分治算法

在树的路径问题中的应用

长沙市雅礼中学 漆子超

树的路径问题

以路径为询问对象的题目

POJ1741, 树中点对统计

SPOJ QTREE,FTOUR2,QTREE4

Astar2008 复赛 黑白树

论文内容

一、树的分治算法

树的分治的两种常见形式：

基于点的分治

基于边的分治

二、树的路径剖分算法

归纳为基于链的分治

三、树的分治算法的进一步探讨

如何改进基于边的分治的时间复杂度

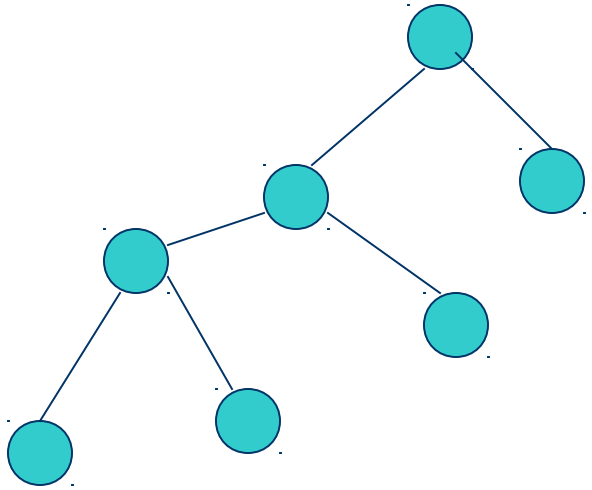
一、树的分治算法

树的分治算法是分治思想在树型结构上的体现。

T：除去树中的某些对象，使原树被分解成若干互不相交的部分。

两种常见的形式

基于点的分治

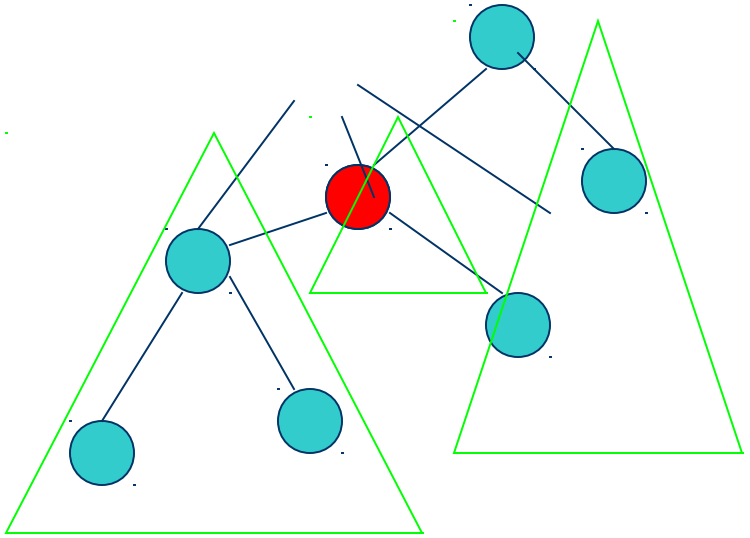


两种常见的形式

基于点的分治

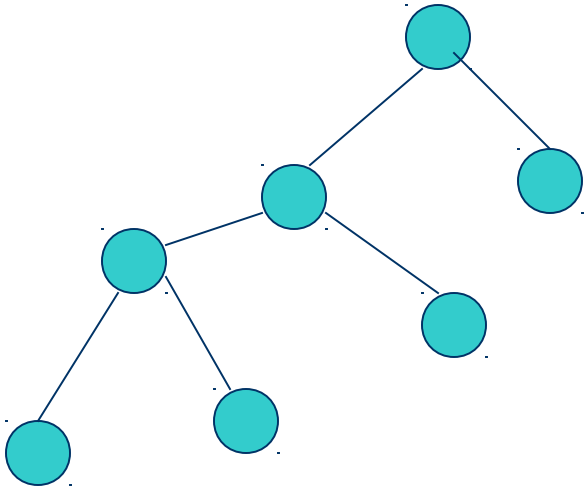
1. 选取一个点将无根树转为有根树

2. 递归处理每一颗以根结点的儿子为根的子树



两种常见的形式

基于边的分治

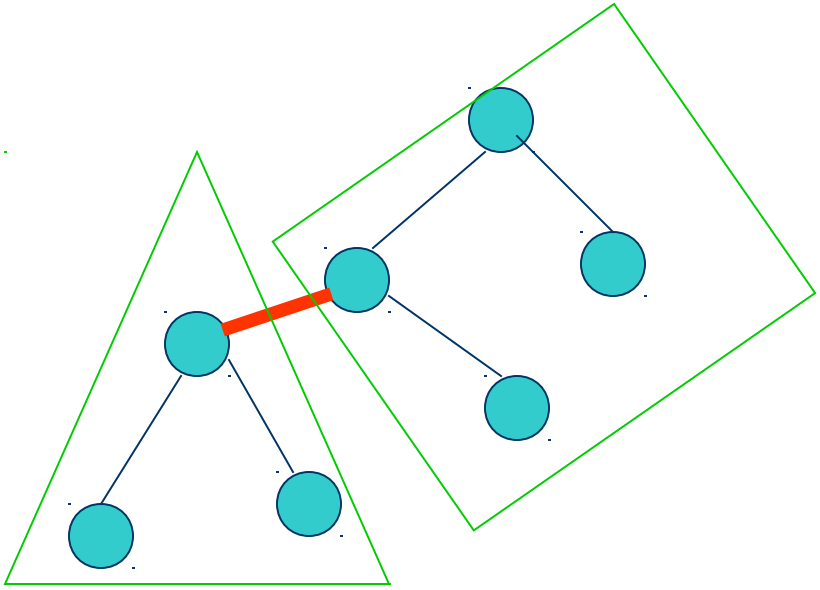


两种常见的形式

基于边的分治

1. 在树中选取一条边

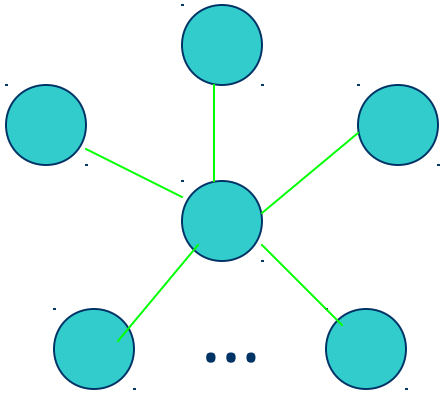
2. 将原有的树分成两棵不相交的树，递归处理。



效率分析

可以证明在基于点的分治中，如果每次都选取树的重心，那么至多递归 O(LogN) 次。

基于边的分治最坏情况下递归次数为 O(N) 。



【例一】树中点对统计

给定一棵 N 个结点的带权树。

定义 dist(u ， v)=u ， v 两点间的路径长度，路径的长度定义为路径上所有边的权和。

给定一个 K ，如果对于不同的两个结点 a ， b ，如果满足 dist(a ， b)≤K ，则称 (a ， b) 为合法点对。

求合法点对个数。

$N \le 10000, K \le 10^9$

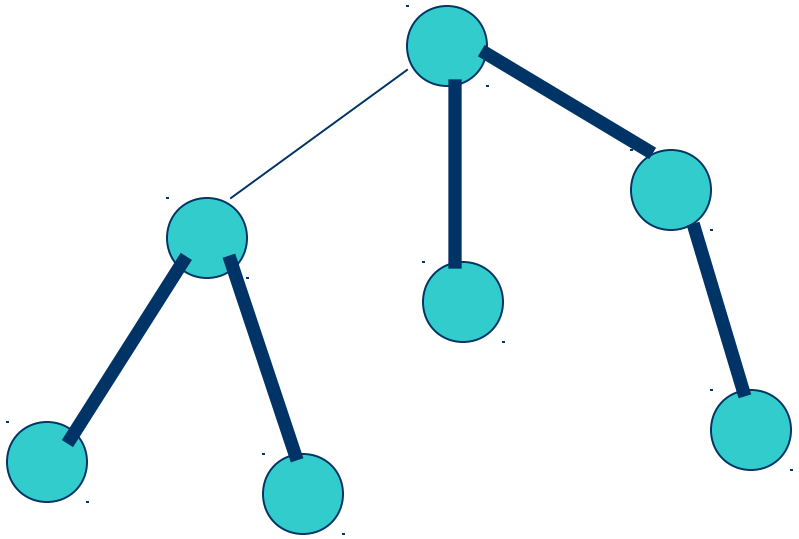
树中点对统计

一条路径：

1. 过根节点 ？

2. 在一颗子树内

——→递归处理



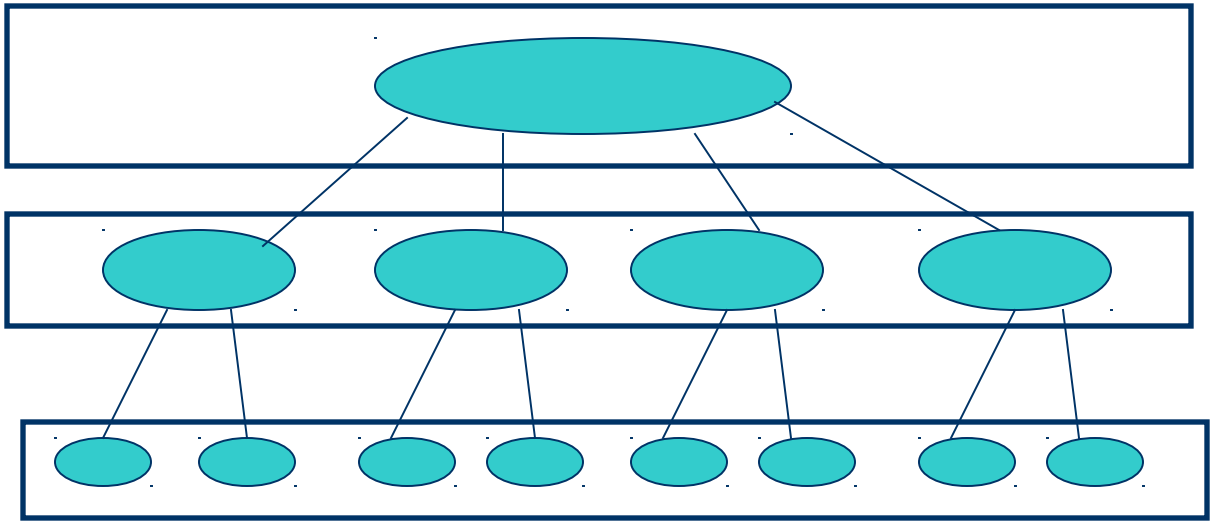
树中点对统计

记 D(i) 表示节点 i 到根节点路径的长度

Answer = 满足 $D(i)+D(j)\le K$ 的 (i,j) 个数　i,j 属于不同的子树

O(NlogN)

时间复杂度分析



每层的时间复杂度不超过 O(NlogN)

最多递归 O(logN) 次

$O(N\log^2 N)$

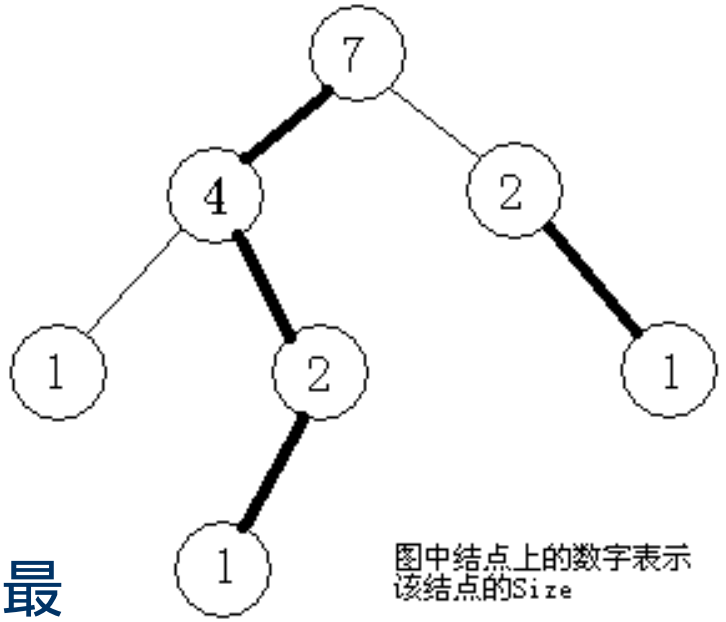
二、路径剖分算法

轻重边路径剖分

将树中的边分为两类：轻和重边。

记 Size(U) 表示以 U 为根的子树的结点个数。

令 V 为 U 的儿子中 Size(V) 最大的一个，那么我们称边 (U ， V) 为重边，其余边为轻边。



轻重边路径剖分

我们称某条路径为重路径，当且仅当它全部由重边组成。那么对于每个点到根的路径上都不超过 O(logN) 条轻边和 O(logN) 条重路径。

路径剖分算法常用来高效的维护点到根的路径

Spoj 的 Qtree,Astar2008 的黑白树...

【例二】 Query On a Tree Ⅳ

给定一棵包含 N 个结点的树，每个节点要么是黑色，要么是白色。要求模拟两种操作：

1) 改变某个结点的颜色。

2) 询问最远的两个黑色结点之间的距离。

数据范围：

N≤100000, 边权绝对值不超过 1000

另寻他法

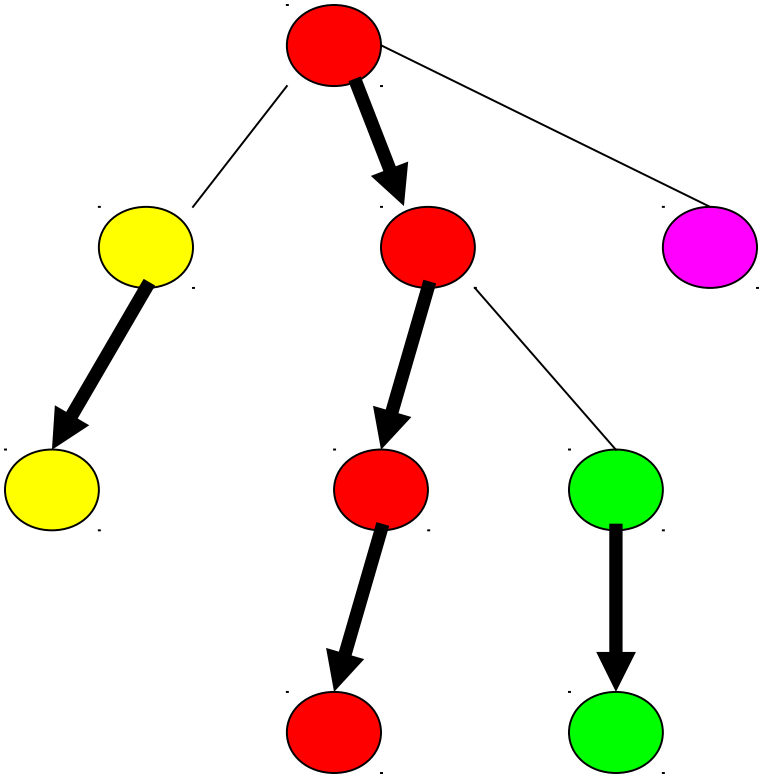
此题出自 2007 年浙江省选，但此题中树的边权可能为负，无法使用括号序列。

路径剖分算法

这道题的算法似乎与路径剖分毫无关系，那么我们是否能用路径剖分算法解决此题呢？

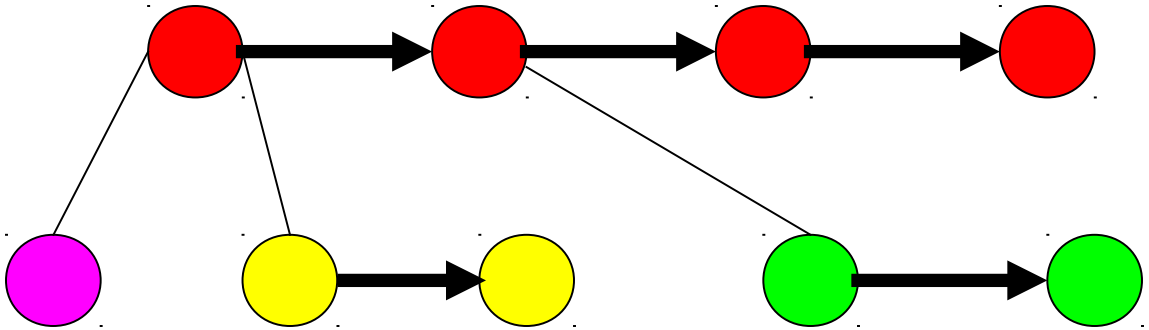
路径剖分与树的分治的联系

一棵树及其剖分



路径剖分与树的分治的联系

按照点到根结点路径上的轻边个数分层摆放。



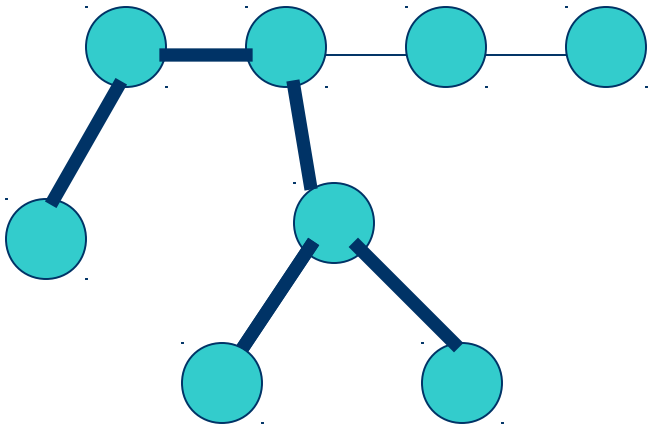
递归树！

路径剖分每次删除了一条链，所以路径剖分算法可以看做是***基于链的分治***

Query On a Tree Ⅳ

将路径剖分理解成基于链的分治后，我们可以用类似基于点的分治的方法将路径分类。

1. 与链有重合部分



2. 与链没有重合部分

⟶递归处理

Query On a Tree Ⅳ

我们的目标就是要求出满足与此链的重合部分在 [1,N] 的路径的最大长度。

我们可以用线段树解决这个问题。

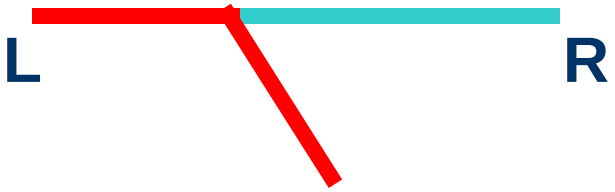
Query On a Tree Ⅳ

记 D(i) 表示第 i 个结点至子树内某个黑色结点的路径中长度的最大值。 Dist(i,j) 表示链上的第 i 个点到第 j 个点的距离。

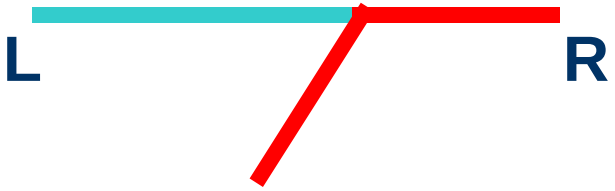
Query On a Tree Ⅳ

对于线段树中的一个区间 [L,R], 我们需要记录下面三个量：

$$MaxL = Max\{Dist(L,i) + D(i)\}$$



$$MaxR = Max\{D(i) + Dist(i,R)\}$$

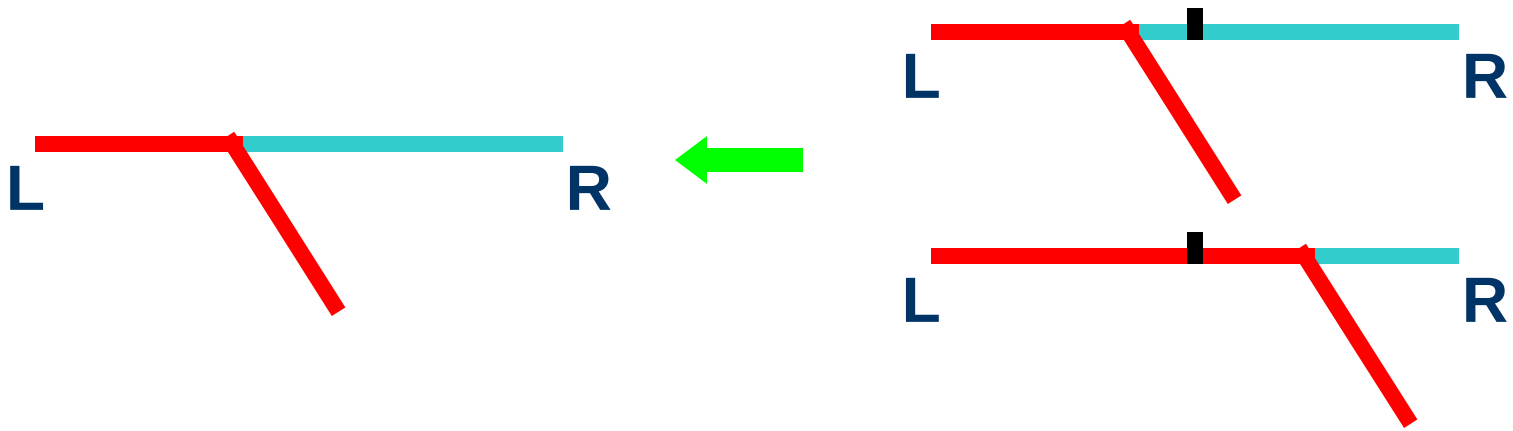


Opt = 与此链的重合部分在 [L,R] 的路径的最大长度

Query On a Tree Ⅳ

设区间 [L,R] 的结点编号为 P， Lc,Rc 分别表示 P 的左右两个儿子，区间 [L,Mid] 和 [Mid+1,R] 。我们可以得到如下转移：

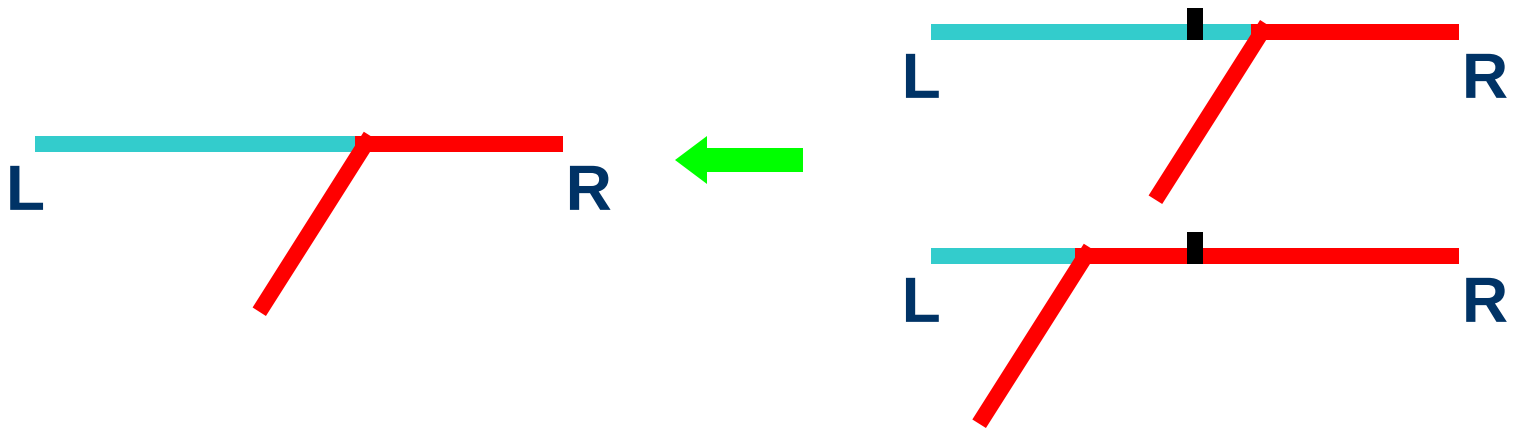
$$MaxL(P) = Max\{MaxL(Lc), Dist(L, Mid+1) + MaxL(Rc)\}$$



Query On a Tree Ⅳ

设区间 [L,R] 的结点编号为 P， Lc,Rc 分别表示 P 的左右两个儿子，区间 [L,Mid] 和 [Mid+1,R] 。我们可以得到如下转移：

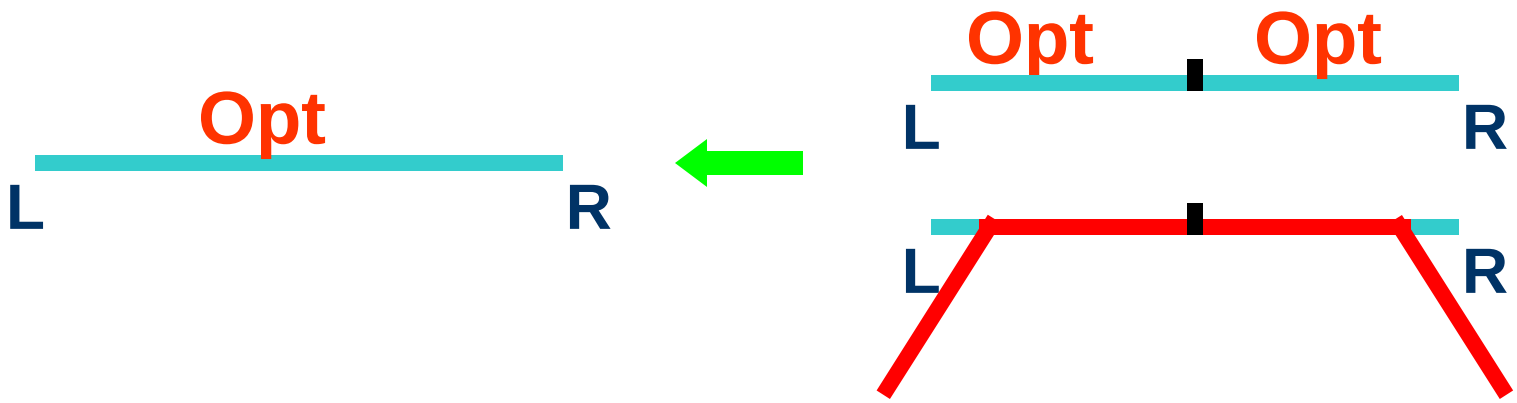
$$MaxR(P) = Max\{MaxR(Rc), MaxR(Lc) + Dist(Mid, R)\}$$



Query On a Tree Ⅳ

设区间 [L,R] 的结点编号为 P， Lc,Rc 分别表示 P 的左右两个儿子，区间 [L,Mid] 和 [Mid+1,R] 。我们可以得到如下转移：

$$Opt(P) = Max\{\begin{matrix} Opt(Lc), Opt(Rc), \\ MaxR(Lc) + MaxL(Rc) + Dist(Mid, Mid+1) \end{matrix}\}$$



Query On a Tree Ⅳ

注意到 Dist(i,j) = Dist(1,j) – Dist(1,i) **O(1)**

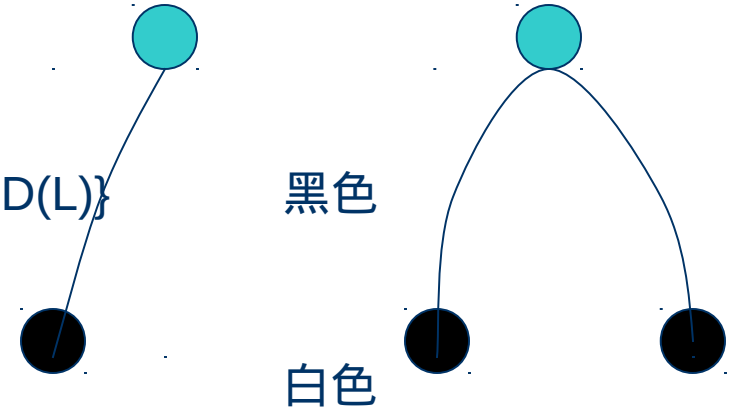
Query On a Tree Ⅳ

对于边界情况 [L,L],

MaxL = D(L)

MaxR = D(L)

Opt = { Max{D(L)+D2(L),D(L)}　黑色
　　 { D(L)+D2(L)　白色

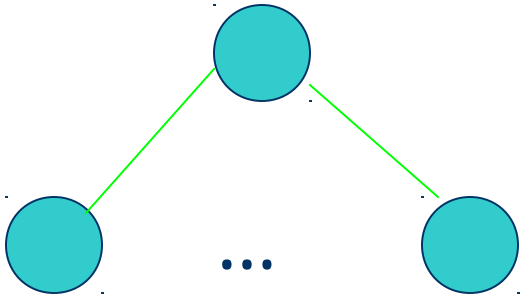


D2(i) 表示第 i 个结点至子树内某个黑色结点的距离。

问题只剩下如何维护 D 和 D2 的值

Query On a Tree Ⅳ

一个点向下至某个黑色结点的路径



链的头结点到某个黑点路径的最大长度

链的头结点

这正是我们前面已经维护了的量 MaxL

链的头结点

Query On a Tree Ⅳ

我们可以使用堆来维护一个点向下至某个黑色结点的路径长度集合

O(1)

时间复杂度分析

询问操作：

我们使用堆来存贮每条链的最优结果

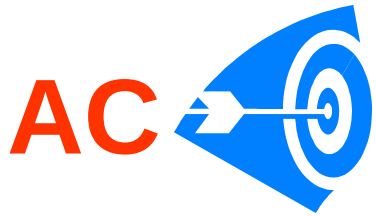
O(1)

修改操作：

修改一个点最多影响 O(logN) 条链，对于每条链我们需要修改堆和线段树， O(logN)

$O(log^2N)$

路径剖分 —进一步的分析→ 基于链的分治



三、树的分治算法的进一步探讨

基于点的分治　　删除一个点后树的个数太多，加大了设计高效算法的难度

基于边的分治　　删除一条边后仅有**两**棵树

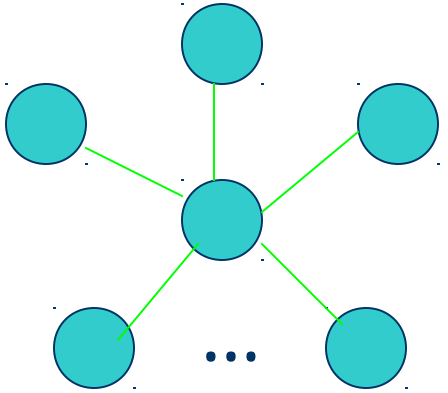
最坏的时间复杂度限制了该算法的应用

改进！

如何改进基于边的分治的时间复杂度

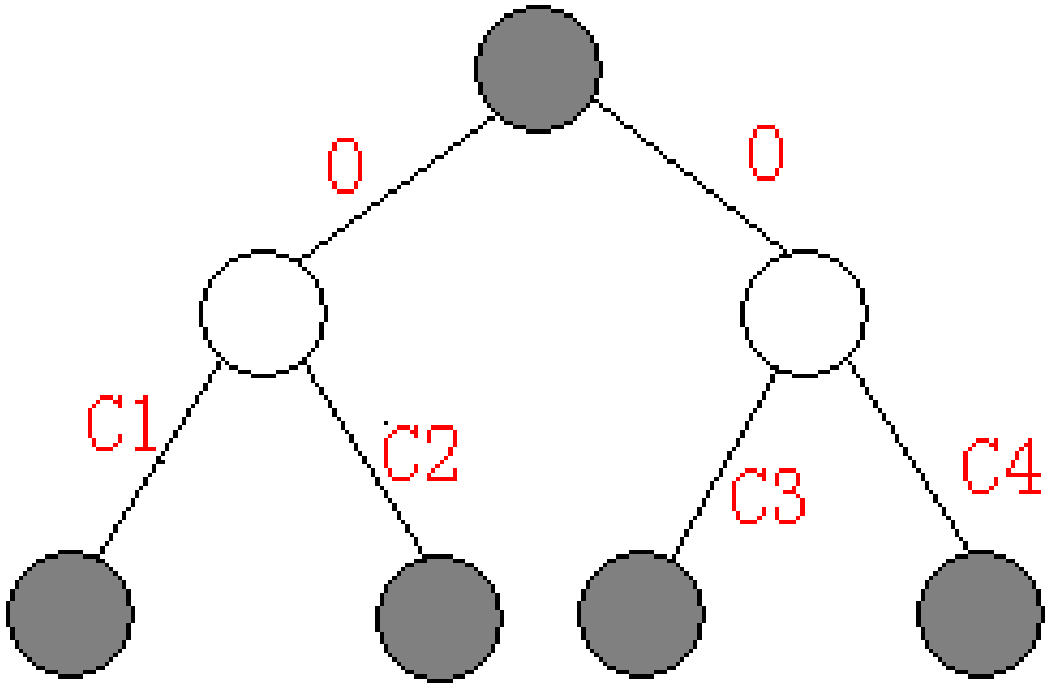
改变选择边的方法？ X

无论选择哪条边，结果都是一样的



改变树的结构！

如何改进基于边的分治的时间复杂度



如何改进基于边的分治的时间复杂度

通过对每个结点到其儿子的路径中加入了白色结点，使之成为了类似**线段树**的结构。

叶节点为 N 的线段树共有 2N 个结点，所以含有 N 个结点的树转化后所得的新树最多包含 2N 个结点。

每个点的度至多为 3

如何改进基于边的分治的时间复杂度

定理：

如果一棵包含 N 个结点的树中每个点的度均不大于 D ，那么存在一条边，使得分出的两棵子树的结点个数在 [N/(D+1),N*D/(D+1)] 。

改进后的算法最坏情况下递归深度为

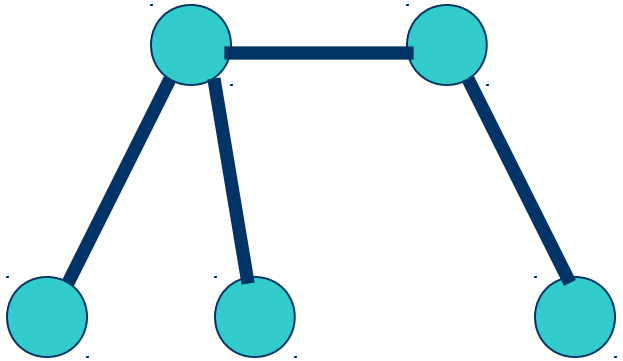
O(LogN)

使用基于边的分治解决上题

一条路径：

1. 过中心边

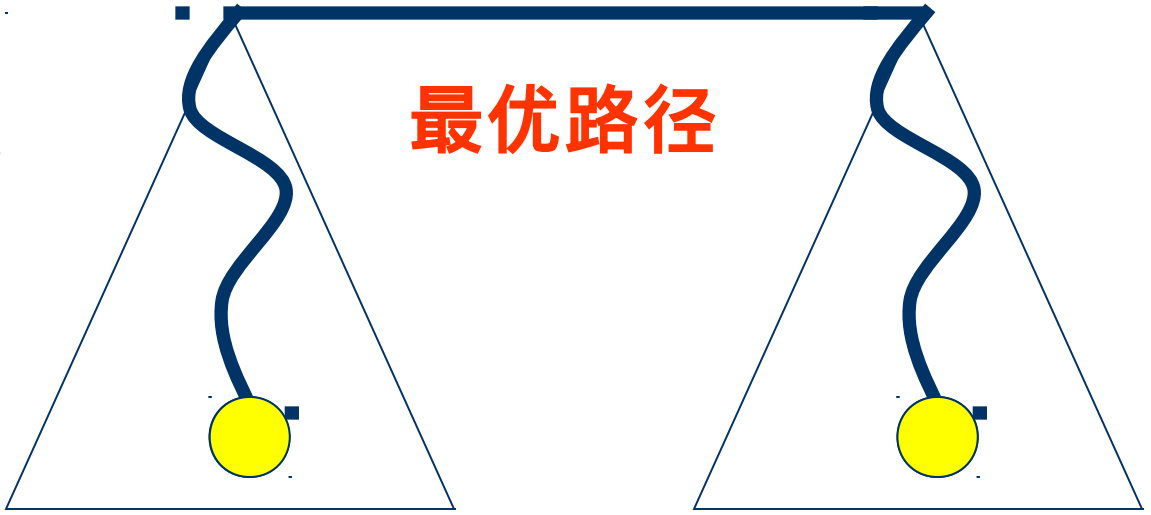
2. 在一颗子树内



⟶递归处理

使用基于边的分治解决上题

记录两个根结点到其子树内某个黑色结点的路径的最大长度



修改 O(logN)

询问 O(1)

时间复杂度分析

询问操作：

对每颗树都记录其两个子树的最优值

O(1)

修改操作：

一个点最多属于 O(logN) 棵树，对于每棵树我们需要修改堆， O(logN)

$O(\log^2 N)$

我们达到了与使用路径剖分同阶的时间复杂度。

算法更加简单

总结

1. 算法的常数：

基于链的分治 < 基于点的分治 < 基于边的分治

2. 基于链的分治可以用来维护路径上的点(边)。如果维护的对象是路径的长度，基于点(边)的分治算法的能力更强。3. 与基于点的分治比较，基于边的分治在设计高效算法的思考难度上明显小于前者。

这几个算法各有所长，需要我们根据具体情况，灵活运用，以最佳的方式解决题目。

算法的常数

1. 在路径剖分算法中，链的长度和链的个数是相互制约的，因此路径剖分算法在实际运行中是很快的。

2. 为了改进基于边的分治的最坏复杂度，我们将一个结点个数为 N 的树改造成了一个结点个数为 2N 的新树，自然增加了常数。

算法的常数

	基于链的分治	基于点的分治	基于边的分治	未改进的 基于边的分治
N=100000 M=100000	1.17s	1.78s	2.28s	2.13s
N=100000 M=500000	2.35s	3.80s	5.79s	5.73s

测试环境:

Intel® Core™2 Duo T7250 2.00GHz,1GB

编译器: Visual C++ 2008 , Release 模式

算法的常数

	hide6.in	hide7.in	hide8.in	hide9.in
基于链的分治	0.26s	0.59s	1.13s	1.90s
线段树	0.21s	0.45s	1.10s	2.35s

测试环境:

Intel® Core™2 Duo T7250 2.00GHz,1GB

编译器: Visual C++ 2008 , Release 模式

Free Pascal 2.1.4

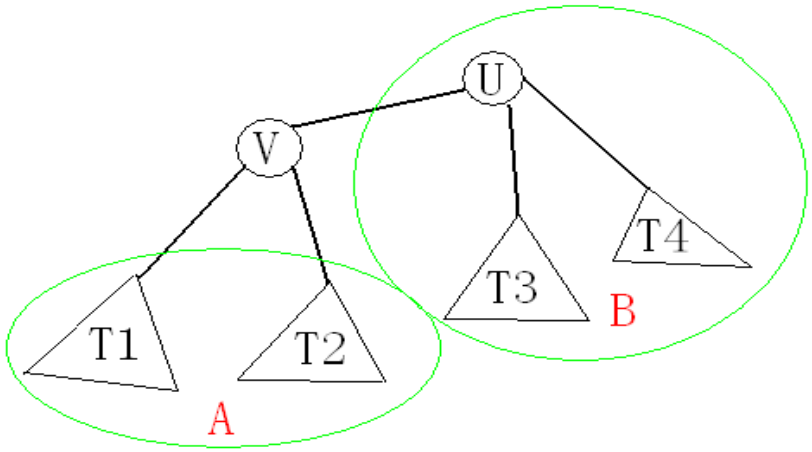
树的重心

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，这个点被称作”树的重心”。

定理：存在一个点使得分出的子树的结点个数均不大于 N/2

证明：

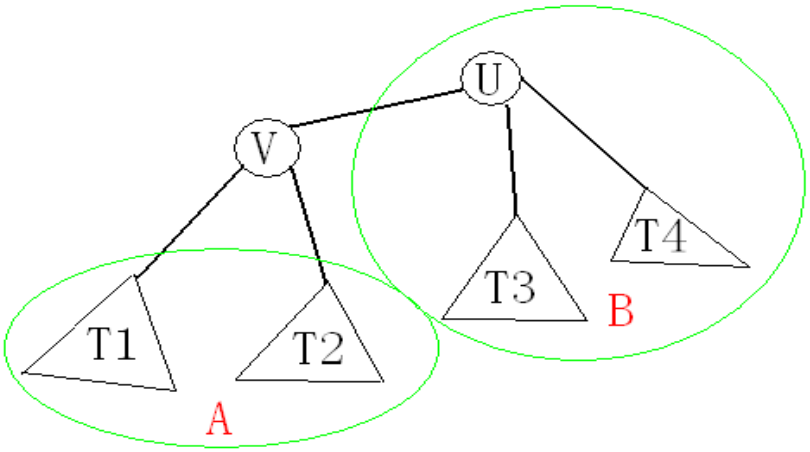
假设 U 是树的重心，记 Size(X) 表示以 X 为根的子树的结点个数。记 V 为 U 的儿子中 Size 值最大的点。



定理：存在一个点使得分出的子树的结点个数均不大于 N/2

证明：

假设 Size(V)>N/2 ，那么我们考虑 V 作为根结点的情况，记 Size’(X) 表示此时以 X 为根的子树的结点个数。



定理：存在一个点使得分出的子树的结点个数均不大于 N/2

证明：

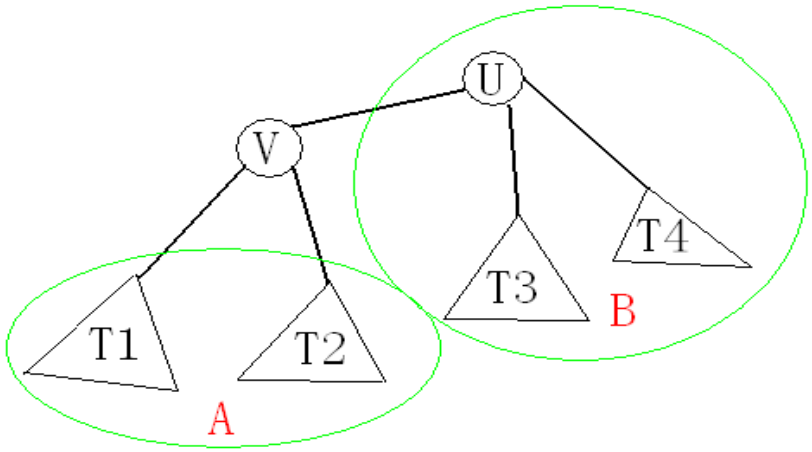
如图。

对于 A 部分，显然 Size’(Ti)<Size(V)

对于 B 部分， Size’(U) = N-Size(V)<Size(V)

这与树的重心定义矛盾。

定理得证。



定理：

如果一棵包含 N 个结点的树中每个点的度均不大于 D ，那么存在一条边，使得分出的两棵子树的结点个数在 [N/(D+1),N*D/(D+1)] 。

证明：

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 。

最优方案指选取一条边使得删除这条边后所分离出来的两棵子树的结点个数较大值最小。

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 $P \geq (S-1)/(D-1)$ ，由于 $P<S$ 且边 (U,V) 是最优方案，所以 $N-P \geq S$ ，与 $P \geq (S-1)/(D-1)$ 联立可得 $S \leq ((D-1)N+1)/D$ ，又 $N \geq D+1$ ，所以 $S \leq N*D/(D+1)$。

证毕