# kruskal 重构树

xiaolilsq

HNFMS

2020 年 9 月 18 日

# 引入

**问题**

给定一张无向图，边有边权，每次询问 $u$ 到 $v$ 的所有路径中，边权最大的边最小的边权是多少。

# 引入

**问题**

给定一张无向图，边有边权，每次询问 $u$ 到 $v$ 的所有路径中，边权最大的边最小的边权是多少。

比较显然的做法就是建出最小生成树，然后树剖或者倍增求两点边权最大值。

# 引入

让我们来考虑另一种思路，在建出最小生成树后进行边分治，每次找出边权最大的边进行分治。
如果在某一次分治过程中，$u$ 和 $v$ 在断掉当前边后不再联通了，就说明 $u$ 和 $v$ 之间的边权最大值就是当前边的边权。

# 引入

让我们来考虑另一种思路，在建出最小生成树后进行边分治，每次找出边权最大的边进行分治。
如果在某一次分治过程中，$u$ 和 $v$ 在断掉当前边后不再联通了，就说明 $u$ 和 $v$ 之间的边权最大值就是当前边的边权。
具体实现我们可以不这样做，我们可以建出边分树，而 $u$ 和 $v$ 之间的边权最大值就是它们在边分树上面的 lca 。

# 引入

但是遗憾的是，我们并不能像平常建立边分树一样去建立这棵边分树，否则复杂度就无法保证。
考虑到在 kruskal 的过程中，我们是每次找出边权最小的边，然后合并两个连通块，这就是我们建立这棵边分树的逆过程。
所以我们可以在 kruskal 的过程中就顺便建立这棵边分树，每次合并两个连通块的时候就顺便记录一下儿子和父亲就行了，这样复杂度就得到了保证。

# 引入

事实上，这棵边分树，我们也把它叫做 kruskal 重构树。
是不是觉得 kruskal 重构树很简单，其实确实很简单。

# kruskal 重构树的性质

kruskal 重构树有很多优美的性质：

# kruskal 重构树的性质

kruskal 重构树有很多优美的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。

# kruskal 重构树的性质

kruskal 重构树有很多优美的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。

# kruskal 重构树的性质

kruskal 重构树有很多优美的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。
- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

# kruskal 重构树的用途

- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。

# kruskal 重构树的用途

- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。

在“引入”中提到的问题用 kruskal 重构树加上 ST 表查询 lca 就可以做到 $\mathcal{O}(nlog_2n)$ 预处理 $\mathcal{O}(1)$ 查询了。

# kruskal 重构树的用途

- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

# kruskal 重构树的用途

- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

利用这条性质我们可以很好地找到一个点 $x$ 仅经过边权小于等于 $y$ 的边能够到达的连通块。

# kruskal 重构树的用途

- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

利用这条性质我们可以很好地找到一个点 $x$ 仅经过边权小于等于 $y$ 的边能够到达的连通块。

首先我们可以从 $x$ 在重构树上对应的节点一直往上跳，直到权值要超过 $y$ 为止的时候停下来，那么到达的节点对应的这棵子树中的所有点 $x$ 都可以到达，正确性比较显然，而跳的过程可以用倍增实现。

一般我们会在重构树上的每一个节点记录一下整棵子树对应的答案，然后询问的时候就可以直接做了。

# kruskal 重构树的用途

对于这个问题：

**问题**

找到一个点 $x$ 仅经过边权小于等于 $y$ 的边能够到达的连通块。

我们还有一种解法，首先把所有询问离线下来，然后按边权从小到大依次加边，使用并查集维护连通块，同时维护答案，查询的时候就可以直接找到 $x$ 对应的连通块了。

# kruskal 重构树的用途

如果这个问题强制在线咋办？可以使用可持久化并查集和其他可持久化数据结构维护答案，但是这样的话复杂度多个 log 。

# kruskal 重构树的用途

如果这个问题强制在线咋办？可以使用可持久化并查集和其他可持久化数据结构维护答案，但是这样的话复杂度多个 log 。
事实上，可持久化并查集在这里大材小用了，因为这个问题是弱于可持久化并查集的模板题的，可持久化并查集有一个操作是在历史版本上面修改，而这个问题所需要的操作进行的修改永远是在当前版本上面进行的。

# kruskal 重构树的用途

如果这个问题强制在线咋办？可以使用可持久化并查集和其他可持久化数据结构维护答案，但是这样的话复杂度多个 log 。
事实上，可持久化并查集在这里大材小用了，因为这个问题是弱于可持久化并查集的模板题的，可持久化并查集有一个操作是在历史版本上面修改，而这个问题所需要的操作进行的修改永远是在当前版本上面进行的。
我们可不可以利用这个性质来降低时间复杂度呢？

# kruskal 重构树的用途

可持久化并查集的目的就是记录合并的全过程，我们可以考虑使用另外的方法来记录合并全过程。

# kruskal 重构树的用途

可持久化并查集的目的就是记录合并的全过程，我们可以考虑使用另外的方法来记录合并全过程。
在并查集每次合并的时候，我们并不直接将两个集合合并，而是增加一个节点，把这两个集合作为这个节点的儿子，我们再在这个节点上标上这次合并的权值（执行时间）。

# kruskal 重构树的用途

可持久化并查集的目的就是记录合并的全过程，我们可以考虑使用另外的方法来记录合并全过程。
在并查集每次合并的时候，我们并不直接将两个集合合并，而是增加一个节点，把这两个集合作为这个节点的儿子，我们再在这个节点上标上这次合并的权值（执行时间）。
最终会形成一棵树，那么每一个这样的节点对应的子树就记录下来每次合并的连通情况。

# kruskal 重构树的用途

可持久化并查集的目的就是记录合并的全过程，我们可以考虑使用另外的方法来记录合并全过程。
在并查集每次合并的时候，我们并不直接将两个集合合并，而是增加一个节点，把这两个集合作为这个节点的儿子，我们再在这个节点上标上这次合并的权值（执行时间）。
最终会形成一棵树，那么每一个这样的节点对应的子树就记录下来每次合并的连通情况。
其实我们最终建出来的就是 kruskal 重构树，这也是 kruskal 重构树的第二种理解：对于并查集每次合并的记录。

# 例题

## BZOJ 3388 & 洛谷 P4197 Peaks

在 Bytemountains 有 $n$ 座山峰，每座山峰有他的高度 $h_i$。有些山峰之间有双向道路相连，共 $m$ 条路径，每条路径有一个困难值，这个值越大表示越难走。

现在有 $q$ 组询问，每组询问询问从点 $v$ 开始只经过困难值小于等于 $x$ 的路径所能到达的山峰中第 $k$ 高的山峰，如果无解输出 $-1$ 。

# 例题

这道题目还是很基础的，首先建出 kruskal 重构树，每个节点上存上这棵子树里面所有点的权值的线段树，可以使用线段树合并来得到这些线段树。
询问直接倍增跳到对应节点询问就行了。

# 小结

kruskal 重构树的两种理解：

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。
- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。
- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

kruskal 重构树的用途：

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。
- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

kruskal 重构树的用途：

- 将求两点间的边权最大/最小值转化成 lca 问题。

# 小结

kruskal 重构树的两种理解：

- 按边权分治构建得到的边分树。
- 使用特殊方式记录并查集合并顺序构建得到的树。

kruskal 重构树的性质：

- 由于 kruskal 重构树是一棵边分树，所以 kruskal 重构树是一棵**完整二叉树**。
- 任意两点间的 lca 就是它们之间的路径中边权最大/最小的边。
- 除去叶子节点（也就是表示点而不是边的节点）外，kruskal 重构树满足堆的性质（即父亲节点边权小于等于/大于等于儿子节点边权）。

kruskal 重构树的用途：

- 将求两点间的边权最大/最小值转化成 lca 问题。
- 可以很好地找到一个点 $x$ 仅经过边权小于等于/大于等于 $y$ 的边能够到达的连通块。

# The end

Thank Thee!