从预处理后得到的DSG出发，DSG中的DSGNode已经按照index由小到大排列，用point-wise方法计算group skyline

基本思路

point-wise方法的基本思路就是从空集开始，逐层生成大小为1，2…k-1，k的G-skyline groups。当已经得到大小为i的G-skyline groups之后，遍历每个大小为i的G-skyline group，每个G-skyline group都有一个尾集合，尾集合包含所有index大于G-skyline group中所有点的index的点，遍历尾集合中的点，向G-skyline group中增加尾集合中的点构成大小为i+1的candidate group，然后检查该candidate group是否是G-skyline group，如果是，就将candidate group加入大小为i+1的G-skyline groups。point-wise尽可能的在每一步剪枝。以下是point-wise的几个剪枝方法：

1. Subtree Pruning：一旦某个大小为i的candidate group被检查之后发现不是G-skyline group，那么往其中加入任何点都不能构成大小为i+1的G-skyline group.所以我们可以直接舍弃这个点不存储
2. Tail Set Pruning：

A、G-skyline group中如果包含某个点p，那么点p的父母节点一定也在这个G-skyline group中。所以我们可以对G-skyline group的tail set进行删减，我们首先计算得到G-skyline group中所有节点的孩子节点集合，tail set中的点如果不在孩子节点集合中或者属于第一层skyline则直接从tail set中删除该点。这样可以减少candidate groups的大小。

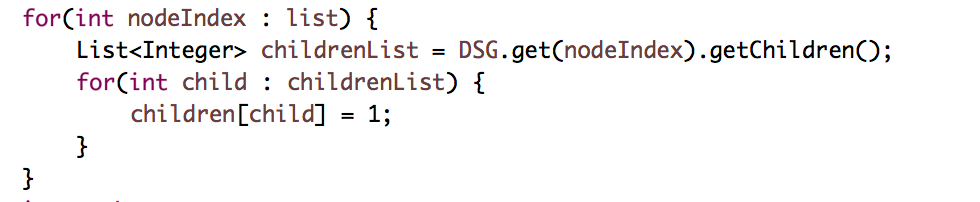
B、假设G-skyline group中节点的最大层数为i，则加入的点p必须来自前i+1层。所以tail set中的点如果层数大于i+1也被直接删除。

总的来说，point-wise包含三个循环，最外的循环每循环一次生成一层的所有G-skyline groups，层数从1到k；在生成每一层的G-skyline groups时，循环遍历所有上一层得到的的G-skyline groups，遍历每一个上一层的G-skyline group时，对tail set进行剪枝；然后遍历剪枝后的tail set，每次遍历得到一个candidate group，检查是否为G-skyline group，如果是，加入集合，否则直接舍弃。

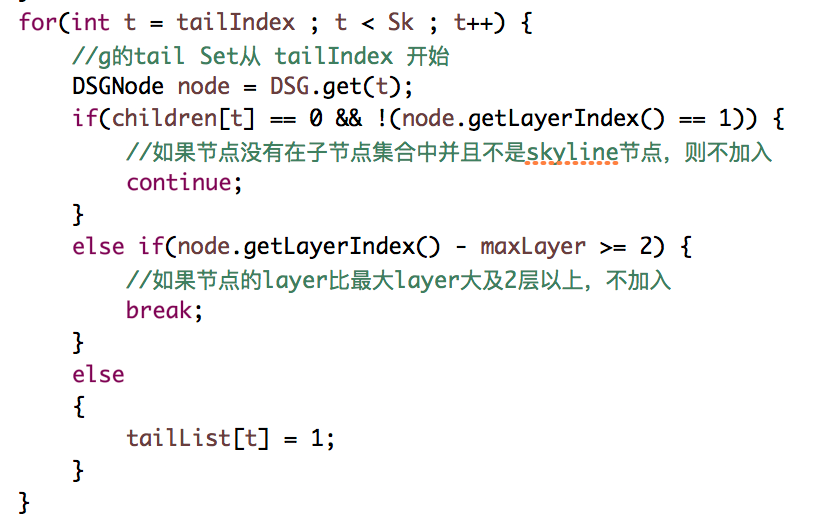
代码实现：

代码使用java实现。

1. 每一个G-skyline group使用List<Integer>表示，每个节点使用其index代表。最后得到的大小为k的G-skyline groups用List<List<Integer>> groupListNew表示。
2. 使用int[] children来统计每个G-skyline group的孩子集合，下标i中的整数为1代表index为i的节点是G-skyline group的孩子，为0代表不是孩子。具体代码实现如下，遍历G-skyline group的每个节点的孩子节点列表，将children数组对应于该孩子节点的下标index的整数设为1。



1. 使用int[] tailList来统计删减过后的尾集合。下标i中的整数为1代表index为i的节点是没有删减的节点。具体实现如下：



1. 遍历tailList，如果下标为i的位置整数为1，将index为i的节点加入G-skyline group，检查是否为新的G-skyline group，这里使用了一个检查是否为G-skyline group的检查方法：检查candidate group的unit group（自己的父母节点加上自身的集合）的大小是否等于candidate group自身的大小，这里使用java的数据结构Set／HashSet来去重检查。具体代码实现如下：



最后groupListNew中存储所有大小为k的G-skyline groups，groupListNew加上DSG预处理阶段得到的unit大小为k的节点集合perfectNodeList就是所有的解。