数据结构 for div1

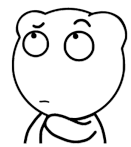
Newnode (2952643618@qq.com)

清华大学交叉信息研究院

January 22, 2019

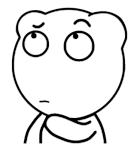
前言

树相比于一般的图有着很特殊的性质，是一种非常重要的数据结构，现在让我们看看算法竞赛中常用的对树的处理技巧。



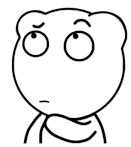
分治

分治法就是把一个问题拆成若干个规模更小的子问题分别解决，对于序列我们也常常分治，将一个区间拆成两半分块处理，那么对于树我们可以分治吗?



点分治

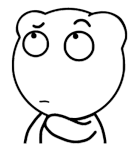
什么叫点分治？当然是选择一个点使其作为根，再递归处理其的每一棵子树了。



点分治

什么叫点分治？当然是选择一个点使其作为根，再递归处理其的每一棵子树了。

就像序列分治一样，我们希望将树尽量平分，使得递归层数尽可能小，那么问题就是，这个点如何选取呢？

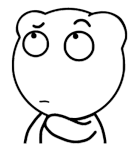


点分治

什么叫点分治？当然是选择一个点使其作为根，再递归处理其的每一棵子树了。

就像序列分治一样，我们希望将树尽量平分，使得递归层数尽可能小，那么问题就是，这个点如何选取呢？

贪心的来，当然是选择最大子树最小的点了！这样的点被称为“重心”，可以用一次树型dp求出该点。



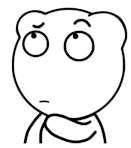
点分治

什么叫点分治？当然是选择一个点使其作为根，再递归处理其的每一棵子树了。

就像序列分治一样，我们希望将树尽量平分，使得递归层数尽可能小，那么问题就是，这个点如何选取呢？

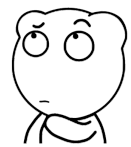
贪心的来，当然是选择最大子树最小的点了！这样的点被称为“重心”，可以用一次树型dp求出该点。

分析复杂度，一定存在一个节点使得最大子树不超过点数一半，那么层数至多$O(\log n)$层。



题1

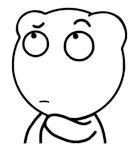
树上的点有黑色或白色，询问最长的路径使得经过的黑点数不超过k个。



题1

树上的点有黑色或白色，询问最长的路径使得经过的黑点数不超过k个。

如何点分治?

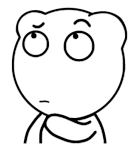


题1

树上的点有黑色或白色，询问最长的路径使得经过的黑点数不超过k个。

如何点分治?

每次考虑那些经过根的路径，剩下的可以递归下去做。这些路径一定是来自于两个不同子树的前缀链的并，用线段树维护即可，复杂度$O(n\log^2 n)$。



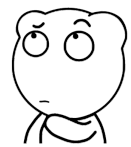
题1

树上的点有黑色或白色，询问最长的路径使得经过的黑点数不超过k个。

如何点分治?

每次考虑那些经过根的路径，剩下的可以递归下去做。这些路径一定是来自于两个不同子树的前缀链的并，用线段树维护即可，复杂度$O(n\log^2 n)$。

能快一点吗?



题1

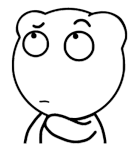
树上的点有黑色或白色，询问最长的路径使得经过的黑点数不超过k个。

如何点分治?

每次考虑那些经过根的路径，剩下的可以递归下去做。这些路径一定是来自于两个不同子树的前缀链的并，用线段树维护即可，复杂度$O(n\log^2 n)$。

能快一点吗?

two-pointer? $O(n^2)$?



题1

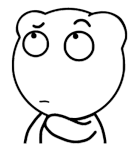
树上的点有黑色或白色，询问最长的路径使得经过的黑点数不超过k个。

如何点分治?

每次考虑那些经过根的路径，剩下的可以递归下去做。这些路径一定是来自于两个不同子树的前缀链的并，用线段树维护即可，复杂度$O(n\log^2 n)$。

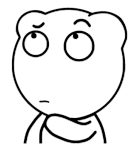
能快一点吗?

two-pointer? $O(n^2)$? 启发式合并!



边分治

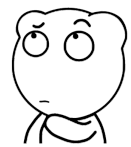
点分治之后会剩下许许多多的子树，不如序列分治分成两段舒服，那么树分治也能分成两块吗?



边分治

点分治之后会剩下许许多多的子树，不如序列分治分成两段舒服，那么树分治也能分成两块吗?

不妨选择一条边，就直接分成两棵子树了！那么这条边如何选取呢?

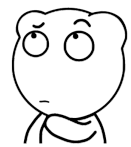


边分治

点分治之后会剩下许许多多的子树，不如序列分治分成两段舒服，那么树分治也能分成两块吗?

不妨选择一条边，就直接分成两棵子树了！那么这条边如何选取呢?

恭喜你得到了一个$O(n^2)$算法！



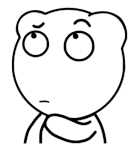
边分治

点分治之后会剩下许许多多的子树，不如序列分治分成两段舒服，那么树分治也能分成两块吗?

不妨选择一条边，就直接分成两棵子树了！那么这条边如何选取呢?

恭喜你得到了一个$O(n^2)$算法!

如果最大的点的度数为常数，可以证明复杂度为$O(n \log n)$。



边分治

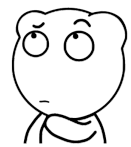
点分治之后会剩下许许多多的子树，不如序列分治分成两段舒服，那么树分治也能分成两块吗?

不妨选择一条边，就直接分成两棵子树了！那么这条边如何选取呢?

恭喜你得到了一个$O(n^2)$算法！

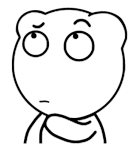
如果最大的点的度数为常数，可以证明复杂度为$O(n \log n)$。

加虚点！



题2

树上的点有黑色或白色，修改颜色或者询问最远白色点对距离。

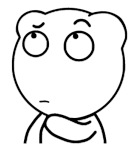


题2

树上的点有黑色或白色，修改颜色或者询问最远白色点对距离。
没有修改？直接点分治。

题2

树上的点有黑色或白色，修改颜色或者询问最远白色点对距离。
没有修改？直接点分治。
用堆维护每个子树的所有白点到根距离即可。



题3

树上的点有颜色，每次询问一个颜色区间中的点到某个给定点的距离和，每个点度数不超过3，要求在线。

题3

树上的点有颜色，每次询问一个颜色区间中的点到某个给定点的距离和，每个点度数不超过3，要求在线。

如果不是区间？点分治。

题3

树上的点有颜色，每次询问一个颜色区间中的点到某个给定点的距离和，每个点度数不超过3，要求在线。

如果不是区间？点分治。

按颜色排序存下来，复杂度$O(n \log^2 n)$。

题3

树上的点有颜色，每次询问一个颜色区间中的点到某个给定点的距离和，每个点度数不超过3，要求在线。

如果不是区间？点分治。

按颜色排序存下来，复杂度$O(n \log^2 n)$。

能再快一些吗？

题3

树上的点有颜色，每次询问一个颜色区间中的点到某个给定点的距离和，每个点度数不超过3，要求在线。

如果不是区间？点分治。

按颜色排序存下来，复杂度$O(n \log^2 n)$。

能再快一些吗？

可持久化，似乎边分治更好写？

题4

修改点权，最小化$\sum_v dist(u,v)val_v$，点的度数不超过20。

题4

修改点权，最小化$\sum_v dist(u,v)val_v$，点的度数不超过20。
其实就是求带权重心，点分治?

题4

修改点权，最小化$\sum_v dist(u,v)val_v$，点的度数不超过20。
其实就是求带权重心，点分治?
如果不在根一定在最大的孩子里，递归下去做咯。

题5

加点，维护$dist(u,v) \leq r_u + r_v$的点对数量。

题5

加点，维护$dist(u,v) \leq r_u + r_v$的点对数量。
没有加点，直接分治。

题5

加点，维护$dist(u,v) \leq r_u + r_v$的点对数量。
没有加点，直接分治。
加点使得分治结构不平衡？替罪羊思想，暴力重构。

题5

加点，维护$dist(u,v) \leq r_u + r_v$的点对数量。
没有加点，直接分治。
加点使得分治结构不平衡？替罪羊思想，暴力重构。
难写？900B即可。

询问点集

如果每次询问不是一个点，而是一个点集，点集总大小并不大，怎么做呢?

询问点集

如果每次询问不是一个点，而是一个点集，点集总大小并不大，怎么做呢?

对于这个点集来说，树中的大部分点都是无意义的，那么可以缩为虚树，点数不超过原点数的两倍。

询问点集

如果每次询问不是一个点，而是一个点集，点集总大小并不大，怎么做呢?

对于这个点集来说，树中的大部分点都是无意义的，那么可以缩为虚树，点数不超过原点数的两倍。

如何构建虚树？按dfs序依次加入，维护最右端的链即可。

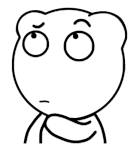
题6

给一张图，询问一个边集和点集，加上这些边之后这些点是否在同一个边双连通分量中，要求在线。

题6

给一张图，询问一个边集和点集，加上这些边之后这些点是否在同一个边双连通分量中，要求在线。

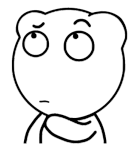
缩边双，虚树。



动态树问题

动态树问题和之前的问题有些不同，它一般要求你支持如下操作：

1. 加边或者删边；
2. 对树上一条路径的修改或者询问；
3. 对一个子树的修改或者询问。



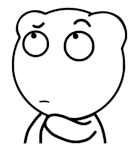
静态树问题

很多试题中树的形态都不会变化，只要你对路径或子树进行修改或询问，这类问题我们可以称为静态树问题。

静态树问题

很多试题中树的形态都不会变化，只要你对路径或子树进行修改或询问，这类问题我们可以称为静态树问题。

解决这类问题的方法一般是轻重链剖分。



子树操作

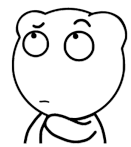
如果只存在子树操作，怎么办?

子树操作

如果只存在子树操作，怎么办?
dfs序，每个子树都是一个区间。

路径操作

如果存在路径操作，怎么办?

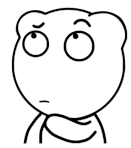


路径操作

如果存在路径操作，怎么办?

实际上，dfs序是由某些路径组合起来的，某个点的第一个被访问的孩子会和它组合成一条路径。

也就是说这棵树被剖成了许多路径，一条路径就可以分解为若干条这些路径的子段，怎么剖最优?



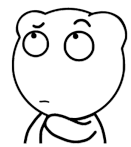
路径操作

如果存在路径操作，怎么办?

实际上，dfs序是由某些路径组合起来的，某个点的第一个被访问的孩子会和它组合成一条路径。

也就是说这棵树被剖成了许多路径，一条路径就可以分解为若干条这些路径的子段，怎么剖最优?

怎么分呢? 随机一波?



路径操作

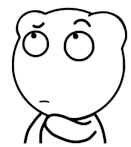
如果存在路径操作，怎么办?

实际上，dfs序是由某些路径组合起来的，某个点的第一个被访问的孩子会和它组合成一条路径。

也就是说这棵树被剖成了许多路径，一条路径就可以分解为若干条这些路径的子段，怎么剖最优?

怎么分呢? 随机一波?

按照较大的子树剖就好了。



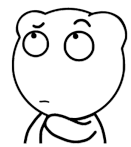
题7

一棵n个点的带边权的基环树，环的大小是奇数，m次操作，将一条最短路径取反或者询问最短路径的最大子段和。

题7

一棵n个点的带边权的基环树，环的大小是奇数，m次操作，将一条最短路径取反或者询问最短路径的最大子段和。

是树很好处理，基环树我们就断掉环上一条边，再判断一下就好了。



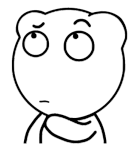
题8

树上点是黑或白色，询问某个点所在同色连通块大小或者修改某个点的颜色。

题8

树上点是黑或白色，询问某个点所在同色连通块大小或者修改某个点的颜色。

用$f[u][0/1]$记录子树中黑白色连通块大小，修改只会修改一条链。



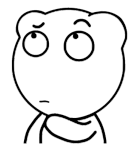
题9

一个交互请求存在于一条路径上，有一个优先级，每次加入或者删除一个请求，或者询问删去某个点还存在的最高优先级。

题9

一个交互请求存在于一条路径上，有一个优先级，每次加入或者删除一个请求，或者询问删去某个点还存在的最高优先级。

一条路径会被剖成$O(\log n)$个区间，那么补集也只会有$O(\log n)$个，于是就转化为了区间的问题。删除不好处理可以离线分治。



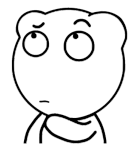
题10

在序列末尾加入或删除一个向量，询问区间中的向量和给定向量的最大叉积。用树剖可做吗?

题10

在序列末尾加入或删除一个向量，询问区间中的向量和给定向量的最大叉积。用树剖可做吗?

建出操作树，于是变成询问一条链中的最大叉积。最大叉积一定是在凸包上。

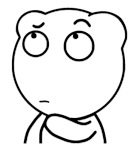


题10

在序列末尾加入或删除一个向量，询问区间中的向量和给定向量的最大叉积。用树剖可做吗?

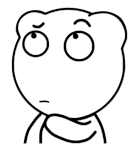
建出操作树，于是变成询问一条链中的最大叉积。最大叉积一定是在凸包上。

剖分，除了最上面的一个区间，其余都是某条重链的前缀！对于最上面区间分治求解，其余部分可以用平衡树动态维护凸包。



动态树问题

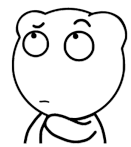
在更加难的题中，存在一些加边和删边操作，并要求你对路径或子树进行修改或询问，这类问题我们可以称为动态树问题。



动态树问题

在更加难的题中，存在一些加边和删边操作，并要求你对路径或子树进行修改或询问，这类问题我们可以称为动态树问题。

为了解决这类问题我们需要一些动态树算法，常用的算法有Link-Cut Tree，Euler Tour Tree和Self-Adjusting Top Tree。

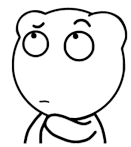


Link-Cut Tree

LCT的主要思想就是动态维护剖分，所以对于路径操作它是一个非常有效的算法。

一些定义

树的结构在变，不能再用原来的方法划分轻边重链了，下面对LCT的一些名词给出定义。



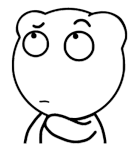
一些定义

树的结构在变，不能再用原来的方法划分轻边重链了，下面对LCT的一些名词给出定义。

Preferred Child:偏爱子节点，一个点x的偏爱子节点y满足，最后一次访问x子树中的点是在y的子树中。

Preferred Edge:偏爱边，连接点和偏爱子节点的边就是偏爱边。

Preferred Path:偏爱路径，一条极长的仅由偏爱边组成的路径称为偏爱路径。



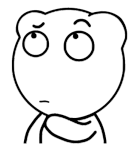
Access

LCT的最核心操作就是访问一个节点（Access）。根据之前的定义，如果访问一个节点，会将它到根的路径变成一条偏爱路径，这样就很容易进行路径操作。

Access

LCT的最核心操作就是访问一个节点（Access）。根据之前的定义，如果访问一个节点，会将它到根的路径变成一条偏爱路径，这样就很容易进行路径操作。

如何维护LCT，也就是如何维护偏爱路径。偏爱路径会不断改变，那就只能用Splay。

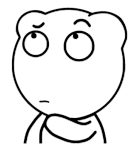


Access

LCT的最核心操作就是访问一个节点（Access）。根据之前的定义，如果访问一个节点，会将它到根的路径变成一条偏爱路径，这样就很容易进行路径操作。

如何维护LCT，也就是如何维护偏爱路径。偏爱路径会不断改变，那就只能用Splay。

对每条偏爱路径都用一个Splay来维护，在Splay的根处记录上这条偏爱路径的父亲，这样就很容易进行Access操作了。



Makeroot

如何换根？只需要Access以后将Splay完全翻转就好。

Link

如果加边(x,y)，就先Makeroot(y)，然后将这条偏爱路径的父亲记作x。

Cut

如果删边(x,y)，就先Makeroot(x)，再Access(y)，然后切断。

时间复杂度

所有的操作都需要Access，其余都是$O(1)$次Splay的操作，所以只要分析Access的复杂度就好了。

时间复杂度

所有的操作都需要Access，其余都是$O(1)$次Splay的操作，所以只要分析Access的复杂度就好了。

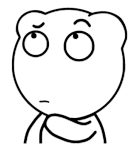
这里给出一个简单的证明，证明偏爱边的切换是均摊$O(\log n)$的。

时间复杂度

所有的操作都需要Access，其余都是$O(1)$次Splay的操作，所以只要分析Access的复杂度就好了。

这里给出一个简单的证明，证明偏爱边的切换是均摊$O(\log n)$的。

先轻重链剖分，一次Access只会将$O(\log n)$条轻边切换成偏爱边，而重边则可能非常多。但是将重边切换为偏爱边和将重边切换为非偏爱边的级别相同，而将重边切换为非偏爱边肯定会有轻边切换为偏爱边，那么切换次数就均摊$O(\log n)$了。



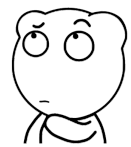
题11

有n个未知数和方程，每个都是$x_i = k_i x_{p_i} + b_i \mod 10007$，询问某个$x$的解或者修改一个方程。

题11

有n个未知数和方程，每个都是$x_i = k_i x_{p_i} + b_i \mod 10007$，询问某个$x$的解或者修改一个方程。

一个基环森林，对每个基环树都删去一条边变成树，在根处记录一下就好。



题12

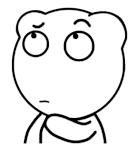
一个无向图，加边删边询问两点是否连通，可以离线。

题12

一个无向图，加边删边询问两点是否连通，可以离线。
每条边存在时间是一个区间，分治+并查集，复杂度$O(n\log^2 n)$。

题12

一个无向图，加边删边询问两点是否连通，可以离线。
每条边存在时间是一个区间，分治+并查集，复杂度$O(n \log^2 n)$。
直接用LCT维护删除时间的最大生成树，边权如何转化为点权?



题12

一个无向图，加边删边询问两点是否连通，可以离线。
每条边存在时间是一个区间，分治+并查集，复杂度$O(n\log^2 n)$。
直接用LCT维护删除时间的最大生成树，边权如何转化为点权?
加入虚点就好了。复杂度$O(n\log n)$。

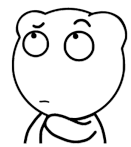
题13

每个点最多属于一个简单环，边有边权，m次操作，询问两个点的最大流或者修改边权。

题13

每个点最多属于一个简单环，边有边权，m次操作，询问两个点的最大流或者修改边权。

最大流就是最小割，只可能割环上两条边或者一条环间的边，这两条边一定会包含最小的那一条，割掉它并把它的边权加到其它边上。



题14

换父亲和修改点权，问子树最大点权，能用LCT做吗?

题14

换父亲和修改点权，问子树最大点权，能用LCT做吗?

对每个节点维护一个堆，把每个轻儿子（非偏爱子节点）的子树最大点权扔进去，重儿子部分用Splay维护一下。

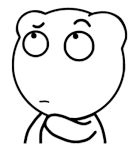
子树操作

路径操作可以使用LCT，那么对于子树操作该怎么办呢?

子树操作

路径操作可以使用LCT，那么对于子树操作该怎么办呢?

在静态树上子树操作可以通过dfs序列来解决的，能否动态维护dfs序?



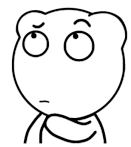
括号序列

如果没有换根，加入删除点只是在dfs序中插入删除点，同样子树也就是插入或删除区间，用Splay或者Treap维护dfs序就好。

括号序列

如果没有换根，加入删除点只是在dfs序中插入删除点，同样子树也就是插入或删除区间，用Splay或者Treap维护dfs序就好。

为了方便找到子树位置，可以将一个点拆成两个构成括号序列，中间就是其子树。



题15

修改子树点权，加点，删子树或者询问子树权值和。

题15

修改子树点权，加点，删子树或者询问子树权值和。
直接使用Splay维护dfs序就好啦！

题16

每个点的儿子有顺序。询问两个点的距离，或者将一棵子树接到它的k层祖先上，成为它的最后一个儿子，或者询问从一个点开始dfs，第k层的最后一个点是什么。

题16

每个点的儿子有顺序。询问两个点的距离，或者将一棵子树接到它的k层祖先上，成为它的最后一个儿子，或者询问从一个点开始dfs，第k层的最后一个点是什么。

dfs序中任意两个相邻的点深度只会差1，记录区间深度最大值和最小值，就可以找到深度为k的最后一个点以及某个点的k层祖先。

题16

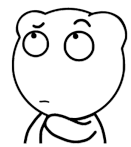
每个点的儿子有顺序。询问两个点的距离，或者将一棵子树接到它的k层祖先上，成为它的最后一个儿子，或者询问从一个点开始dfs，第k层的最后一个点是什么。

dfs序中任意两个相邻的点深度只会差1，记录区间深度最大值和最小值，就可以找到深度为k的最后一个点以及某个点的k层祖先。

而LCA即为两个点之间深度最小点的父亲。

子树操作+路径操作

对不起打扰了，告辞。



Thank you!