浅谈一类树分块的构建算法及其应用

周欣

杭州第二中学

2021年2月

内容概要

1. top cluster的概念，以及以之为基础的树分块算法
2. 一些例题，传统做法与新做法的对比

top cluster概念的介绍

1. **树簇（cluster）的定义**

   树簇是树上的一个连通子图，有至多两个点和全树的其他位置连接

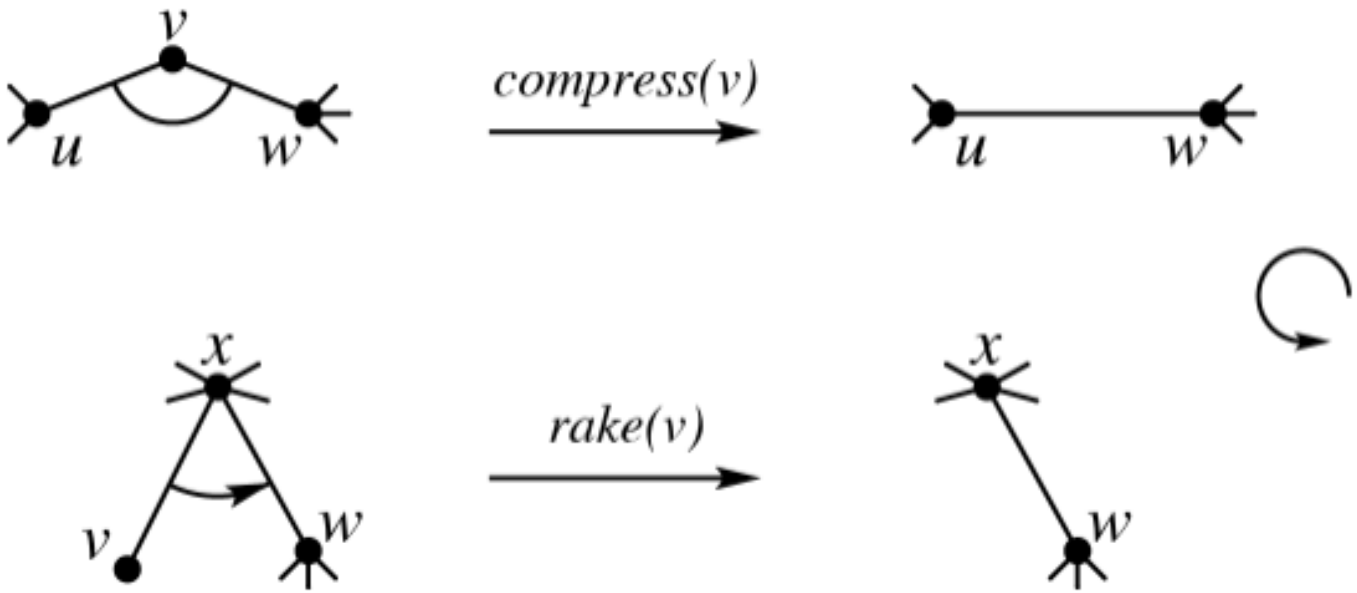
2. **簇的界点，内点与簇路径**

   一个簇与外界相接触的点被称做界点（boundary Node）

   簇中不是界点的点被称做内点（internal node）

   两个界点之间的路径被称做簇路径（cluster path）

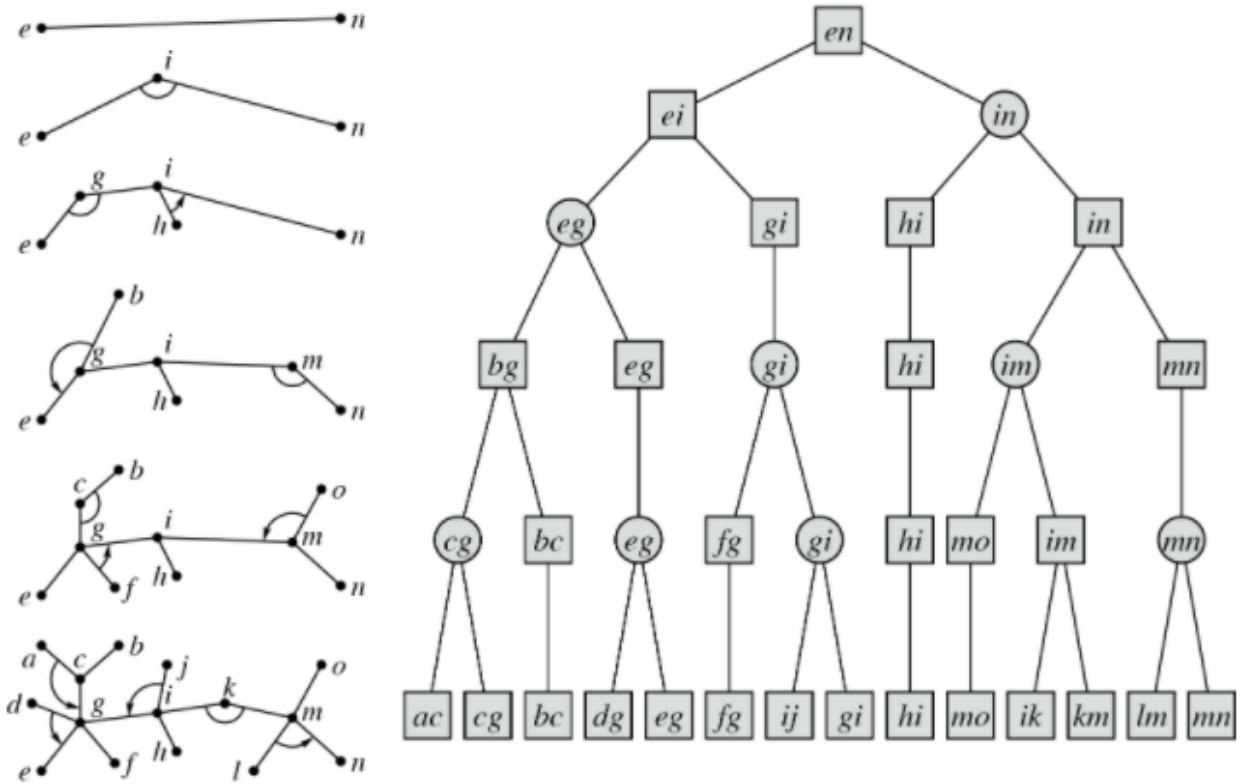
簇合并的两种基本方式



1. 第一种合并方式被命名为compress，效果为将一个二度点的相邻两簇合并为一个新簇
2. 第二种合并方式被命名为rake，效果为将一个一度点的相邻簇合并进入相邻点的另一个相邻簇

簇合并形成的二叉树结构

1. 通过不断执行簇合并的操作，可以将整棵树收缩为单个簇
2. 合并过程形成的二叉树结构被命名为top tree



序列分块带来的启示

序列上信息合并的基本单位是区间。序列分块可以看作是一种只有两层节点的特殊树形数据结构。

当我们将序列分块推广到树结构上的时候，自然的想法是，寻找一个与区间相对应的信息合并基本单位，以及将树表示为多层树形结构的基底数据结构，然后将节点强行拍扁成两层。

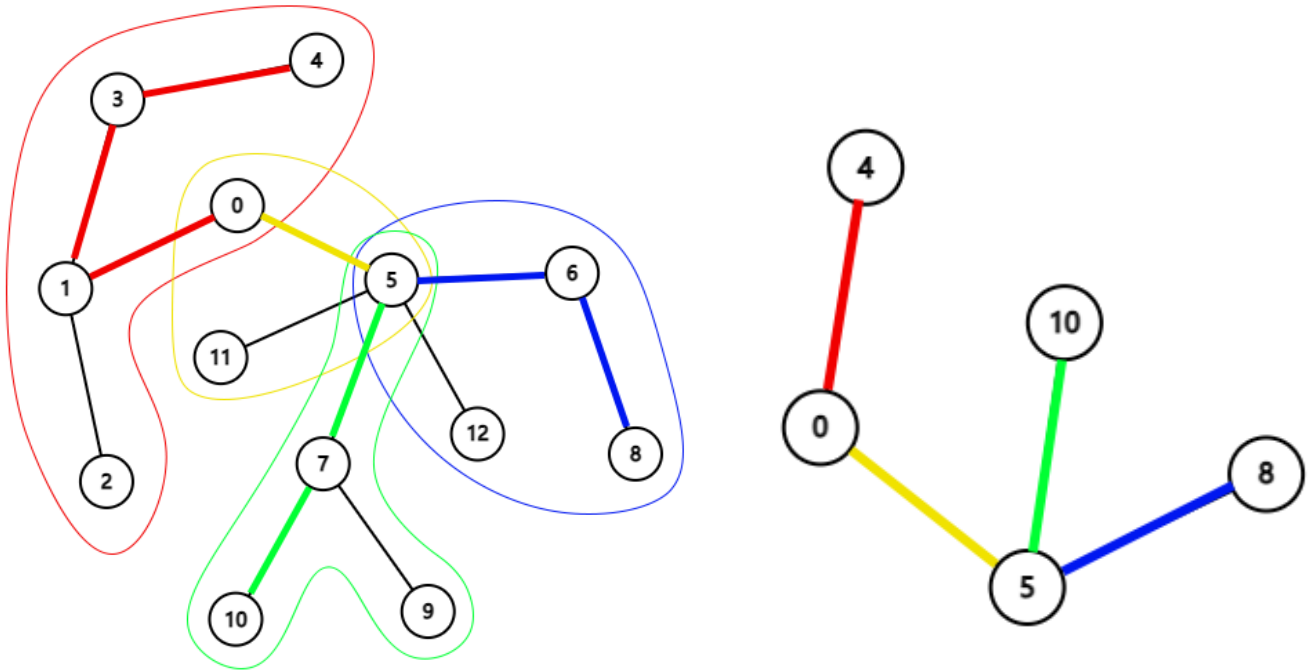
而上文中簇之于树就像区间之于序列，top tree的概念，恰好为我们提供了这样一种基底数据结构。

top cluster与树分块

我们借用top cluster的概念来精确地描述树分块构造算法所应当解决的问题： **对于一棵给定的 n 个点的树和一个块大小 B，要求将原树划分为 $O(\frac{n}{B})$ 个不交簇的并，并且每个簇的大小均为 $O(B)$。**

top cluster与树分块

如下左图即为对一棵树进行分块的示意



可以发现将每个簇看作一条边后，所有簇的界点构成另一个树的结构。我们称之为收缩树，如上右图。

树分块的构建与维护

对于静态树上的构建问题，存在一个 $O(n)$ 复杂度的算法。

算法的具体流程可以在本人的集训队论文中查阅，出于一些原因这里不详细展开了。

例题1

简要题意

动态维护关于加入时间的最大生成树，带点权修改，多次询问树上给定两点间路径k小值。

点数 n 和操作数 m 均不超过 10^5，可以认为 n, m 同阶。

例题1

传统做法

记块大小 $B = O(\sqrt{n})$。

对时间分块，在重构点处提前取出一段时间内修改涉及到的边和点，将其预先切掉，可以得到 $O(B)$ 个连通块的森林。

对每个连通块预处理树上主席树，查询时将涉及到的各个连通块的树链对应主席树，还有修改到的单点放在一起查询。

这个算法是可以强制在线的，首先树上主席树可以对重构时刻的整棵原树建立，而不是分连通块进行。之后用个LCT维护修改边对询问路径产生的划分即可。

最后总复杂度是 $O(n\sqrt{n}\log n)$。

例题2

题意

有一棵 n 个节点的有根树，每个节点上有点权 v_i。

支持五种操作：

1. 链加
2. 子树加
3. 链上第k大
4. 子树第k大
5. 修改一个点的父亲，保证修改后仍为一棵树

这里认为节点数与操作数同阶。

例题2

传统做法

由于要求支持对子树信息的修改与维护，这题是比上一题要强的。

首先，我们可以用LCT维护整棵树的一组链剖分，并对这组剖分所对应的实儿子先行得到的dfn序，使用用块状链表来维护。

~~块链ETT~~

子树操作就是块状链表上的区间操作，容易完成。

链操作和修改父亲操作通过LCT的access可以变为区间操作

最后总复杂度 $\tilde{O}(n\sqrt{n})$。

例题2

另解

首先我们以阈值 B 对原树分块，cut的时候暴力分裂所涉及的块，每 $O(\frac{n}{B})$ 次操作后重构整棵树的分块结构即可。

这样单次link/cut的复杂度是均摊 $O(B)$，但也存在做法可以保证单次最坏复杂度，感兴趣的您可以参阅我的集训队论文。

之后，关于点权信息的维护，在每个簇中，我们维护簇路径上的点权构成的有序数组，以及所有簇内点的点权构成的有序数组，还有修改操作带来的加法标记。

例题2

执行修改操作（同时包括点权修改和树形态修改）时。对于端点所处的簇，我们暴力重构的有序数组。为了避免复杂度增加 log 因子，我们可以使用归并来代替重新排序。这一步复杂度 $O(B)$。对于其它簇，只需修改加法标记。对于涉及到的界点，由于数量有上界，可以暴力处理。这一步复杂度 $O(\frac{n}{B})$。

执行查询操作时，对于端点所处的簇和界点，我们可以暴力提取出涉及到的点的点权。这样，问题就变成了，在多个有序数组中寻找第 k 小值。这可以做到 $O(\frac{n}{B}\log n)$ 的复杂度。

综上，取 $B = O(\sqrt{n\log n})$ 时本解法有最优复杂度 $O(n\sqrt{n\log n}))$。

小结

先前由于缺乏科学的树分块理论，大家处理相关问题时多是使用一些通用化的办法，比如对时间分块，或者将树转成序列后再用序列数据结构维护。

固然在特殊问题上使用通用方法具有一定借鉴意义，后面的例题中还会再次用到定期重构的思想。但对于树上相关问题，树分块确实更好地利用了问题特性，也更容易做到更简洁的思路与更优的复杂度。

例题3

给定一张 n 个点的图，每个点有点权，最开始没有边。之后需要进行 q 次操作，每次形如：

1. 添加一条x与y之间的双向边。
2. 回到第x次操作后的状态（注意这里的x可以是0，即回到初始状态）。
3. 查询x所在联通块中点权第y小的值。

保证 $n, q \leq 10^5$，可以认为 n, q 同阶。

时间限制 0.5s,空间限制19.53MB。

~~对不起我只是毒瘤了一点.jpg~~

例题3

传统做法1

操作3的存在需要我们保证单次最坏复杂度。

通过离散化的方法可以将点权值域压缩为 $O(n)$。

之后，最暴力的想法就是对每个连通块维护bitset，在操作树上边dfs边维护bitset即可。时间复杂度 $O(\frac{n^2}{w})$

为保证空间复杂度，bitset需要动态开点，通过启发式合并可以将空间复杂度降至 $O(n \log n)$。

例题3

传统做法2

首先我们在操作树上dfs，同时维护并查集用于判断每次加边操作是否有效（x和y是否属于同一个连通块）。由于需要支持回溯操作，所以并查集需要按秩合并，这一步复杂度 $O(n \log n)$。

之后我们设定阈值 $B = \sqrt{n}$，将值域切成 $\sqrt{n}$ 块，并对每组询问求出答案所属的值域块。

例题3

传统做法2

具体的，我们需要对每个值域块都在操作树上进行一次dfs，同时维护每个点所属连通块中，属于该值域块的点数。这一步复杂度为 $O(n\sqrt{n})$。

求出答案所属值域块后，我们还需要进一步求出答案的精确值。对此，在操作树的每个节点处我们只需从小到大枚举值域块中的元素，在并查集中查询其所属的连通块即可计算答案。

最后时间复杂度为 $O(n\sqrt{n}\log n)$，空间复杂度为 $O(n)$。时间复杂度中的 log 因子来自于按秩合并并查集的查询复杂度。

例题3

传统做法2的一些改进

我们的主要目标是规避按秩合并并查集。

一个想法是使用块链ETT维护每个时刻的生成森林，不过直接实现空间复杂度是 $O(n\sqrt{n})$的，难以承受。

尝试更换思路，我们以 $B = \sqrt{n}$ 为阈值将操作树进行分块。

对于每个界点，我们可以 $O(n)$ 处理出该时刻整张图的连通信息，再以该界点为基础回答相邻簇的询问。

例题3

对于相邻簇的所有内点，其到界点的路径上至多只有 $O(\sqrt{n})$ 次加边操作。

我们将这些加边操作涉及到的点取出来执行bfs，就可以以 $O(\sqrt{n})$ 复杂度得到该内点处的每个点所属的连通块，之后再从小到大枚举值域块中的元素计算答案即可。

最后时间复杂度为 $O(n\sqrt{n})$，空间复杂度为 $O(n)$，常数较小。

具体实现的时候，由于只需保证每个簇的直径而不是大小，可以使用随机选取 $O(\sqrt{n})$ 个界点的方法，减少代码量。

总结

先前大家对定期重构的认识大多停留在序列或时间轴的线性结构上。例题3中借助树分块的技术将定期重构推广到了树上，优化了复杂度与常数。

总的来说树分块相对于传统算法而言，时常具有一定优势。感兴趣的读者可以阅读我的集训队论文进行更深入的了解。

提问环节

欢迎大家提问

例题来源

例题1的命题报告可以在 https://immortalco.blog.uoj.ac/blog/670 查看。

例题2可以在ZJOI2019 Round1 范致远同学的讲课课件《数据结构　题选讲》中查看，题目名称为动态树练习题 V。

例题3可以在 https://www.luogu.com.cn/problem/P5064 查看。