从DSG出发，用unit-wise方法计算group skyline。

### 基本思路

主要分为以下几个步骤：

1. 以上一步骤输出的DSG为输入，简化数据结构，构造RefDSG，即Reference-based DSG。这是因为上一步输出的DSG的节点DSGNode是通过List中的下标来存储其父母节点和孩子节点的，而在unit-wise方法中，在涉及unit group的操作时需要频繁查询和合并许多节点的父母节点和孩子节点。将DSGNode转换为基于引用的RefNode，用HashSet保存节点的父母节点和孩子节点的引用，并抛弃无用属性，可以提高效率。另一方面，将排序后的unit group和node作为RefDSG的成员变量，可以提高抽象层次，提高可维护性。
2. 按照论文中的实现方式，将nodes和unit groups按照原本的索引反向排序。
3. 构造仅由单一unit group组成的初始groups列表，即singleUnitGroups。
4. 对每一个初始group：
5. 检查Glast。
6. 以该group为初始节点，构建枚举树，并按照论文中的方法进行剪枝。在这里我们结合java的语言特性做了一些改进。
7. 输出结果

### 代码实现

1. RefNode：

基于引用的DSG Node。用originPointIdx表示原本的节点索引，parents表示其所有的祖先节点，children表示其所有的后代节点。这里沿用了论文中parent和children的称谓，虽然这个命名不是很精确。这里使用HashSet是为了提高查询和合并的效率。

1. Unit类

单个unit group，记录其起始节点和所有长辈节点。

1. UnitGroup类

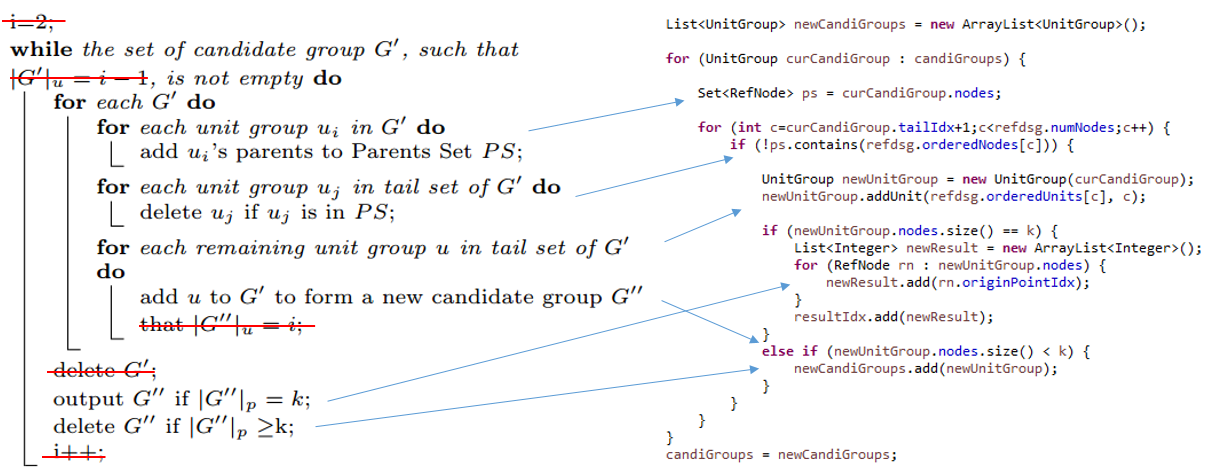
由unit group组合而成的group。记录其在枚举树tail list中的位置和包含的节点。

1. RefDSG

基于引用的DSG。保存排序的nodes和units。

1. UnitWise类

负责应用论文中描述的unit-wise方法进行计算并输出结果。

在具体的实现上，我们结合java的语言特性对论文中的方法进行了优化。以下面这段代码为例。

论文中所说的一个node的parent set其实可以等价替换为对应unit group的nodes集合。这两个集合的差集就是这个node本身，这一差别对剪枝没有影响。这个集合其实我们已经在之前求出来了，因而大大提高了效率。

在求tail set的时候，我们没有像论文中一样“删去在parent set中的unit group”，而是“只添加不在parent set中的unit group”。考虑到java中ArrayList的删除操作效率低下，LinkedList的插入操作效率又低于ArrayList，这样做可以使我们既使用ArrayList又避免删除操作，提高了效率。

文中使用变量i来索引不同深度的枚举集合。考虑到java的回收机制，我们直接用新的枚举集合替代旧的，以使得旧的集合占据的资源可以被释放和回收。

我们对论文提出算法的若干循环做了最大限度的合并，以提高效率。