有关Link-Cut Tree

hyj

雅礼中学

May 31, 2018

前言

Link-Cut Tree，LCT的全称

前言

Link-Cut Tree，LCT的全称

可以说是从树剖引出的问题

前言

Link-Cut Tree，LCT的全称

可以说是从树剖引出的问题

树剖可以解决静态的修改或查询树的链上信息；那如果树会不断改变，怎么办?

前言

Link-Cut Tree，LCT的全称

可以说是从树剖引出的问题

树剖可以解决静态的修改或查询树的链上信息；那如果树会不断改变，怎么办?

于是LCT应运而生（又是Tarjan和他的一个小伙伴发明的）

前言

Link-Cut Tree，LCT的全称

可以说是从树剖引出的问题

树剖可以解决静态的修改或查询树的链上信息；那如果树会不断改变，怎么办?

于是LCT应运而生（又是Tarjan和他的一个小伙伴发明的）

LCT要做的就是在不断的加边、删边等等改变树的操作中维护树的信息

链剖分——实链剖分

链剖分——实链剖分

每个节点，将它和它的某一个儿子之间的边划为实边，其余的边划为虚边，维护实链

链剖分——实链剖分

每个节点，将它和它的某一个儿子之间的边划为实边，其余的边划为虚边，维护实链

虚实会改变，所以要用splay

链剖分——实链剖分

每个节点，将它和它的某一个儿子之间的边划为实边，其余的边划为虚边，维护实链

虚实会改变，所以要用splay

于是LCT的大体结构就出来了，就是splay

链剖分——实链剖分

每个节点，将它和它的某一个儿子之间的边划为实边，其余的边划为虚边，维护实链

虚实会改变，所以要用splay

于是LCT的大体结构就出来了，就是splay

每个实链对应一个splay，用splay来维护它

性质

- 每个点包含于且仅包含于一个splay中

性质

- 每个点包含于且仅包含于一个splay中
- 同一深度的点不能在同一个splay中

性质

- 每个点包含于且仅包含于一个splay中
- 同一深度的点不能在同一个splay中
- 每个splay要满足中序遍历是按原树中的深度递增的

性质

- 每个点包含于且仅包含于一个splay中
- 同一深度的点不能在同一个splay中
- 每个splay要满足中序遍历是按原树中的深度递增的
- 每个节点与它实边相连的点在同一个splay中，实边包含在splay当中；虚边总是由一棵splay连向另一棵splay中的节点，连向的节点记录父亲，连出的节点不记录儿子，即子认父不认

Access

LCT的关键操作~~（好像没有中文名字）~~

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

实现：不停地进行以下步骤：

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

实现：不停地进行以下步骤：

1. 把当前点旋到所在splay的根

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

实现：不停地进行以下步骤：

1. 把当前点旋到所在splay的根
2. 把当前点的父亲的实边变成虚边

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

实现：不停地进行以下步骤：

1. 把当前点旋到所在splay的根
2. 把当前点的父亲的实边变成虚边
3. 把父亲连向自己的变变成实边

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

实现：不停地进行以下步骤：

1. 把当前点旋到所在splay的根
2. 把当前点的父亲的实边变成虚边
3. 把父亲连向自己的变变成实边
4. 更新维护的信息

Access

LCT的关键操作~~（好像没有中文名字）~~

作用：把一个点到根节点的路径上打通，经过的边全部变成实边，使得根与当前点在同一splay中，以维护当前点到根的链上的信息

实现：不停地进行以下步骤：

1. 把当前点旋到所在splay的根
2. 把当前点的父亲的实边变成虚边
3. 把父亲连向自己的变变成实边
4. 更新维护的信息
5. 跳到父亲

Makeroot

换根

Makeroot

换根

作用：把一个点换成原树的根

Makeroot

换根

作用：把一个点换成原树的根

实现：access当前点之后，由于LCT的性质，当前点只有左儿子，以保证它深度最大；翻转整棵splay，使得当前点只有右儿子，变成深度最小，即为根

Findroot

找根

Findroot

找根

作用：找一个点所在原树的根

Findroot

找根

作用：找一个点所在原树的根

实现：access当前点之后，利用LCT的性质，当前splay最左边的点深度最小，即根

Split

拉链

Split

拉链

作用：将两个点之间的路径放到一个splay中，以寻得信息

Split

拉链

作用：将两个点之间的路径放到一个splay中，以寻得信息

实现：将其中一个点makeroot成为原树的根，再将access另一个点

Link

连边

Link

连边

作用：连一条边 $x-y$

Link

连边

作用：连一条边 $x-y$

实现：首先要用findroot判两点是否已经在同一子树中，若是这样，再连边不合法；如若不然，将 x 变成原树的根之后，向 y 连一条轻边

Cut

断边

Cut

断边

作用：断开一条边 $x-y$

Cut

断边

作用：断开一条边 $x-y$

实现：判断两点是否真的存在一条边；若存在，直接切断

Pushup & Pushdown

两者就是splay中的pushup和pushdown

Pushup & Pushdown

两者就是splay中的pushup和pushdown

由于makeroot的存在，pushdown需要自带reverse的懒标记

维护连通性

按照题目link与cut

维护连通性

按照题目link与cut

查询两点连通性：findroot

维护连通性

按照题目link与cut

查询两点连通性：findroot

如果没有删边操作，那么可以直接用并查集维护。能用并查集尽量用并查集

BZOJ 2049

两种指令：

1. connect u v 出现一条u到v的边
2. destroy u v u到v的边消失
3. query u v 询问u和v当前是否连通

保证任意时刻任意两个洞穴之间至多只有一条路径

维护链上信息

要修改原树中u到v的路径上的信息

维护链上信息

要修改原树中u到v的路径上的信息

split(u,v)

维护链上信息

要修改原树中u到v的路径上的信息

split(u,v)

在splay中修改，利用pushdown的lazy完成所有点的改变

BZOJ 2243

给定一棵有n个节点的无根树和m个操作，操作有2类：

1、将节点a到节点b路径上所有点都染成颜色c；

2、询问节点a到节点b路径上的颜色段数量（连续相同颜色被认为是同一段），

如“112221”由3段组成：“11”、“222”和“1”。

请你写一个程序依次完成这m个操作。

只是高级一点的pushup

只是高级一点的pushup

考虑静态剖分用线段树pushup的时候怎么做

只是高级一点的pushup

考虑静态剖分用线段树pushup的时候怎么做

每个节点维护四个信息，col（结点本身颜色），lco（结点包括其Splay子树代表的一段颜色的最左边颜色），rco（这个就是最右边的），sum（结点包括其splay子树代表的一段的答案）

只是高级一点的pushup

考虑静态剖分用线段树pushup的时候怎么做

每个节点维护四个信息，col（结点本身颜色），lco（结点包括其Splay子树代表的一段颜色的最左边颜色），rco（这个就是最右边的），sum（结点包括其splay子树代表的一段的答案）

对于pushup，直接把两个儿子的sum与自己的长度1全部加上后，判断连接的地方是否相同，如果相同就减去多加的

维护生成树

维护一个图的最小或最大生成树

维护生成树

维护一个图的最小或最大生成树

化边为点

维护生成树

维护一个图的最小或最大生成树

化边为点

边建成一个新的点，点权为边的边权

维护生成树

维护一个图的最小或最大生成树

化边为点

边建成一个新的点，点权为边的边权

原图中一条边的连接，在新图中变为两条边的连接，原本的点——边化的新点——原本的点

维护生成树

维护一个图的最小或最大生成树

化边为点

边建成一个新的点，点权为边的边权

原图中一条边的连接，在新图中变为两条边的连接，原本的点——边化的新点——原本的点

但注意，LCT不能做到把在生成树中用非树边代替树边，即只能加边，不能删边

维护生成树

维护一个图的最小或最大生成树

化边为点

边建成一个新的点，点权为边的边权

原图中一条边的连接，在新图中变为两条边的连接，原本的点——边化的新点——原本的点

但注意，LCT不能做到把在生成树中用非树边代替树边，即只能加边，不能删边

时光倒流法

UOJ 274

http://uoj.ac/problem/274

维护边双

加边的时候，发现这条边的两端如果已经联通，那么这条边加入后就一定会形成一个环，那么这个环里的所有点就处在一个边双里

维护边双

加边的时候，发现这条边的两端如果已经联通，那么这条边加入后就一定会形成一个环，那么这个环里的所有点就处在一个边双里

用另一个点表示这个边双

维护边双

加边的时候，发现这条边的两端如果已经联通，那么这条边加入后就一定会形成一个环，那么这个环里的所有点就处在一个边双里

用另一个点表示这个边双

把这个环里的所有点的父亲指向新点

维护边双

加边的时候，发现这条边的两端如果已经联通，那么这条边加入后就一定会形成一个环，那么这个环里的所有点就处在一个边双里

用另一个点表示这个边双

把这个环里的所有点的父亲指向新点

类似缩点

BZOJ 2959

学校中有n个地点，用1到n的整数表示，每个地点设有若干个刷卡机。有以下三类事件：

1.修建了一条连接A地点和B地点的跑道。
2.A点的刷卡机台数变为了B。
3.进行了一次长跑。问从A出发，到达B最多可以刷卡多少次。

具体的要求如下： 到达一个地点时，可以在这里的每一台刷卡机上都刷卡。但每台刷卡机只能刷卡一次，即使多次到达同一地点也不能多次刷卡。边有向。最多的刷卡次数即为在任意设定跑道方向，按照任意路径从A地点到B地点能刷卡的最多次数。

维护子树信息

一个splay存的是一条实链，维护的只能是链上信息

维护子树信息

一个splay存的是一条实链，维护的只能是链上信息

如果要维护一个点的子树信息怎么办

维护子树信息

一个splay存的是一条实链，维护的只能是链上信息

如果要维护一个点的子树信息怎么办

注意LCT的构造，原树中一个节点的儿子（除实儿子）在LCT中一定是这个节点的虚儿子，那么如果要知道这个节点的原树信息，就是原树中的实儿子信息加虚儿子信息，实儿子信息LCT本身可以维护；那就只要知道虚儿子信息怎么维护。

维护子树信息

对于每个父亲与自己不在同一splay中的点，把自己的信息加入到自己存的父亲的虚儿子信息库

维护子树信息

对于每个父亲与自己不在同一splay中的点，把自己的信息加入到自己存的父亲的虚儿子信息库

如果维护信息是可减的，一个数组存，更新是 $O(1)$

维护子树信息

对于每个父亲与自己不在同一splay中的点，把自己的信息加入到自己存的父亲的虚儿子信息库

如果维护信息是可减的，一个数组存，更新是 $O(1)$

如果维护信息不可减（例如min,max），就需要堆或者其它数据结构维护，复杂度多一个log

维护子树信息

对于每个父亲与自己不在同一splay中的点，把自己的信息加入到自己存的父亲的虚儿子信息库

如果维护信息是可减的，一个数组存，更新是 $O(1)$

如果维护信息不可减（例如min,max），就需要堆或者其它数据结构维护，复杂度多一个log

相对于原来的LCT，发生一点改变的操作有access，link，pushup

BZOJ 4530

小强要在N个孤立的星球上建立起一套通信系统。这套通信系统就是连接N个点的一个树。

这个树的边是一条一条添加上去的。在某个时刻，一条边的负载就是它所在的当前能够联通的树上路过它的简单路径的数量。

现在，你的任务就是随着边的添加，动态的回答小强对于某些边的负载的询问。

接下来的题目会稍微有一点难度

你们肯定眼杀

BZOJ 3626

给出一个n个节点的有根树（编号为0到n-1，根节点为0）。一个点的深度定义为这个节点到根的距离+1。

设dep[i]表示点i的深度，LCA(i,j)表示i与j的最近公共祖先。

有q次询问，每次询问给出l r z，求sigma(l $\leq$ i $\leq$ r)dep[LCA(i,z)]。

（即，求在[l,r]区间内的每个节点i与z的最近公共祖先的深度之和）

共5组数据，n与q的规模分别为10000,20000,30000,40000,50000。

答案要求 $\sum_{i=l}^{r} dep[lca(i,z)]$

答案要求 $\sum_{i=l}^{r} dep[lca(i,z)]$

考虑单次询问如何得到答案，把 l 到 r 中所有点到根的路径上所有点的点权+1，答案变成 z 到根的路径上的权值和

答案要求 $\sum_{i=l}^{r} dep[lca(i,z)]$

考虑单次询问如何得到答案，把 l 到 r 中所有点到根的路径上所有点的点权+1，答案变成 z 到根的路径上的权值和

考虑多次询问，将询问按 l 排序。从 1 到 n 枚举树上的点加点权，如果有任意的询问的 $l-1$ 或 r 正好是当前枚举的点，记录下来

答案要求 $\sum_{i=l}^{r} dep[lca(i,z)]$

考虑单次询问如何得到答案，把 l 到 r 中所有点到根的路径上所有点的点权+1，答案变成 z 到根的路径上的权值和

考虑多次询问，将询问按 l 排序。从 1 到 n 枚举树上的点加点权，如果有任意的询问的 $l-1$ 或 r 正好是当前枚举的点，记录下来

最后利用查分思想，$ans[l,r] = ans[1,r] - ans[1,l-1]$

BZOJ 1977

给出一个n点m边的无向图，求出它的次小生成树，而且这个次小生成树还得是严格次小的

数据中无向图无自环；50% 的数据N⩽2 000 M⩽3 000；80% 的数据N⩽50 000 M⩽100 000； 100% 的数据N⩽100 000 M⩽300 000 ，边权值非负且不超过 1e9 。

首先肯定要求一个MST出来

首先肯定要求一个MST出来

考虑把剩下的边换入生成树，得到次小生成树

首先肯定要求一个MST出来

考虑把剩下的边换入生成树，得到次小生成树

显然，必定会换且只换一条边

首先肯定要求一个MST出来

考虑把剩下的边换入生成树，得到次小生成树

显然，必定会换且只换一条边

只需要枚举剩下的边，取最小的增量

首先肯定要求一个MST出来

考虑把剩下的边换入生成树，得到次小生成树

显然，必定会换且只换一条边

只需要枚举剩下的边，取最小的增量

但因为要是严格次小，所以增量不能为0，因此LCT中除维护一个Mx外，还要维护一个sMx，当枚举的边替换Mx的边增量为0时，不替换Mx，替换sMx

BZOJ 4817

Bob有一棵n个点的有根树，其中1号点是根节点。Bob在每个点上涂了颜色，并且每个点上的颜色不同。定义一条路径的权值是：这条路径上的点（包括起点和终点）共有多少种不同的颜色。Bob可能会进行这几种操作：

1 x: 把点x到根节点的路径上所有的点染上一种没有用过的新颜色。

2 x y: 求x到y的路径的权值。

3 x: 在以x为根的子树中选择一个点，使得这个点到根节点的路径权值最大，求最大权值。

Bob一共会进行m次操作

考虑查分，$ans = ans(u) + ans(v) - 2 * ans(lca(u, v)) + 1$

考虑查分，$ans = ans(u) + ans(v) - 2 * ans(lca(u, v)) + 1$

看修改操作，每次都是把一个点到根路径上的颜色都改成一种新的颜色，所以不可能存在 u 到 lca（lca 除外）的路径上与 v 到 lca（lca 除外）的路径上有相同的颜色，路径上也不可能出现颜色断层现象。所以只有两种情况：一是 lca 上的颜色不与任何它到 u，v 路径上的颜色相同，这样减去后，因为 lca 上是一种新的颜色，所以要加1；二是 lca 上的颜色与它到 u，v 的某一条路径上连着的一段颜色相同，这样减去之后这连着一段颜色相同的贡献多减了一次，所以还是要加1

第一个操作类似access，那么用access维护

第一个操作类似access，那么用access维护

每次access的时候，如果断开了右儿子的链，那么右儿子与当前点的颜色就不一样了，所以要给右儿子所在的子树的答案加1；而新接的右子树因为从颜色不一样变成了颜色一样，所以要给新的右子树的答案减1

第一个操作类似access，那么用access维护

每次access的时候，如果断开了右儿子的链，那么右儿子与当前点的颜色就不一样了，所以要给右儿子所在的子树的答案加1；而新接的右子树因为从颜色不一样变成了颜色一样，所以要给新的右子树的答案减1

既要使用access，又要维护子树最值，每个点的答案

第一个操作类似access，那么用access维护

每次access的时候，如果断开了右儿子的链，那么右儿子与当前点的颜色就不一样了，所以要给右儿子所在的子树的答案加1；而新接的右子树因为从颜色不一样变成了颜色一样，所以要给新的右子树的答案减1

既要使用access，又要维护子树最值，每个点的答案

树是静态！！！

第一个操作类似access，那么用access维护

每次access的时候，如果断开了右儿子的链，那么右儿子与当前点的颜色就不一样了，所以要给右儿子所在的子树的答案加1；而新接的右子树因为从颜色不一样变成了颜色一样，所以要给新的右子树的答案减1

既要使用access，又要维护子树最值，每个点的答案

树是静态！！！

LCT+树剖

UOJ 207

http://uoj.ac/problem/207

注意到，一条 x 到 y 路径被所有路径经过，当前仅当以 x 为根时，所有路径的端点都恰好有一个在 y 的子树中。

这样，我们对每一个路径随机一个权值，然后每次加入删除路径时，将两个段点的点权异或上这个权值，然后用动态树维护子树权值异或和，每次询问对应子树内的权值异或和是否是当前所有路径的权值异或和

洛谷 P4319

小 w 和小 c 在 H 国，近年来，随着 H 国的发展，H 国的道路也在不断变化着

根据 H 国的道路法，H 国道路都有一个值 w，表示如果小 w 和小 c 通过这条道路，那么他们的 L 值会减少 w，但是如果小 w 和 小 c 在之前已经经过了这条路，那么他们的 L 值不会减少

H 国有 N 个国家，最开始 H 国有 $N-1$ 条道路，这 $N-1$ 条道路刚好构成一棵树

小 w 将和小 c 从 H 国的城市 1 出发，游览 H 国的所有城市，总共游览 32766 天，对于每一天，他们都希望游览结束后 L 值还是一个正数，那么他们出发时 L 值至少为多少

考虑时间线段树，一条边在 l 到 r 中出现，就在线段树中 l 到 r 的区间加上这条边

考虑时间线段树，一条边在 l 到 r 中出现，就在线段树中 l 到 r 的区间加上这条边

最后访问线段树的每个叶子节点，然后往下递归的时候如果区间上有边的标记，就加边；到叶子节点的时候，算答案；回溯的时候，把在这个区间内加的边又删去。

洛谷 P3613

https://www.luogu.org/problemnew/show/P3613

起床困难综合症的升级版 采取同样的贪心策略，如果知道全0或全1跑完一遍运算后的结果，就可以知道最后答案

起床困难综合症的升级版 采取同样的贪心策略，如果知道全0或全1跑完一遍运算后的结果，就可以知道最后答案

维护这个全0或全1跑完一遍的结果

起床困难综合症的升级版 采取同样的贪心策略，如果知道全0或全1跑完一遍运算后的结果，就可以知道最后答案

维护这个全0或全1跑完一遍的结果

首先看一下直接维护我们要的东西可不可以，所以对于一个节点，维护 $r0$，$r1$ 分别代表全0跑过这个节点包含的一段运算的结果，以及全1跑完这一段运算的结果

起床困难综合症的升级版 采取同样的贪心策略，如果知道全0或全1跑完一遍运算后的结果，就可以知道最后答案

维护这个全0或全1跑完一遍的结果

首先看一下直接维护我们要的东西可不可以，所以对于一个节点，维护 $r0$，$r1$ 分别代表全0跑过这个节点包含的一段运算的结果，以及全1跑完这一段运算的结果

$$x_{r0} = (\sim lc(x)_{r0} \& rc(x)_{r0}) | (lc(x)_{r0} \& rc(x)_{r1})$$
$$x_{r1} = (\sim lc(x)_{r1} \& rc(x)_{r0}) | (lc(x)_{r1} \& rc(x)_{r1})$$

动态树——LCT

动态树——LCT

因为位运算是有序的，reverse的时候行不通

动态树——LCT

因为位运算是有序的，reverse的时候行不通

多维护一个反向运算的结果，reverse时的时候swap

动态树——LCT

因为位运算是有序的，reverse的时候行不通

多维护一个反向运算的结果，reverse时的时候swap

对于每次询问，拉链后贪心

BZOJ 3514

N个点M条边的无向图，询问保留图中编号在[l,r]的边的时候图中的联通块个数。

离线+强制在线

对于100%的数据，1⩽N、M、K⩽200,000。

依次考虑每一条边，如果加入这条边 i 会生成环，那就删除这个环里最早加入的边 j，并且记录下来 $fout[i] = j$，代表 i 的加入弹掉了 j 号边

依次考虑每一条边，如果加入这条边 i 会生成环，那就删除这个环里最早加入的边 j，并且记录下来 $fout[i]=j$，代表 i 的加入弹掉了 j 号边

然后发现对于一个询问 l,r，就看在 l 到 r 之间有多少条边 i，$fout[i]<l$，然后用 n 减掉这个数，就是一次询问的答案

依次考虑每一条边，如果加入这条边 i 会生成环，那就删除这个环里最早加入的边 j，并且记录下来 $fout[i] = j$，代表 i 的加入弹掉了 j 号边

然后发现对于一个询问 l, r，就看在 l 到 r 之间有多少条边 i，$fout[i] < l$，然后用 n 减掉这个数，就是一次询问的答案

解释：如果 i 边的 $fout$ 小于 l，那么如果只存在 l 到 r 的边话，i 边必定会连接上两个联通块，答案就要-1；反之，如果它的 $fout$ 大于等于 l，那么这条边连的是一个联通块里的两个点，不会对答案产生贡献

维护 $fout$ 数组，用LCT

查询 l 到 r 之间有多少 $fout$ 小于 l 的，用主席树

BZOJ 4573

https://www.lydsy.com/JudgeOnline/problem.php?id=4573

询问的操作永远可以放在修改操作之后，因为树的边只会增加，不会减少或变更

询问的操作永远可以放在修改操作之后，因为树的边只会增加，不会减少或变更

对于一棵树，直接考虑它的最终状态，然后处理询问

询问的操作永远可以放在修改操作之后，因为树的边只会增加，不会减少或变更
对于一棵树，直接考虑它的最终状态，然后处理询问

考虑只建一棵树，想办法在遍历的过程中快速从前面一棵树的形态变成接下来一棵树的形态

询问的操作永远可以放在修改操作之后，因为树的边只会增加，不会减少或变更
对于一棵树，直接考虑它的最终状态，然后处理询问

考虑只建一棵树，想办法在遍历的过程中快速从前面一棵树的形态变成接下来一棵树的形态

发现，如果先让所有的长出来的节点接在第一个生长节点（就是根）上，那么如果中间变更了生长节点，可以直接把在变更之后生长出的节点全部换根，变成新的生长节点的儿子。这个正好是新的形态。而对于 r 之后的，也就是没有变这个生长节点的树们，可以直接把换根过去的节点再换回来就变成应该的状态了

YY出一个“虚点”

YY出一个“虚点”

这个点可以用来当作它出现之后生长出的节点的根，换根的时候，直接带上这个虚点一起换，这样就只要改变两条边，就只是常数复杂度了

YY出一个“虚点”

这个点可以用来当作它出现之后生长出的节点的根，换根的时候，直接带上这个虚点一起换，这样就只要改变两条边，就只是常数复杂度了

建最开始的树

YY出一个“虚点”

这个点可以用来当作它出现之后生长出的节点的根，换根的时候，直接带上这个虚点一起换，这样就只要改变两条边，就只是常数复杂度了

建最开始的树

对于每次长节点，就是在最近一次出现的虚点下加一个点，点权为1

YY出一个“虚点”

这个点可以用来当作它出现之后生长出的节点的根，换根的时候，直接带上这个虚点一起换，这样就只要改变两条边，就只是常数复杂度了

建最开始的树

对于每次长节点，就是在最近一次出现的虚点下加一个点，点权为1

对于每次有要更改节点的操作，暂时不进行更改（因为有个 l 到 r 的范围嘛），只是在上一个虚点之下再建一个虚点，之后的实点都加在这个新的虚点之下，保证换根的优秀复杂度

YY出一个“虚点”

这个点可以用来当作它出现之后生长出的节点的根，换根的时候，直接带上这个虚点一起换，这样就只要改变两条边，就只是常数复杂度了

建最开始的树

对于每次长节点，就是在最近一次出现的虚点下加一个点，点权为1

对于每次有要更改节点的操作，暂时不进行更改（因为有个 l 到 r 的范围嘛），只是在上一个虚点之下再建一个虚点，之后的实点都加在这个新的虚点之下，保证换根的优秀复杂度

离线扫一遍

BZOJ 4025

神有一个n个节点的图。因为神是神，所以在T时间内一些边会出现后消失。神要求出每一时间段内这个图是否是二分图。

静态判能不能二分?

静态判能不能二分?

一条一条边加进去，如果这条边加上会变成一个环，并且这个环还是奇环，那就不能二分；反之，可以

静态判能不能二分?

一条一条边加进去，如果这条边加上会变成一个环，并且这个环还是奇环，那就不能二分；反之，可以

动态?

动态的麻烦就在删除，因为可能把一条树边删掉之后，会有一条非树边代替它变成一条树边，这个LCT是做不了的

LCT的用法改变一下

LCT的用法改变一下

维护这个图以删除时间为关键字的最大生成树，离线解决

BZOJ 1453

https://www.lydsy.com/JudgeOnline/problem.php?id=1453

不强制在线动态图问题，LCT

不强制在线动态图问题，LCT

对于四个方向，颜色相同的连边；那么每次翻转就变成了几次删边和几次加边（注意加边在删边之后）；联通块数量就变成了LCT维护的森林的数量

不强制在线动态图问题，LCT

对于四个方向，颜色相同的连边；那么每次翻转就变成了几次删边和几次加边（注意加边在删边之后）；联通块数量就变成了LCT维护的森林的数量

类似上一题，LCT维护以时间为权值的MST

不强制在线动态图问题，LCT

对于四个方向，颜色相同的连边；那么每次翻转就变成了几次删边和几次加边（注意加边在删边之后）；联通块数量就变成了LCT维护的森林的数量

类似上一题，LCT维护以时间为权值的MST

删掉了一条树边，就一定会把一棵树变成两棵树
连边，如果这条边的两端还没联通，那这条边就一定会把两棵树变成一棵树

Thanks