树分治基础

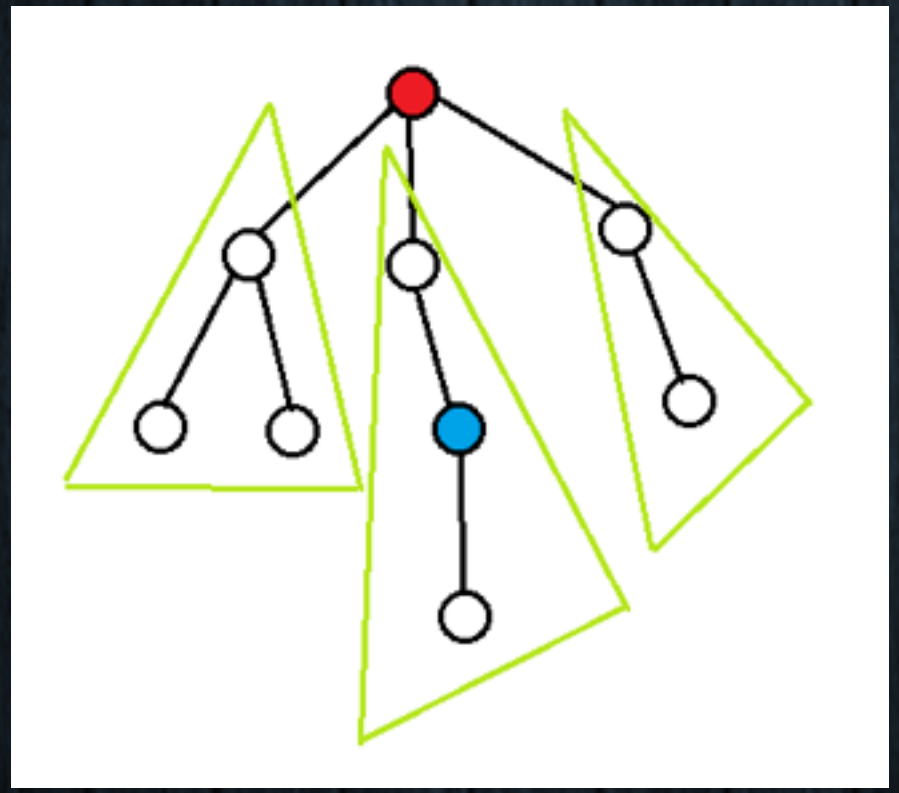
余姚中学 李昊

为什么要讲树分治

- 由于树具有一般图所没有的特点，因此在竞赛中有着更加广泛的应用
- 尤其是关于树中路径的问题，更是频繁的出现在各种比赛中
- 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。

基于点的分治

- 首先选取一个点，将无根树变为有根树，再递归处理每一棵以根节点的儿子为根的子树



如何选点

- 因为是递归的，所以我们当然希望递归的层数尽量小
- 每次选取那个点需要保证将其删去后，结点最多的联通块的结点个数最小，这个点称为树的“重心”
- 重心可以通过树上的动态规划来解决
- （1）dfs一次算出以每个点为根的子树的大小
- （2）选一个点u为根并删去后，结点最多的联通块的结点个数为
  - max{size[v1]……size[vx],n-size[u],v1~vx为点u的儿子节点}
- （3）从中选取最优的那个点作为这一次的根
- 这样做一次的时间复杂度为O(n)

效率分析

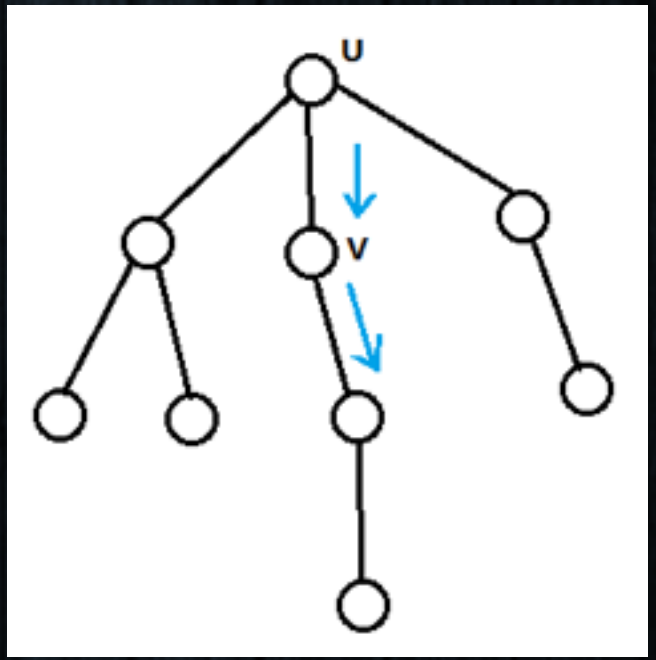
存在一个点使得分出的子树的结点个数均不大于n/2

- 随机选取一个点u作为根
- 如果u存在一个儿子节点v，使得size[v]>n/2，则用v代替u作为根，重复这一步骤直到某个点找不出这样儿子节点。

- 为什么一定能停止?

效率分析

- 考虑如下图，当根从u变为v后，不可能再从v变回u
- 因为当size[v]>n/2，点u所在联通块的大小=n-size[v]<n/2
- 所以根在调整的过程中只会不断向下移动
- 那么一定能找到一个点符合条件



效率分析

- 基于上述定理，在每次点分治之后，联通块的大小至少减少一半
- 因此递归深度最多log n层
- 总复杂度O(nlog n)

代码实现

- int solve(int u)//处理u所在联通块
- {
  - u=get(u);//找出该联通块的重心
  - work(u);//处理该联通块中通过点u的所有路径
  - p[u]=1;//将点u删除
  - 枚举每个与u相邻的点v
    - if(!p[v])solve(v);//如果点v未被删除则处理其所在联通块
- }

IOI2011 Race

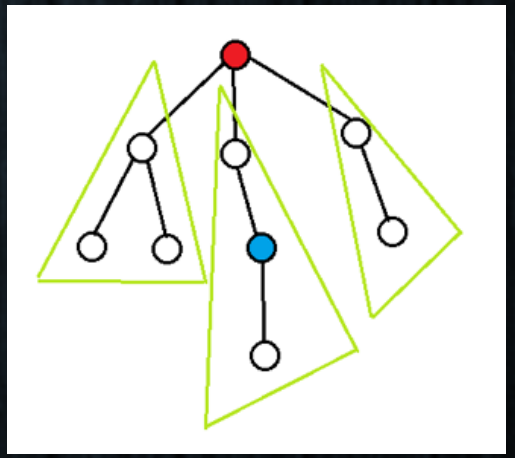
- 给一棵树,每条边有权.
- 求一条路径,权值和等于K,且边的数量最小.
- 数据范围：n<=200,000，k<=1,000,000

Solution

- 在每一棵点分治的树中只考虑经过根的路径
- （1）某一点到根的路径
  - 只需要算出每个点到根的距离即可判断
- （2）来自根节点不同儿子所在子树的两个点构成的路径
  - 每个点相当于有三个参数belong[i],dis[i],s[i]，分别表示每个点在删除根后属于哪个联通块，到根路径长度，到根路径上经过的点数
  - 原问题相当于求min{s[i]+s[j]-1 | belong[i]≠belong[j],dis[i]+dis[j]=k}

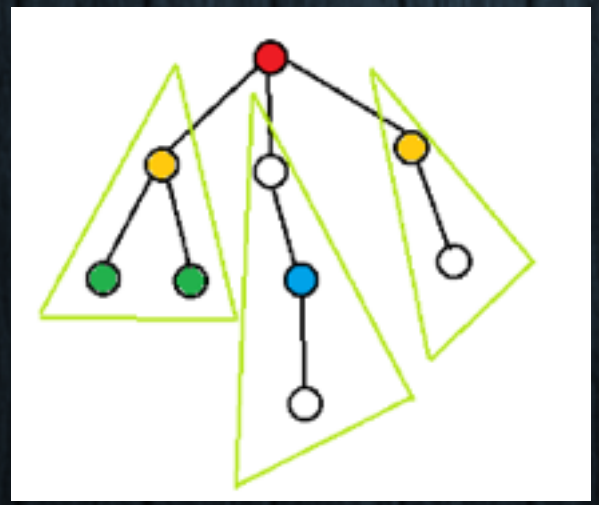
Solution

- 依次处理根的每一棵子树
- f[i]表示已经处理过的子树中到根距离为i的点中s值最小为多少
- 当处理下一颗子树时，每个点所能匹配的点到根的距离都是固定的，直接拿出对应的f值更新答案即可，然后再用这一棵子树更新f数组即可
- 这样保证了更新答案的两点belong值不同，dis相加为k，同时直接找出了当前最优解



Solution

- 这样为什么是对的？每条路径都被考虑了吗？
- 对于任意一条合法的路径，都至少在一棵点分治的树中被考虑过
- 如图中的黄色点对，会在以红色点为根的点分治树中被考虑
- 图中的绿色点对则会在递归到以左边的黄色点为根的子树时被考虑



Spoj 1825 Free tour II

- 给定一棵含有n个结点的带权树，其中结点分为两类，黑点和白点。
- 要求找到一条路径，使得经过的黑点数不超过K 个，且路径长度最大。
- 数据范围：n<=200000

Solution

- 与上题相同，我们只需要考虑过根结点的路径，其余的递归处理即可。
- （1）某一点到根的路径
  - 只需要算出每个点到根的距离和路径上的黑点个数即可
- （2）来自根节点不同儿子所在子树的两个点构成的路径
  - 每个点相当于有三个参数belong[i],dis[i],s[i]，分别表示每个点在删除根后属于哪个联通块，到根路径长度，到根路径上经过的黑点点数
  - 原问题相当于求max{dis[i]+dis[j] | belong[i]≠belong[j],s[i]+s[j]<=k}

Solution

- 依次处理根的每一棵子树
- f[i]表示已经处理过的子树中到根路径上黑点个数为i的点中dis值最大为多少
- 当处理点u时，我们可以用max{f[1]~f[K-s[u]]}+dis[u]来更新答案，做完一棵子树后再用这棵子树的信息来更新f数组即可
- 那这里使用线段树来维护f数组即可
- 总时间复杂度O(nlog^2n)

Solution

- 考虑不用数据结构
- f[i]表示已经处理过的子树中到根路径上黑点个数小于等于i的点中dis值最大为多少
- 当处理点u时，我们可以用f[K-s[u]]+dis[u]来更新答案
- 做完一棵子树后再用这棵子树的信息来更新f数组

Solution

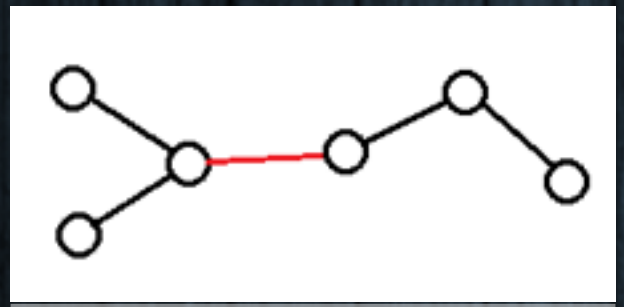
- 如何更新f数组?
- 直接暴力更新，复杂度max{len1,len2}{len1表示当前子树的最大s值，len2表示之前处理过的子树中最大的s值}
- 时间复杂度?
- 记根的第i个儿子所在的子树中最大的s值为maxs[i]，总共m个儿子
- 复杂度为maxs[1]+max{maxs[1],maxs[2]},max{maxs[1]…maxs[m]}
- 最坏情况下复杂度O(n^2)
- 如何改进?

Solution

- 我们发现根结点的儿子顺序对答案是不会有影响的
- 所以可以按照maxs值将所有子树排序后再处理
- maxs[1]+maxs[2]…maxs[m]<=n
- 时间复杂度O(n)
- 因为每次需要排序的子树的个数的总和等于边数，所以总复杂度为O(nlogn)

基于边的分治

- 在树中选取一条边，将原树分成两棵不相交的树，递归处理。



如何选边

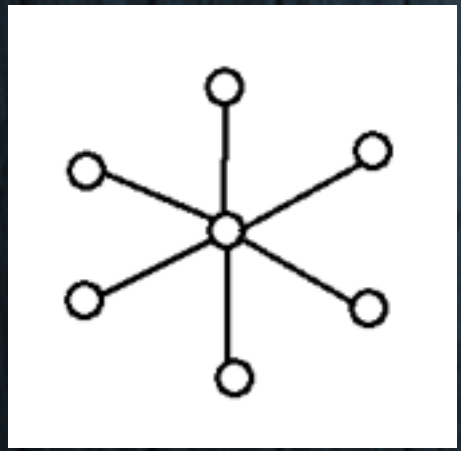
- 回顾点分治选点的策略：
- “每次选取那个点需要保证将其删去后，结点最多的联通块的结点个数最小”
- 同样的，这里选择的边需要保证将其删去后，结点最多的联通块的结点个数最小
- 也是用树形动态规划来实现，做法与找重心及其相似

效率分析

- 不妨令D为所有点的度的最大值。
- 当D=1时，命题显然。
- 当D>1时，我们设最优方案为边U-V，且以U,V为根的两棵子树的结点个数分别为S和n-S，不妨设S>=n-S。
- 设X为U的儿子中以X为根的子树的结点个数最大的一个，
- 我们考虑另一种方案X-U，设除去边X-U后以X为根的子树结点个数为P。显然P>=(S-1) /(D-1)，由于P<S 且边U-V是最优方案，所以N-P>=S，与P>=(S-1)/(D-1)联立可得S<=(n*(D-1)+1)/D

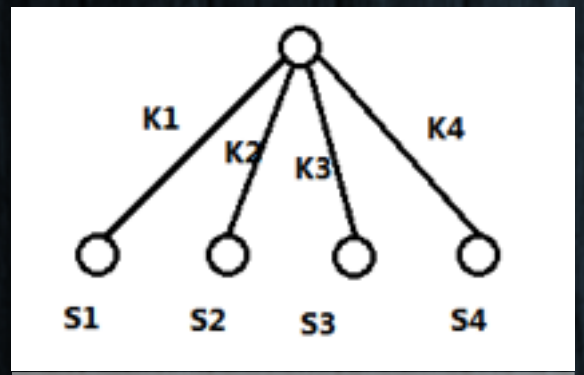
效率分析

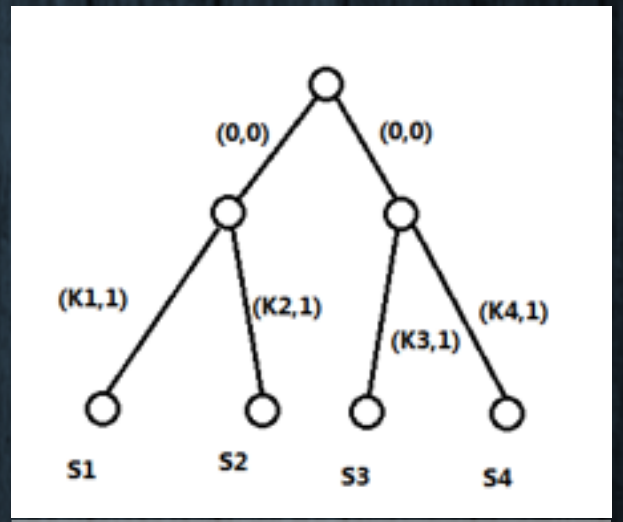
- 当D为常数时，基于边的分治递归最坏深度为O(logn)。
- 但是在一般的题目中，D可能较大甚至达到O(n)，这时这个算法的效率十分低



如何改进基于边的分治的时间复杂度

- 考虑race那题，有用的只有边的长度和条数，这提醒我们可以在树中加入不影响答案的边和点





如何改进基于边的分治的时间复杂度

- 如上图那样就可以做到等价转换，做成类似线段树一样的结构，那么每个点的度数至多为3，新树的节点个数最多为2n
- 边分治的递归层数也降到log n的级别

使用基于边的分治解决Free tour II

- 给定一棵含有n个结点的带权树，其中结点分为两类，黑点和白点。
- 要求找到一条路径，使得经过的黑点数不超过K 个，且路径长度最大。
- 数据范围：n<=200000

Solution

- 解法与点分治类似
- 边分治只需考虑两棵子树，用第一棵子树维护f数组，第二棵子树直接在f数组上找答案即可
- 无需像点分治那样考虑处理子树的顺序等问题，思维难度更低
- 复杂度仍然为O(nlogn)

ZJOI2007 Hide 捉迷藏

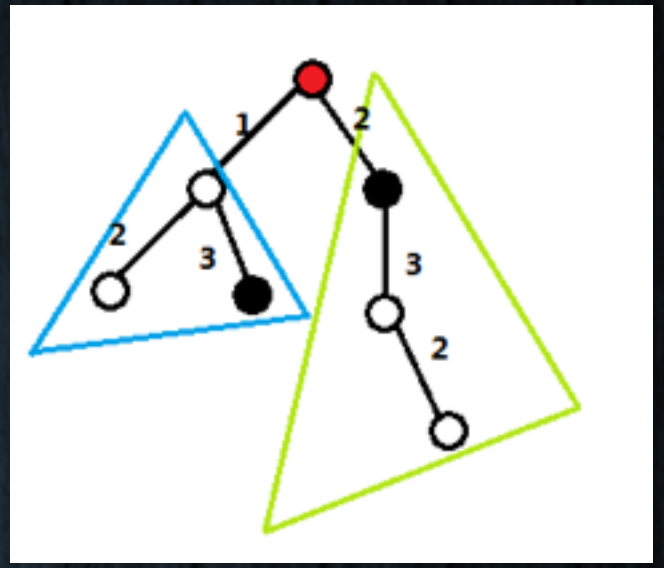
- 给定一棵树，每个点为黑色或白色，每次修改一个点的颜色或询问最远的白色点对的距离。
- 数据范围：n<=10^5，m<=5*10^5

Solution

- 如果不修改，用树分治怎么做?
- 对于每一层点分治，对根的每一个儿子的子树中找出距离根最远的白色节点，将属于不同子树的最远的两个白色节点作为当前层的答案。
- 如何维护修改颜色?
- 为了维护每一棵子树中到根最远的白色节点，可以为每一棵子树建一个数据结构来维护，比如：大根堆、线段树、平衡树等
- 为了找出每一层点分治中来自不同子树的最远的两个白色节点，建一个堆，从上述维护子树的堆中找出最大的存入当前堆，那么当前堆中最大的两个元素即可用来更新答案
- 又因为有多个点分治的树，每个都可用来更新答案，且每个都可能会被修改，所以我们再开一个全局的堆来存每一棵点分治树上的答案。

Solution

- 如右图：
- 红点为根，蓝色区域和绿色区域皆为子树
- 对于蓝色区域，维护的堆为{3,1}
- 对于绿色区域，维护的堆为{7,5}
- 对于这个点分治的树，维护的堆为{7,3}
- 对于全局的堆，其中存了10，作为这一棵点分治树的答案

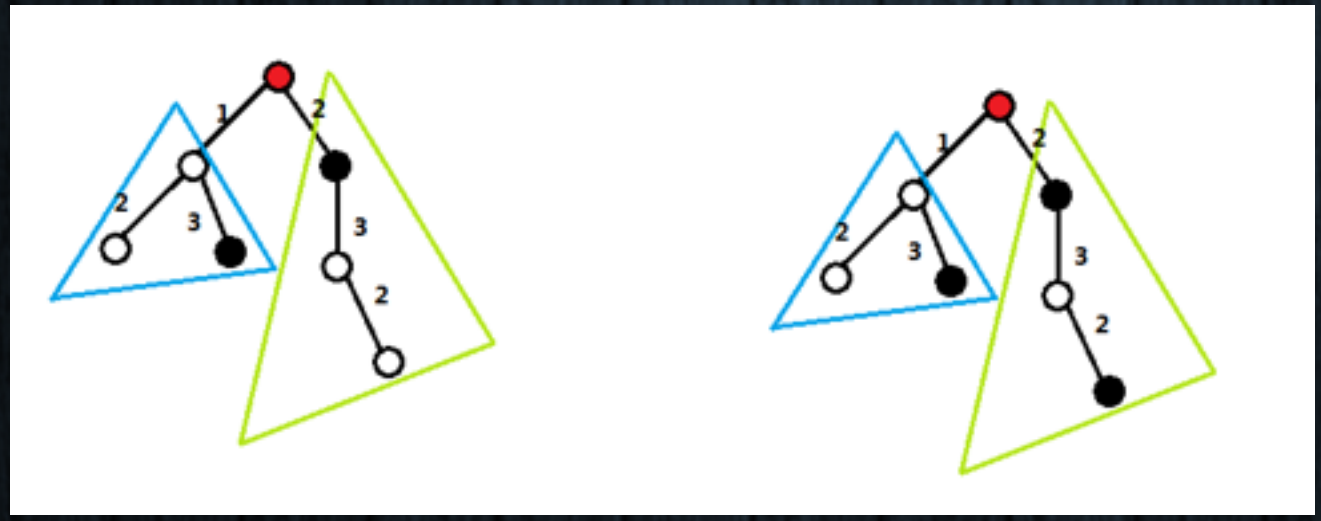


Solution

- 对于每一次修改：
  - 在每一层点分治树对应的子树中，在堆中删除或添加该点到根的距离
  - 若影响到这一点分治树的堆，将其修改
  - 若影响到了全局的堆，将其修改

Solution

- 如下图，将图中绿色区域最下方的点变为黑色：
- 对于绿色区域，维护的堆删除元素7，变为{5}
- 对于这个点分治的树，维护的堆删除7添加5变为{5,3}
- 对于全局的堆，删除元素10，添加元素8，作为这一棵点分治树的答案



Solution

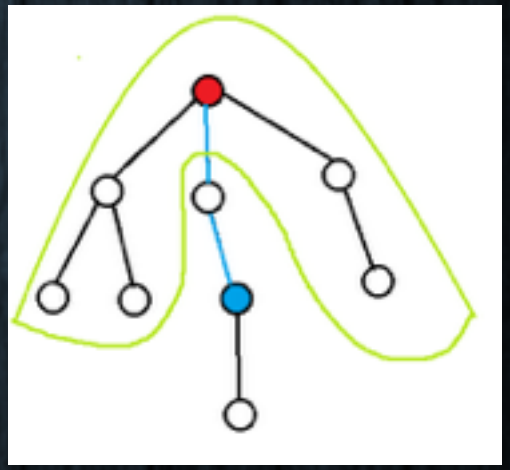
- 对于询问，直接取出维护全局堆的最大值就是答案了
- 时间复杂度O(mlog^2n)
- 常数偏大的数据结构可能会被卡

HNOI2015 开店

- 给定一棵树，每个点有一个颜色，多次询问颜色在[l,r]区间内的所有点与某个点之间的距离之和，强制在线
- 数据范围： n<=150000，Q<=200000，颜色<=10^9

Solution

- 如果没有颜色限制该怎么做?
- 考虑询问的点为图中蓝色的点
- 这里只考虑经过根的路径（即只考虑图中以图中绿色部分的点为起点，蓝色点为终点的路径）
- 这一部分的贡献为图中绿色部分的点到根距离总和+图中绿色部分的点个数*蓝色点到根的距离
- 剩余的答案只需递归处理即可
- 时间复杂度O(nlogn)



Solution

- 如果有颜色限制，我们只需将每一棵点分治的子树中，按每个点的颜色排序，并记录每个点到根的距离，询问时只需要二分合法范围即可
- 总时间复杂度O(nlog^2n)

Zjoi2015 幻想乡战略游戏

- 给定一棵N个结点的有正的边权、初始点权为0的无根树，进行M次操作，每次将一个点u的权值增加e(0≤|e|≤1000)，保证任意时刻点权非负。你的任务是在每次操作后找到一个带权重心u，使得所有点到重心的距离与点权的乘积之和最小，即最小化sum{dist(u, v)*val[v]}
- 数据范围：n,m<=10^5.保证每个点的度数均不超过20.

Solution

- 如果已经知道带权重心是哪一个，那么这题就是上一题的简化版本，只需要再支持修改点权即可，而这个只需要在logn棵点分治的树上简单维护即可
- 所以这里我们只考虑，改变点权后如何快速找到带权重心

Solution

- “可以考虑使用点分治。注意到如果当前分治点是u，那么显然，我们可以先看看u是不是重心，如果不是，那么重心只可能在u最大的孩子里，这样问题就变成了u的一个子分治的子问题。”
- 总时间复杂度O(nlog^2n)

Wc2014 紫荆花之恋

- 给定一棵树，每次添加一个节点并询问当前有多少点对满足dis(i,j)<=ri+rj 强制在线
- 数据范围：n<=100000

Solution

- 对于每一棵点分治的树
- 设点u到根的距离为dis[i]，点i的权值为r[i]
- 两个点是合法点对当且仅当
- belong[i]≠belong[j]，d[i]+d[j]<=r[i]+r[j]
- 移项得d[i]-r[i]<=r[j]-d[j]
- 所以我们用数据结构记录每一个d[i]-r[i]
- 当新加入一个点j时，在每一层点分治树中，与点j是合法点对的数量的点个数为：
- 与其不在同一子树内且d[i]-r[i]<=r[j]-d[j]的个数

Solution

- 当数据随机时新加入一个点计算新答案时需要寻找logn层左右的点分治树
- 但是当数据不随机，每次加入一个新节点如果一直往一边插入会导致树不平衡，就不满足点分治树的性质了
- 我们可以用动态点分治来解决这个问题，用替罪羊树的思想，当一个子树i中有一个子树j的大小大于子树i的大小*k时就暴力重构子树i
- 总复杂度是 O(nlog^3(n))，具体复杂度证明可以参考VFK的论文

参考文献

- 《分治算法在树的路径问题中的应用》漆子超
- http://wjmzbmr.com/ 陈老师博客
- http://vfleaking.blog.163.com/VFK博客

祝大家省选顺利