树分治及其应用

hzwer

PekingUniversity

2016 年 7 月 22 日

1. 点分治
   - 定义
   - 构建
   - 动态点分治
2. 树链剖分

分治，指的是分而治之

分治，指的是分而治之
即将一个问题分割成一些规模较小的相互独立的子问题，以便各个击破

分治，指的是分而治之

即将一个问题分割成一些规模较小的相互独立的子问题，以便各个击破

我们常见的是在一个线性结构上进行分治

分治，指的是分而治之
即将一个问题分割成一些规模较小的相互独立的子问题，以便各个击破
我们常见的是在一个线性结构上进行分治
分治往往与高效联系在一起，而树的分治正是一种用来解决树的路径问题的高效算法

点分治

首先选取一个点将无根树转为有根树，再递归处理每一棵以根结点的儿子为根的子树

点分治
首先选取一个点将无根树转为有根树，再递归处理每一棵以根结点的儿子为根的子树
边分治
在树中选取一条边，将原树分成两棵不相交的树，递归处理

1. 点分治
   - 定义
   - 构建
   - 动态点分治
2. 树链剖分

对于基于点的分治，我们选取一个点，要求将其删去后，结点最多的树的结点个数最小

对于基于点的分治，我们选取一个点，要求将其删去后，结点最多的树的结点个数最小

树的重心可以通过动态规划来解决

对于基于点的分治，我们选取一个点，要求将其删去后，结点最多的树的结点个数最小

树的重心可以通过动态规划来解决

可以证明，在基于点的分治中每次我们都会将树的结点个数减少一半

因此递归深度最坏是 $O(logn)$ 的，在树是一条链的时候达到上界

对于基于点的分治，我们选取一个点，要求将其删去后，结点最多的树的结点个数最小

树的重心可以通过动态规划来解决

可以证明，在基于点的分治中每次我们都会将树的结点个数减少一半

因此递归深度最坏是 $O(logn)$ 的，在树是一条链的时候达到上界

在基于边的分治中，若树的度数为常数，则递归深度最坏是 $O(logn)$ 的

但树的度数可能达到 $O(n)$，此时边分治效率极低

例题 1 POJ1741

给定一棵 n 个点的边权树，问树中 $\leq K$ 的路径数
其中 $0 \leq n \leq 10^5, 0 \leq K \leq 10^9$

题解

如果使用普通的 DFS，时间复杂度高达 $O(n^2)$
使用时间复杂度为 $O(nk)$ 的动态规划，更无法在规定时限内出解

题解

如果使用普通的 DFS，时间复杂度高达 $O(n^2)$

使用时间复杂度为 $O(nk)$ 的动态规划，更无法在规定时限内出解

树上的一条路径要么过根结点，要么在一棵子树中

这启发了我们可以使用分治算法

题解

如果使用普通的 DFS，时间复杂度高达 $O(n^2)$

使用时间复杂度为 $O(nk)$ 的动态规划，更无法在规定时限内出解

树上的一条路径要么过根结点，要么在一棵子树中

这启发了我们可以使用分治算法

路径在子树中的情况只需递归处理即可，只要分析如何处理路径过根结点的情况

题解

可以对根延伸出的几棵子树各做一次 dfs

题解

可以对根延伸出的几棵子树各做一次 dfs
记录所有结点的距离值并排序
利用单调性我们很容易得出一个 $O(n)$ 的算法，所以可以在 $O(nlogn)$ 的时间内统计答案
要注意排除掉不合法的答案数

题解

可以对根延伸出的几棵子树各做一次 dfs
记录所有结点的距离值并排序
利用单调性我们很容易得出一个 $O(n)$ 的算法，所以可以在 $O(nlogn)$ 的时间内统计答案
要注意排除掉不合法的答案数
综上，此题使用树的分治算法时间复杂度为 $O(nlog^2n)$

例题 2 BZOJ2599

给定一棵 n 个点的边权树，求长度为 3 的倍数的路径数
其中 $0 \le n \le 2 * 10^5$

题解

同样只需要考虑过一个根的路径

题解

同样只需要考虑过一个根的路径

对于每个重心统计出每棵子树距离重心长度为 0/1/2 的点的数量，计算出 ans 即可

题解

同样只需要考虑过一个根的路径
对于每个重心统计出每棵子树距离重心长度为 0/1/2 的点的数量，计算出 ans 即可
其实直接树形 dp 就可以了。。。

题解

同样只需要考虑过一个根的路径
对于每个重心统计出每棵子树距离重心长度为 0/1/2 的点的数量，计算出 ans 即可
其实直接树形 dp 就可以了。。。

例题 3 BZOJ2599

给定一棵 n 个点的边权树，问树中权值和为 K 的路径包含的最少边数
其中 $0 \le n \le 2*10^5, 0 \le K \le 10^6$

题解

同样只需要考虑过一个根的路径

题解

同样只需要考虑过一个根的路径
开一个 10^6 的数组 t，t[i] 表示权值为 i 的路径最少边数

题解

同样只需要考虑过一个根的路径

开一个 10^6 的数组 t，t[i] 表示权值为 i 的路径最少边数

得出一棵子树的所有点到根路径，设其权值和 x，经过了 a 条边，用 $t[k-x]+a$ 更新答案

题解

同样只需要考虑过一个根的路径
开一个 10^6 的数组 t，t[i] 表示权值为 i 的路径最少边数
得出一棵子树的所有点到根路径，设其权值和 x，经过了 a 条边，用 $t[k-x]+a$ 更新答案
再重新更新一遍 t[x]

例题 4 FreeTour2

给定一棵 n 个点的边权树，点有黑白两类
求一条不超过 K 个黑点的路径，且路径长度最大
其中 $0 \leq n \leq 2 * 10^5$

题解

同样只需要考虑过一个根的路径

题解

同样只需要考虑过一个根的路径

设路径起点为根，终点为 x 的路径，经过黑色结点数量为 dep[x]，路径长度为 dis[x]

题解

同样只需要考虑过一个根的路径

设路径起点为根，终点为 x 的路径，经过黑色结点数量为 dep[x]，路径长度为 dis[x]

依次处理 root 的每棵子树, 处理到子树 S 的时

我们需要知道出发点为根，终点在前 S-1 棵子树中，经过 t(t<K) 个黑点的路径的最长长度 mx[t]

题解

同样只需要考虑过一个根的路径

设路径起点为根，终点为 x 的路径，经过黑色结点数量为 dep[x]，路径长度为 dis[x]

依次处理 root 的每棵子树, 处理到子树 S 的时

我们需要知道出发点为根，终点在前 S-1 棵子树中，经过 t(t<K) 个黑点的路径的最长长度 mx[t]

则对于子树 S 的结点 x

$mx[t] + dis[x](dep[x] + t \leq K) \xrightarrow{update} ans$

(若根为黑色处理时将 K–1，处理完恢复)

题解

若按照 dep 倒序处理每个结点，mx 指针 now 按照升序扫，则

$mx[now-1] \xrightarrow{update} mx[now]$

这样就能得到符合条件的 mx[t] 的最大值

题解

若按照 dep 倒序处理每个结点，mx 指针 now 按照升序扫，则
$mx[now-1] \xrightarrow{update} mx[now]$
这样就能得到符合条件的 mx[t] 的最大值
然后再考虑用子树 S 的信息更新 mx，将两个数组合并的操作次数是 $max(L1, L2)$
可以考虑按照每棵子树 dep 的升序依次合并子树

题解

若按照 dep 倒序处理每个结点，mx 指针 now 按照升序扫，则 $mx[now-1] \xrightarrow{update} mx[now]$
这样就能得到符合条件的 mx[t] 的最大值
然后再考虑用子树 S 的信息更新 mx，将两个数组合并的操作次数是 $max(L1, L2)$
可以考虑按照每棵子树 dep 的升序依次合并子树
从总体来看，由于总边数是 n-1，则排序的复杂度 $O(nlogn)$，同时这样启发式合并使得合并的复杂度降为 $O(nlogn)$

例题 5 BZOJ3784

给定一棵 n 个点的边权树，输出树上前 m 大的路径长度

其中 $0 \le n \le 5 * 10^4, 0 \le m \le 3 * 10^5$

解法 1

二分第 m 大的路径长度，就直接得到了一个经典的点分治问题，可以在 $O(nlog^2n)$ 的复杂度解决

解法 1

二分第 m 大的路径长度，就直接得到了一个经典的点分治问题，可以在 $O(nlog^2n)$ 的复杂度解决
整个算法的复杂度 $O(nlog^3n)$, 但是有一个 log 是单次点分治时 sort 需要的

解法 1

二分第 m 大的路径长度，就直接得到了一个经典的点分治问题，可以在 $O(nlog^2n)$ 的复杂度解决

整个算法的复杂度 $O(nlog^3n)$，但是有一个 log 是单次点分治时 sort 需要的

可以在一次点分治时把 sort 的结果存下来，所需的空间复杂度为 $O(nlogn)$

解法 1

二分第 m 大的路径长度，就直接得到了一个经典的点分治问题，可以在 $O(nlog^2n)$ 的复杂度解决

整个算法的复杂度 $O(nlog^3n)$，但是有一个 log 是单次点分治时 sort 需要的

可以在一次点分治时把 sort 的结果存下来，所需的空间复杂度为 $O(nlogn)$

得到 m 大的路径最后一次点分治暴力统计路径

解法 2

若将当前子树内的点作为路径的一个端点，另一个端点可以落在一个点分治序列的区间内（之前扫过的子树）
这样得出一个长度为 nlogn 的点分治序列，加上每个点所对应的区间，然后就完全转为 NOI2010 超级钢琴

1. 点分治
   - 定义
   - 构建
   - 动态点分治
2. 树链剖分

定义

如果我们把每次分治的中心连边成一棵树的话，这棵树的深度是 logn 的！

定义

如果我们把每次分治的中心连边成一棵树的话，这棵树的深度是 logn 的！
由于我们对于每个中心处理过这个中心的路径，如果我们对于某个结点信息进行修改的话，影响的中心是它到根的一条 logn 长的链！

定义

如果我们把每次分治的中心连边成一棵树的话，这棵树的深度是 logn 的！
由于我们对于每个中心处理过这个中心的路径，如果我们对于某个结点信息进行修改的话，影响的中心是它到根的一条 logn 长的链！
于是很多点分治问题可以支持修改辣

树链剖分就是把树拆成一系列链，然后用数据结构对链进行维护

树链剖分就是把树拆成一系列链，然后用数据结构对链进行维护
通常的剖分方法是轻重链剖分，所谓轻重链就是对于节点 u 的所有子结点 v，$size[v]$ 最大的 v 与 u 的边是重边，其它边是轻边
其中 $size[v]$ 是以 v 为根的子树的节点个数，全部由重边组成的路径是重路径

树链剖分就是把树拆成一系列链，然后用数据结构对链进行维护
通常的剖分方法是轻重链剖分，所谓轻重链就是对于节点 u 的所有子结点 v，$size[v]$ 最大的 v 与 u 的边是重边，其它边是轻边
其中 $size[v]$ 是以 v 为根的子树的节点个数，全部由重边组成的路径是重路径
根据论文上的证明，任意一点到根的路径上存在不超过 logn 条轻边和 logn 条重路径

这样我们考虑用数据结构来维护重路径上的查询，轻边直接查询

这样我们考虑用数据结构来维护重路径上的查询，轻边直接查询
通常用来维护的数据结构是线段树，splay 较少见

第一遍 dfs

第一遍 dfs

求出树每个结点的深度 $dep[x]$，其为根的子树大小 $size[x]$

第二遍 dfs

第二遍 dfs
以根节点为起点，向下拓展构建重链

第二遍 dfs
以根节点为起点，向下拓展构建重链
选择最大的一个子树的根继承当前重链

第二遍 dfs
以根节点为起点，向下拓展构建重链
选择最大的一个子树的根继承当前重链
其余节点，都以该节点为起点向下重新拉一条重链

第二遍 dfs
以根节点为起点，向下拓展构建重链
选择最大的一个子树的根继承当前重链
其余节点，都以该节点为起点向下重新拉一条重链
给每个结点分配一个位置编号，每条重链就相当于一段区间，用数据结构维护

第二遍 dfs
以根节点为起点，向下拓展构建重链
选择最大的一个子树的根继承当前重链
其余节点，都以该节点为起点向下重新拉一条重链
给每个结点分配一个位置编号，每条重链就相当于一段区间，用数据结构维护
把所有的重链首尾相接，放到同一个数据结构上，然后维护这一个整体即可

```
void dfs1(int x)
{
    size[x]=1;
    for(int i=head[x];i;i=e[i].next)
    {
        if(e[i].to==fa[x])continue;
        dep[e[i].to]=dep[x]+1;
        fa[e[i].to]=x;
        dfs1(e[i].to);
        size[x]+=size[e[i].to];
    }
}
void dfs2(int x,int chain)
{
    int k=0;sz++;
    pos[x]=sz;//分配x结点在线段树中的编号
    bl[x]=chain;
    for(int i=head[x];i;i=e[i].next)
        if(dep[e[i].to]>dep[x]&&size[e[i].to]>size[k])
            k=e[i].to;//选择子树最大的儿子继承重链
    if(k==0)return;
    dfs2(k,chain);
    for(int i=head[x];i;i=e[i].next)
        if(dep[e[i].to]>dep[x]&&k!=e[i].to)
            dfs2(e[i].to,e[i].to);//其余儿子新开重链
}
```

例题 1 BZOJ1036

给定一棵 n 个点的点权树，m 次操作，支持修改结点权值，询问路径最大权值，询问路径权值和

其中 $0 \le n \le 10^5, 0 \le m \le 10^5$

题解

1、单独修改一个点的权值
根据其编号直接在数据结构中修改就行了

题解

1、单独修改一个点的权值
根据其编号直接在数据结构中修改就行了
2、询问点 u 到点 v 的路径上的权值
(1) 若 u 和 v 在同一条重链上
直接在数据结构询问 pos[u] 至 pos[v] 间的信息
(2) 若 u 和 v 不在同一条重链上
一边进行询问，一边将 u 和 v 往同一条重链上靠，然后就变成了情况 (1)

题解

1、单独修改一个点的权值
根据其编号直接在数据结构中修改就行了
2、询问点 u 到点 v 的路径上的权值
(1) 若 u 和 v 在同一条重链上
直接在数据结构询问 pos[u] 至 pos[v] 间的信息
(2) 若 u 和 v 不在同一条重链上
一边进行询问，一边将 u 和 v 往同一条重链上靠，然后就变成了情况 (1)
区间修改操作类似