类型配置）。

在该接口函数中，主要做两件事，首先获取缓存的胡牌组合信息，然后利用所取得的胡牌组合缓存信息，得到所有胡牌组合。

该接口函数的具体代码，对应bin\game\common\core\Base\CacheManager.lua中的getTotalGroups函数。

以下简要说明缓存信息检测及使用简要步骤：

1. 获取缓存的胡牌组合信息（对应CacheManager中的getCacheHuGroupsInfos函数） 主要分为七个步骤：

（1） 根据params参数，将玩家的手牌移除一张牌（params.card），而这张牌可能是摸到的，也可能是别人打的，还可能是检测听牌时所构造的；

（2） 根据玩家的手牌，获取对应的映射值（可调用g\_CacheManager:getMappingValByHandCards获取）；

（3） 判断缓存数据中是否已存在该映射值的信息，若已存在则直接跳到（7）；

（4） 构造一张万能牌【即癞子牌】，添加到玩家的手牌中，后调用获取胡牌组合（huUtil.getTotalHuGroupByGroup）函数，得到所有胡牌组合后，将万能牌从玩家的手牌中移除；

（5） 获取所有胡牌组合，进行遍历，将有万能牌的组合找出，然后构造所需的缓存数据信息；

（6） 将所获得的缓存数据信息保存到指定的变量；

（7） 将params.card重新加入到 玩家的手牌中，返回所找到的缓存数据。

2. 利用胡牌组合缓存信息，得到所有胡牌组合（对应CacheManager中的getTotalGroupsByCacheHuGroupsInfos函数）

该段逻辑，最主要的是进行缓存信息（可能有多种组合）中所缺失组合的检测。

其中最重要的是检测的手牌，不能直接使用玩家的手牌去检测（毕竟不是检测所有组合），这时候，可以构造一个新的麻将类（为了不重复构建，可以保存在外部变量中，只不过每次使用 要记得清除该类）。

将params.card（或params.curHuCard）和 缺少一张牌的组合中的牌，加入到 新麻将类中，然后根据该组合类型，调用组合函数的检测逻辑，若该组合能组合完成，则保存对应组合到所有组合中（初始值是个空表）。

四、 特点分析

1. 优点

（1） 除开缓存的性能影响外，使用该找组合的方式，能减少听牌检测中的胡牌组合检测次数，使得对于手中牌数量较多的情况下，听牌检测时间能减少，即使是第一次检测听牌；

（2） 使用缓存会减少重复检测的次数，使得不会造成不必要的性能浪费；

（3） 由于目前是一局结束后，立即清除缓存，所以不会造成多局后运行内存增长过快的情况。

2. 缺点

（1） 对于既无癞子又无听牌的麻将，使用这种缓存方式，性能提升不大，而且还可能在手中牌数量较少的情况下，出现性能降低的情况（由于检测中使用了万能牌的原因）。