树上问题

zzq

树上倍增

- 用来解决求 lca 等问题。
- 记 dep[x] 为 x 的深度， up[x][d] 为 x 的第 2^d 个祖先。
- 考虑如何求 lca(a,b) ，首先把较深的一个跳到和较浅的一个一样高，如果不相等，我们再两边一起跳，跳到最高的不相等的地方，这个点的父亲就是 lca 。

dfs 序

- 比如我们现在要处理，单点加，子树加，单点查询，子树求和
- 每个子树在 dfs 序上是一个区间，直接维护即可

dfs 序 -cont

- 比如我们现在要处理，单点加，单点查询，链求和
- 链剖？线段树？

dfs 序 -cont

- 考虑搞一个双 dfs 序，进 dfs 的时候记一下，出 dfs 的时候记一下，第一次给 +1 的系数，第二次给 -1 的系数，那么一个点到根的链和就是带系数的 dfs 序里的权值前缀和。
- 求一个 lca 减一下就行了。

dfs 序 -cont - O(1) lca

- 双 dfs 序也可以用来求 lca ，可以证明 a 和 b 两点的 lca 就是在 a 和 b 第一个 dfs 序之间的点深度最小的。
- 求区间最小值我们有一个 O(nlogn) 预处理 O(1) 询问的做法。我们记 f[i][j] 为 i 开始长度为 2^j 的区间的最小值，我们预处理出所有 f ，询问时只需要对两个 f 取 min 即可。那么我们就可以在 O(nlogn) 的预处理之后 O(1) 求 lca 。

树上差分

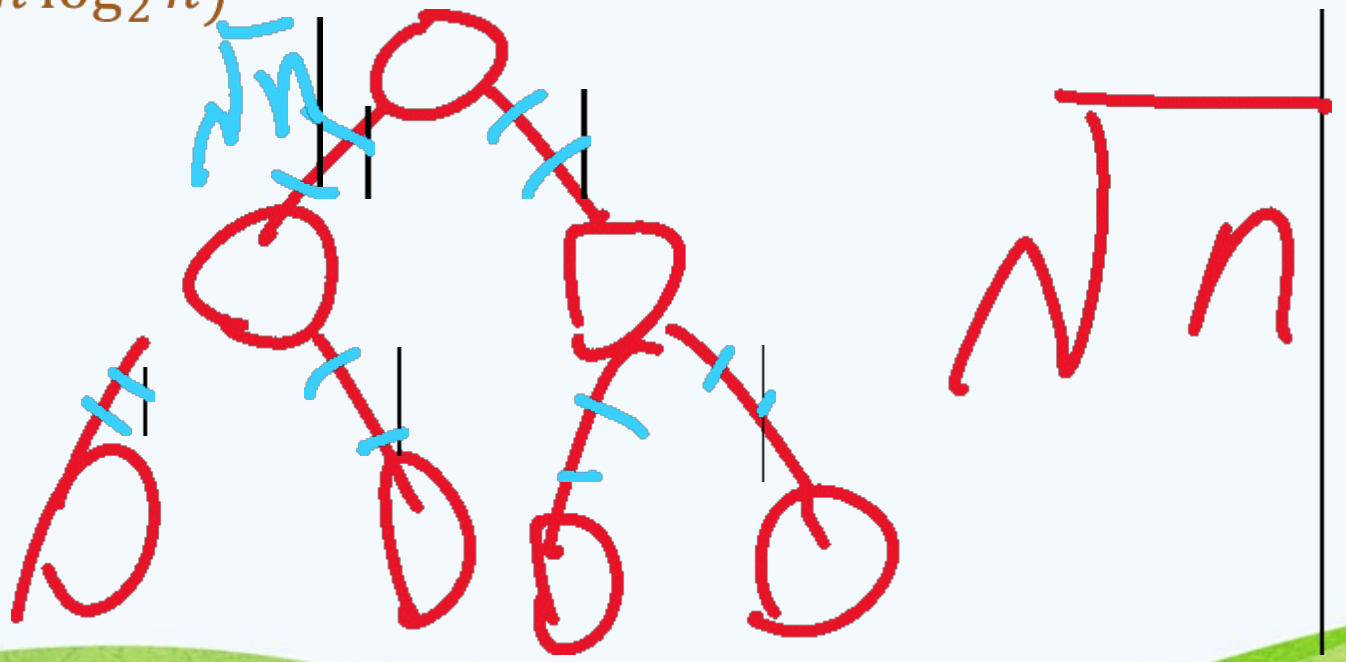
- 我们来看一个简单问题：链加，子树加，输出最后每个点的值。
- 对于子树加，我们可以先记在点上，最后求每个点到根的路径和，这个只要 dfs 即可。
- 对于链加，考虑先改成到根链加，那么最后只需要求每个点的子树和，这个也只要 dfs 即可。

重链剖分

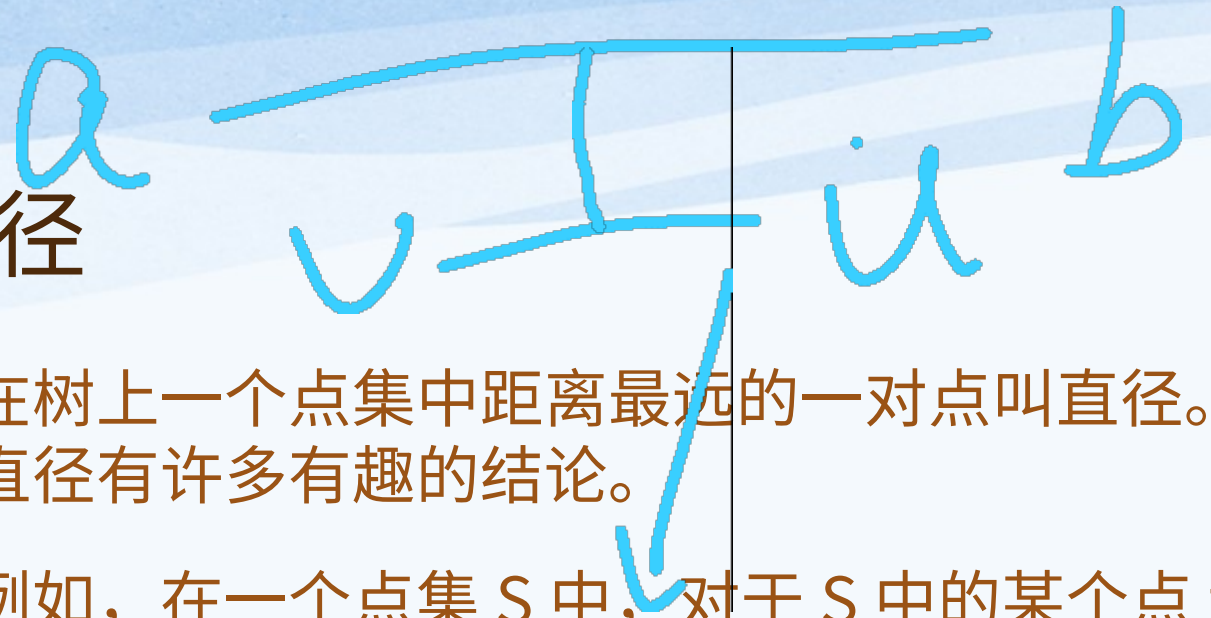
- 会做你就做全点
- 单点加，单点查询，链加，链求和，子树加，子树求和
- 考虑给每个点分配一个重孩子，由重孩子串成的链叫重链。 dfs 时先 dfs 重孩子，这样这种 dfs 序就可以满足把每个重链分配在一起。
- 注意到每个点往上只会有 O(log) 个重链段，所以我们可以把一条链分解成 O(log) 个 dfs 区间，每个子树也是一个 dfs 区间，那么就可以解决了。
- 注意树链剖分用线段树维护的复杂度是 O(log^2) 。即使你把每个重链分开建线段树也是 O(log^2) （考虑一个边长 sqrt 的 sqrt 个点的二叉树）。

卡掉树链剖分

- 先建一颗大小为n的二叉树
- 然后对每一条边拆成$\sqrt{n}$个点
- 时间复杂度$0\left(n\log_2^2 n\right)$

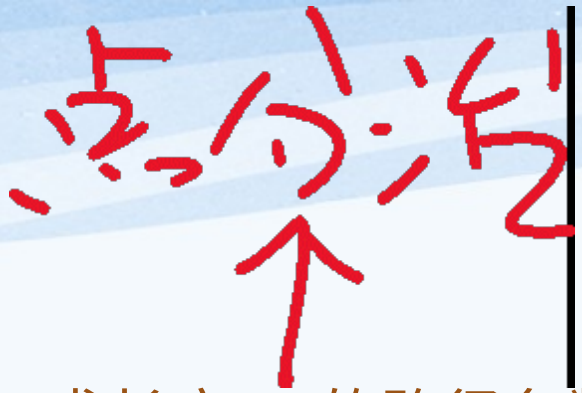


直径



- 在树上一个点集中距离最远的一对点叫直径。在边权非负的树中，关于直径有许多有趣的结论。
- 例如，在一个点集 S 中，对于 S 中的某个点 t ，对于另一个点 g ，我们一定有 dis(t,g) 的最大值等于 dis(t, 直径某端点) ，即在 g= 直径两端时会取到最大值。
- 另外，如果我们知道了点集 A 的直径和点集 B 的直径，我们就可以找到点集 (A∪B) 的直径，可以证明它的端点一定可以从 A 直径和 B 直径的端点中挑出。

长链剖分



- 我们来考虑一个问题，求长度 =t 的路径条数。
- 我们考虑记 f[i][j] 表示 i 的子树中从 i 开始往下 j 条边的链有几条（也就是子树中深度比 i 大 j 的点有几个），我们把最大深度最大的孩子定为 preferred child （长孩子？），然后我们每次先从 preferred child 拷贝 f 并挪一位，然后再把其他孩子的 f 拷过来，拷的时候顺便更新一下答案。
- 拷贝 preferred child 并不需要真的拷贝，只需要分配内存连续地分配即可（即像重链剖分那样分配内存）。注意到每次拷非 preferred child 的时候就相当于是在牺牲一段重链，所以复杂度是线性的。
- 一般解决树上 DP ，处理往下伸多少有影响

树上莫队

- 相信大家都会莫队。
- 考虑如何在树上莫队里表示一条链，树链剖分由于需要分解成 log 个部分不太适用。考虑使用两遍 dfs 序，对于一条祖先后代的链，除了链上的点都会被算零次或两次，而不是祖先后代的链除了链上和 lca 的点都会被算零次或两次。我们把 lca 补上即可。
- 设 f[i] 为一个点第一次出现， g[i] 为一个点第二次出现
- 如果是一条祖先链，就是找 f[u],f[v]
- 否则就是 g[u],f[v]+LCA(u,v)
- 那个 WC2013 糖果公园还没过… .

dsu on tree （链分治）

- 假设我们又有一个只能莫队的问题，是关于子树无修改询问的，但是数据范围有点大， 5e5 。
- 考虑要获取 x 子树的答案，我们进行重链剖分，先获取 x 轻子树的答案，并清除 x 轻子树的信息，接下来获取 x 重子树的答案，回溯时保留信息，再加入 x 轻子树的信息和 x 的信息并更新答案。
- 注意到一个点只会在它到祖先路径上的轻边处被更新，所以是一个 log 的（不考虑数据结构）。

线段树合并

- 比如每个点有个点权，我们要支持询问 x 的子树中点权 <=p 的点数，数据范围 5e5 ，不强制在线。
- 显然可以把所有询问离线下来拿个树状数组 dsu on tree ，两个 log 。
- 使用隐式线段树进行线段树合并，一个 log 。

点分治

- 满足以一个点为根，每个子树大小不超过点数的一半的点叫做重心。可以证明树一定有重心。（tips：反证法）
- 我们每次考虑当前树的重心，并处理过根的路径，然后把重心删掉递归各个子树。
- 例如仍然是处理长度 =t 的路径计数，我们把重心为根每个点的深度求出来并自卷积。但是这样无法处理两个点同在一个子树的情况，我们再对于每个子树扣掉即可。

边分治

- 有时候关于点分治不太方便，这时候我们可以使用边分治，即我们选取一个分的尽量平均的边把树分成两半。
- 可惜这个做法的复杂度并不对，考虑一个菊花树。
- 考虑把最大点度改小，我们可以使用左孩子 - 右兄弟对树进行重建。这样就可以进行边分治了。

动态点分治

- bzoj3730
- 有一棵不会动的树，每个点有点权，有 m 次操作，操作是改变一个点的点权，询问是询问和某个点距离不超过某个值的点权和。强制在线。
- 在点分治上套个数据结构！

动态动态点分治

- 紫荆花之恋
- 有一棵树，每次加一个叶子，每条边有边权，每个点还有点权 r ，求 dis(i,j)<=r[i]+r[j] 的点对数。强制在线。
- 如果不强制在线，可以直接建出点分树。
- 强制在线？替罪羊重构点分树。

Link cut tree

- 仍然是把树链组织成重链和轻链，每个点有不超过一个重孩子。
- 我们用 splay 组织各重链，并支持 access 操作：把一个点到根的路径变为重链，把它的孩子变为轻儿子，支持 makeroot 操作：把一个点变成根，这样就可以支持 link 、 cut 等操作并查询 / 修改路径信息。
- 动态维护子树信息的话需要用 top-tree （深坑）

虚树

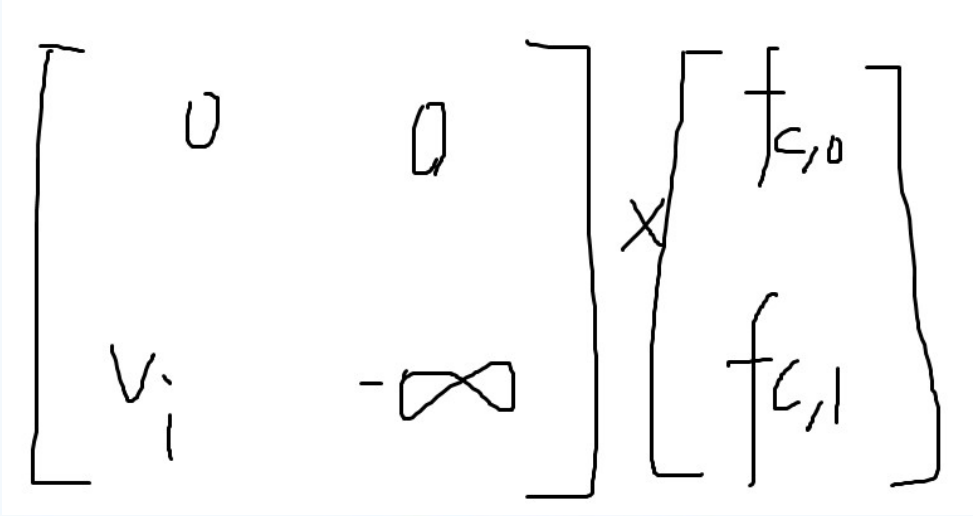
- 给一个点集 S，你要求出一个最小的点集 T 满足 S 是 T 的子集并且 T 中任意两点的 lca 仍然在 T 中。
- 把 S 中的点按 dfs 序排序之后是用栈维护。
- 此外动态虚树也是可以做的。

动态 dp

- 使用线段树进行维护，我们就解决了链的情况。
- 那么一般的树的情况，我们可以发现我们只需要把树重链剖分，每个重链拿个线段树维护，每次遇到轻边就 update 轻边上端点的信息。这样就可以做到两个 log 的复杂度了。

动态 dp

- 考虑一个简单的问题：树上带权最大独立集，我们可以写出 dp 式子：$f[i][0]=\Sigma \max(f[c][0],f[c][1]),f[i][1]=v[i]+\Sigma f[c][0]$。如果我们现在要支持修改 v 并动态询问带权最大独立集怎么办？
- 我们先考虑树是一条链的情况，我们就有 $f[i][0]=\max(f[c][0],f[c][1])$，$f[i][1]=v[i]+f[c][0]$。我们可以把它写成一个 add-max 矩阵乘法：



全局平衡二叉树

- 为什么 lct 和 top tree 可以做到一个 log，树剖只能做到两个 log？
- 仍然考虑树链剖分，但是重链用正常线段树维护太浪费了。我们考虑把重链上的点带上权，每个点的点权是子树大小（不算重子树），按这个点权建线段树。容易看出这样进行链剖就很靠谱了。此外这个结构也可以用于维护子树信息和动态 dp。

树上关键点

- 考虑如何分解链询问，有一个根号复杂度的做法。
- 考虑在树上随机选取 O(sqrtn) 个点和根，那么每个点往上期望走 sqrt 步就会走到一个关键点，一共 sqrt 个关键点，我们预处理关键点两两之间的信息，然后填上 O(sqrt) 个点就可以处理链询问了。
- 当然并不一定需要随机，我们只需要 dfs ，一旦最大子树深度 >sqrt 就选取当前点即可。

最小生成树

- 相信大家都会 kruskal 算法。
- 我们考虑一棵不在最小生成树上的边，我们有这条边的边权 >= 这条边覆盖的树边边权的最大值。如果这条边的边权 = 最大值，那么我们可以把最大值替换成这条边。

Kruskal 重构树

- 考虑 kruskal 算法的过程，我们可以建一棵树表示算法过程：一条长度为 e 的边合并了两个集合，我们就建一个新点，点权为 e ，孩子为这两个集合对应的点。这样就变成了一棵二叉树，原图中的每个点都是叶子。
- 这棵树有许多性质，例如两个点的 lca 就是这两个点在最小生成树中路径上边权最大的边。

运输计划

- 有一棵 n 个点的树，每条边有一个非负的边权。有 m 个运输计划，第 i 个运输计划是从某个点 u[i] 开始到达 v[i] ，需要花费的时间是这条路径上的边权和。完成所有运输计划需要花费的时间是每个运输计划花费的时间的最大值。
- 你可以选择一条边，把它的边权改为 0 ，求最少可能的花费时间。
- n,m<=300000 。

* Source: NOIP2015

运输计划

- 考虑二分答案 t，那么如果一条路径本来长度为 s，若 s>t 就需要在路径上找一条长度至少为 s-t 的边置为 0。
- 那么我们只需要把所有这些路径求交路径，然后就要求交路径里是否有长度为 max 这些值的边。
- 路径加 1，统计每条边的边权即可。树上差分。

树上的最远点对

- n 个点被 n-1 条边连接成了一颗树，每次询问给出 a~b 和 c~d 两个区间，表示点的标号，请你求出两个区间内各选一点之间的最大距离，即你需要求出 max{dis(i,j)|a<=i<=b,c<=j<=d} 。
- n,q<=100000 。

* Source: 51nod 1766

树上的最远点对

- 根据我们前面的结论，我们只需要求出每个区间对应的点集的直径并枚举两头求出答案即可。线段树 +lca 即可。

树点涂色

- 有一棵 n 个点的有根树，一开始在每个点上涂了颜色，并且每个点上的颜色不同。有三种操作：
- 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 y：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。
- 一共会进行 m 次操作。 1<=n,m<=100000 。

* Source: SDOI2017

树点涂色

- 容易发现， 1 操作和 lct 中的 access 操作类似，所以我们只需要用 access 即可模拟 1 操作。
- 2 操作的话，如果 x=1 就是 y 到根的段数，这个可以在 access 的时候拿个线段树维护。如果 x!=1 的话只需要在 lca 处讨论一下。
- 3 操作只需要在前述那个线段树里求区间 max 。

归程

- 有一个 n 个点 m 条边的无向连通图，每条边有两个属性：长度和海拔。当一场大小为 u 的雨降临时，海拔 <=u 的边都会积水。
- 有 q 天，每天 yazid 会从 v 号点出发回到 1 号点的家，该天下着大小为 p 的雨。 yazid 会在 v 号点骑一辆共享单车，但是共享单车不能经过有积水的边。 yazid 可以在任意节点下车并走路到 1 号点，他希望最小化自己走路的总长度。每天独立。强制在线。
- n,m,q<=400000 。

* Source: NOI2018

归程

- 考虑从 v 号点出发能走到的点，那么就是要求只经过海拔 >u 的边能走到的点集。我们建出最大生成树的 kruskal 重构树，那么对应的就是最后一个点权 >u 的祖先对应的子树。
- 我们要问的就是这个子树中的点到 1 号点距离的最小值。我们跑完 dijkstra 树形 dp 求出来即可。找到最后一个 >u 的祖先可以用倍增。

圣诞树

- ξ 得到了一棵圣诞树，ξ 会事先进行 m 个操作，每次在一条链（u[i],v[i]）上的每个点上挂上 a[i] 个种类为 b[i] 的礼物。
- 一个点的 k- 美观度这样计算：把这个点上的所有种类的礼物按照个数从小到大排序，如果个数一样就按照种类从小到大排。它的 k- 美观度就是排好序后前 k 种礼物种类的 xor 值（如果礼物种类不足 k 种，就把这个点上所有礼物的种类 xor 起来）。
- 挂完礼物后给 q 个询问，每次给定 w[i],k[i]，求点 w[i] 的 k[i]- 美观度。
- 所有输入的数 <=100000 。

* source: 51nod 1782

圣诞树

- 考虑对于一个询问，改为在 u[i] 和 v[i] 挂 (a,b) ， lca 和 fa[lca] 挂 (-a, b) ，接下来我们就可以把询问转化为子树询问，使用你喜爱的数据结构套 dsu on tree 维护即可。

Monkey and Tree

- 有一棵带正边权的树，给若干点对 (ai,bi) 。
- 你需要找到两对点对 i 和 j ，使得 dis(ai,aj)+dis(bi,bj) 最大。
- n<=500000 。

* Source: 2017 集训队作业自选题 加强

Monkey and Tree

- 对于 (ai,bi) ，新建一个点连向 ai ，边权为 dep[ai] 。把这个新点记在 a i 上。
- 考虑 lca(ai,aj) ，我们就有 dis(ai,aj)=dep[ai]+dep[aj]-2dep[lca] ，我们在 lca 处求这些新点跨子树的直径即可。
- 此外也有其他神奇的做法。
- 能不能再给力一点?

通道

- 有三棵 n 个点的树 T1 ， T2 ， T3 ，你需要求出一对点 a,b 使得 T1.dis(a,b)+T2.dis(a,b)+T3.dis(a,b) ，三棵树的边权均非负。
- n<=100000 。

* Source: WC2018

通道

- 如果 T3 是链，我们可以在链上分治，这样就相当于是在中点左右各找一个点，使得 T1.dis(a,b)+T2.dis(a,b)+l[a]+l[b] 最大，其中 l 是到中点的距离。
- 上一题的做法可以继续使用，但是我们没法枚举 T1 里的所有点，但是我们可以把要用的点建虚树，这样做法就可以接着用了。
- T3 是树的情况，我们使用左孩子右兄弟法重建之后边分治即可。

Master of Connected Component

- 给定两棵有 n 个节点的树，每个节点上有一对数 (x,y) ，表示图 G 中的一条边。对于每一个 x ，求出两棵树中 x 到根路径上所有边在图 G 中构成的子图的连通块个数。
- 多组数据， n<=10000 。

* Source: CCPC 2017 Hangzhou

Master of Connected Component

- 考虑树上关键点的做法，我们在第一棵树中提取关键点，使得每个点往上走不超过 sqrt 步就会走到关键点。我们对于每个关键点处理最近点为这个关键点的询问。
- 我们维护一个可撤销并查集并 dfs 第二棵树， dfs 到对应询问的点时，我们加入第一棵树中缺少的 O(sqrt) 个点即可。

城池攻占

- 有一棵 n 个点的树，每个点 i 有值 h[i] ， a[i] ， v[i]>=0 。
- 有 m 个骑士，初始攻击力 x[i] ，从某个节点出发向根节点前进。
- 每到达一个节点，如果当前骑士的攻击力 < 该节点的 h ，那么这个骑士牺牲，否则若 a[i]=0 ，攻击力加上 v[i] ，否则乘上 v[i] 。
- 求每个骑士牺牲的城市。
- 保证任意骑士任意时刻攻击力不超过 10^18 。
- n,m<=100000 。

* Source: JLOI2015, 有简化

城池攻占

- 注意到如果两个骑士经过了同一个节点，那么攻击力大小关系不会改变。
- 从下而上模拟，我们对于每个点维护一个小根堆，每次把孩子启发式合并，加和乘在堆上打标记维护即可。