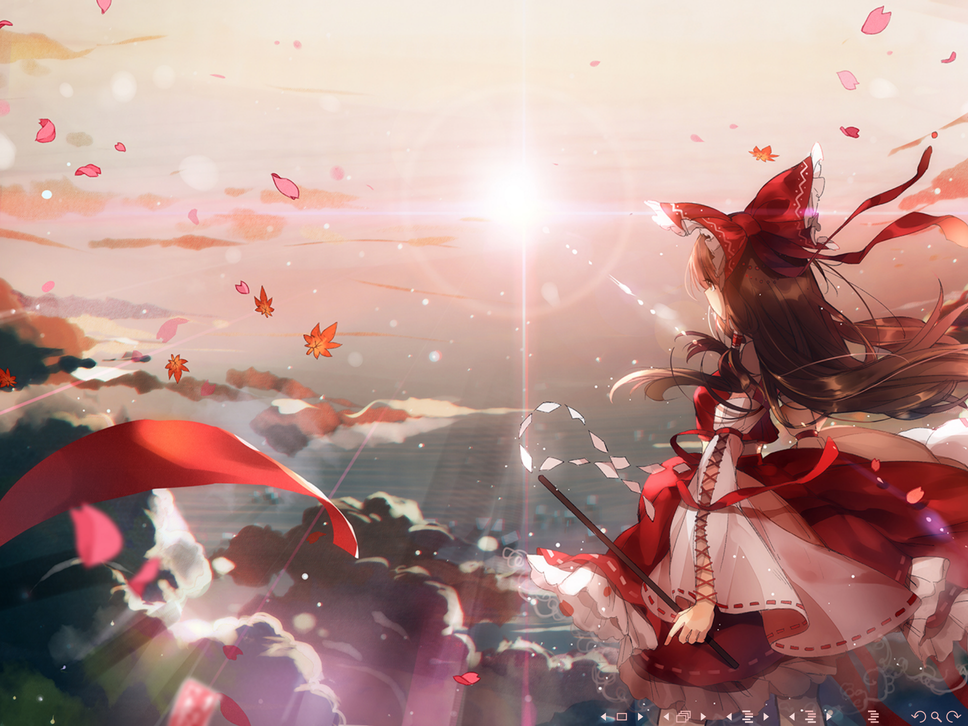


数据结构水题选讲

AwD

杭州学军中学

March 26, 2019

简介

简介

- 我的电竞水平不是很高

简介

- 我的电竞水平不是很高
- 在座的许多人能吊打我

简介

- 我的电竞水平不是很高
- 在座的许多人能吊打我
- 最优化问题是讲不来的

简介

- 我的电竞水平不是很高
- 在座的许多人能吊打我
- 最优化问题是讲不来的
- 因此只剩数据结构题了

简介

- 我的电竞水平不是很高
- 在座的许多人能吊打我
- 最优化问题是讲不来的
- 因此只剩数据结构题了
- 这些题都是比较简单的

简介

- 我的电竞水平不是很高
- 在座的许多人能吊打我
- 最优化问题是讲不来的
- 因此只剩数据结构题了
- 这些题都是比较简单的
- 欢迎各位神仙上台秒题

简介

- 我的电竞水平不是很高
- 在座的许多人能吊打我
- 最优化问题是讲不来的
- 因此只剩数据结构题了
- 这些题都是比较简单的
- 欢迎各位神仙上台秒题
- 退役选手的自我修养「耶」

约定

约定

- 若不加注明，Q 指代操作的数量。

约定

- 若不加注明，Q 指代操作的数量。
- 若不加注明，字母所指代变量的值可视为 int64 范围。

约定

- 若不加注明，Q 指代操作的数量。
- 若不加注明，字母所指代变量的值可视为 int64 范围。
- 若不加注明，复杂度中默认 N, Q 同阶，即，两者会混用。

约定

- 若不加注明，Q 指代操作的数量。
- 若不加注明，字母所指代变量的值可视为 int64 范围。
- 若不加注明，复杂度中默认 N, Q 同阶，即，两者会混用。
- 标算都不是压位，若存在压位做法，欢迎大家上台分享。

约定

- 若不加注明，Q 指代操作的数量。
- 若不加注明，字母所指代变量的值可视为 int64 范围。
- 若不加注明，复杂度中默认 N, Q 同阶，即，两者会混用。
- 标算都不是压位，若存在压位做法，欢迎大家上台分享。
- 数据范围只起参考作用，可以假装时限都是 NOI 的 $3s$。

Problem A

Problem A

- 一个长度为 N 的序列。

Problem A

- 一个长度为 N 的序列。
- 多次询问：$[l, r]$ 中第 k 小的没有出现过的正整数。

Problem A

- 一个长度为 N 的序列。
- 多次询问：$[l, r]$ 中第 k 小的没有出现过的正整数。
- $N, k \leq 10^6$；$Q \leq 10^4$。

Problem A

- 一个长度为 N 的序列。
- 多次询问：$[l, r]$ 中第 k 小的没有出现过的正整数。
- $N, k \le 10^6$；$Q \le 10^4$。
- 来自 ZJOI2018 队长的原创题～

Problem A：题解

Problem A：题解

- 考虑直接莫队。

Problem A：题解

- 考虑直接莫队。
- 显然答案最多只有 $N+k$。

Problem A：题解

- 考虑直接莫队。
- 显然答案最多只有 $N+k$。
- 对值域分块后可以做到 $O(1)$ 移动指针。

Problem A：题解

- 考虑直接莫队。
- 显然答案最多只有 $N+k$。
- 对值域分块后可以做到 $O(1)$ 移动指针。
- 时间复杂度：$O(N\sqrt{Q}+Q\sqrt{N+k})$。

K 小值查询

K 小值查询

- 维护一个 N 个元素的集合。

K 小值查询

- 维护一个 N 个元素的集合。
- 支持两种操作：

K 小值查询

- 维护一个 N 个元素的集合。
- 支持两种操作：
- 1. 对于所有大于 k 的数，将其减去 k。

K 小值查询

- 维护一个 N 个元素的集合。
- 支持两种操作：
- 1. 对于所有大于 k 的数，将其减去 k。
- 2. 询问第 k 小的数。

K 小值查询

- 维护一个 N 个元素的集合。
- 支持两种操作：
- 1. 对于所有大于 k 的数，将其减去 k。
- 2. 询问第 k 小的数。
- $N, Q \leq 10^5$。

K 小值查询

- 维护一个 N 个元素的集合。
- 支持两种操作：
- 1. 对于所有大于 k 的数，将其减去 k。
- 2. 询问第 k 小的数。
- $N, Q \leq 10^5$。
- BZOJ

K 小值查询：题解

K 小值查询：题解

- 对较大的数打上减法标记后平衡树启发式合并即可。

K 小值查询：题解

- 对较大的数打上减法标记后平衡树启发式合并即可。
- 复杂度证明?

K 小值查询：题解

- 对较大的数打上减法标记后平衡树启发式合并即可。
- 复杂度证明？
- 每次只有大小在 $(k, 2k]$ 这个区间里的数会被重新插入，插入后其大小减半。

K 小值查询：题解

- 对较大的数打上减法标记后平衡树启发式合并即可。
- 复杂度证明？
- 每次只有大小在 $(k, 2k]$ 这个区间里的数会被重新插入，插入后其大小减半。
- 每个数最多插入 $\log w$ 次。

K 小值查询：题解

- 对较大的数打上减法标记后平衡树启发式合并即可。
- 复杂度证明？
- 每次只有大小在 $(k, 2k]$ 这个区间里的数会被重新插入，插入后其大小减半。
- 每个数最多插入 $\log w$ 次。
- 时间复杂度：$O(N \log N \log w)$。

“大-魔-法-师”

“大-魔-法-师”

- 一棵 N 个点的树，每个节点可能是黑的，也可能是白的。最初，所有的节点都是白的。

“大-魔-法-师”

- 一棵 N 个点的树，每个节点可能是黑的，也可能是白的。最初，所有的节点都是白的。
- 多次操作：翻转某个节点的颜色，即将白色改为黑色，黑色改为白色。

“大-魔-法-师”

- 一棵 N 个点的树，每个节点可能是黑的，也可能是白的。最初，所有的节点都是白的。
- 多次操作：翻转某个节点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，由所有黑色节点构成的虚树有几个叶子节点。

“大-魔-法-师”

- 一棵 N 个点的树，每个节点可能是黑的，也可能是白的。最初，所有的节点都是白的。
- 多次操作：翻转某个节点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，由所有黑色节点构成的虚树有几个叶子节点。
- $N, Q \le 10^5$。

“大-魔-法-师”

- 一棵 N 个点的树，每个节点可能是黑的，也可能是白的。最初，所有的节点都是白的。
- 多次操作：翻转某个节点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，由所有黑色节点构成的虚树有几个叶子节点。
- $N, Q \le 10^5$。
- 计蒜之道 2018

“大-魔-法-师”

- 一棵 N 个点的树，每个节点可能是黑的，也可能是白的。最初，所有的节点都是白的。
- 多次操作：翻转某个节点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，由所有黑色节点构成的虚树有几个叶子节点。
- $N, Q \leq 10^5$。
- 计蒜之道 2018
- 题意稍有修改。（原来的问题外面还有一层圆方树……）

“大-魔-法-师”：题解

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：
- 1. 所有节点的 LCA。

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：
- 1. 所有节点的 LCA。
- 2. 子树里没有黑点的黑点。

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：
- 1. 所有节点的 LCA。
- 2. 子树里没有黑点的黑点。
- 第一种显然很好判断。

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：
- 1. 所有节点的 LCA。
- 2. 子树里没有黑点的黑点。
- 第一种显然很好判断。
- 对于第二种情况：

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：
- 1. 所有节点的 LCA。
- 2. 子树里没有黑点的黑点。
- 第一种显然很好判断。
- 对于第二种情况：
- 考虑树链剖分，一条重链上最多只会有一个叶子节点，直接维护即可。

“大-魔-法-师”：题解

- 按时间分治去掉删除操作。
- 考虑虚树上怎样的节点会是叶子节点：
- 1. 所有节点的 LCA。
- 2. 子树里没有黑点的黑点。
- 第一种显然很好判断。
- 对于第二种情况：
- 考虑树链剖分，一条重链上最多只会有一个叶子节点，直接维护即可。
- 时间复杂度：$O(N\log^2 N)$。

抓 fafa 的 qzh

抓 fafa 的 qzh

- 维护一棵 N 个结点的无根树。

抓 fafa 的 qzh

- 维护一棵 N 个结点的无根树。
- 支持两种操作：

抓 fafa 的 qzh

- 维护一棵 N 个结点的无根树。
- 支持两种操作：
- 1. 在链 (u, v) 的每个点上放一个可爱值为 k 的 fafa。

抓 fafa 的 qzh

- 维护一棵 N 个结点的无根树。
- 支持两种操作:
- 1. 在链 (u, v) 的每个点上放一个可爱值为 k 的 fafa。
- 2. 问所有可爱值在 $[l, r]$ 内的 fafa 到点 p 的距离之和。

抓 fafa 的 qzh

- 维护一棵 N 个结点的无根树。
- 支持两种操作:
- 1. 在链 (u, v) 的每个点上放一个可爱值为 k 的 fafa。
- 2. 问所有可爱值在 $[l, r]$ 内的 fafa 到点 p 的距离之和。
- $N, Q \leq 10^5$ 。

抓 fafa 的 qzh

- 维护一棵 N 个结点的无根树。
- 支持两种操作:
- 1. 在链 (u, v) 的每个点上放一个可爱值为 k 的 fafa。
- 2. 问所有可爱值在 $[l, r]$ 内的 fafa 到点 p 的距离之和。
- $N, Q \le 10^5$ 。
- Wannafly 挑战赛 13

抓 fafa 的 qzh：一个离线做法

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：
- 维护一个点集，支持加链删链，询问一个点到点集内所有点的距离和。

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：
- 维护一个点集，支持加链删链，询问一个点到点集内所有点的距离和。
- 只要算出了 LCA 深度之和就可以得到距离和。

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：
- 维护一个点集，支持加链删链，询问一个点到点集内所有点的距离和。
- 只要算出了 LCA 深度之和就可以得到距离和。
- 求 A 与 B 的 LCA 深度可以通过对从 A 到根的链上每个点加一然后询问 B 到根的链上每个点的和解决。

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：
- 维护一个点集，支持加链删链，询问一个点到点集内所有点的距离和。
- 只要算出了 LCA 深度之和就可以得到距离和。
- 求 A 与 B 的 LCA 深度可以通过对从 A 到根的链上每个点加一然后询问 B 到根的链上每个点的和解决。
- 求一个点集 S 与 B 的 LCA 深度之和也可以通过类似的方法解决。

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：
- 维护一个点集，支持加链删链，询问一个点到点集内所有点的距离和。
- 只要算出了 LCA 深度之和就可以得到距离和。
- 求 A 与 B 的 LCA 深度可以通过对从 A 到根的链上每个点加一然后询问 B 到根的链上每个点的和解决。
- 求一个点集 S 与 B 的 LCA 深度之和也可以通过类似的方法解决。
- 因此原问题只需要链加等差数列，询问链和即可。

抓 fafa 的 qzh：一个离线做法

- 按可爱值建线段树，仿照时间线段树，可以得到这样一个子问题：
- 维护一个点集，支持加链删链，询问一个点到点集内所有点的距离和。
- 只要算出了 LCA 深度之和就可以得到距离和。
- 求 A 与 B 的 LCA 深度可以通过对从 A 到根的链上每个点加一然后询问 B 到根的链上每个点的和解决。
- 求一个点集 S 与 B 的 LCA 深度之和也可以通过类似的方法解决。
- 因此原问题只需要链加等差数列，询问链和即可。
- 时间复杂度：$O(N\log^2 N)$。

抓 fafa 的 qzh：一个在线做法

抓 fafa 的 qzh：一个在线做法

- 如果直接魔改离线做法，可以得到一个空间为 $O(N\log N\log w)$ 的树套树做法。

抓 fafa 的 qzh：一个在线做法

- 如果直接魔改离线做法，可以得到一个空间为 $O(N \log N \log w)$ 的树套树做法。
- 考虑先修改后询问的问题，显然可以按可爱值建可持久化树剖解决。

抓 fafa 的 qzh：一个在线做法

- 如果直接魔改离线做法，可以得到一个空间为 $O(N \log N \log w)$ 的树套树做法。
- 考虑先修改后询问的问题，显然可以按可爱值建可持久化树剖解决。
- 对于原问题，显然可以通过对时间分块，定期重构树剖解决。

抓 fafa 的 qzh：一个在线做法

- 如果直接魔改离线做法，可以得到一个空间为 $O(N\log N\log w)$ 的树套树做法。
- 考虑先修改后询问的问题，显然可以按可爱值建可持久化树剖解决。
- 对于原问题，显然可以通过对时间分块，定期重构树剖解决。
- 时间复杂度：$O(N^{1.5}\log^{0.5} N)$ 。

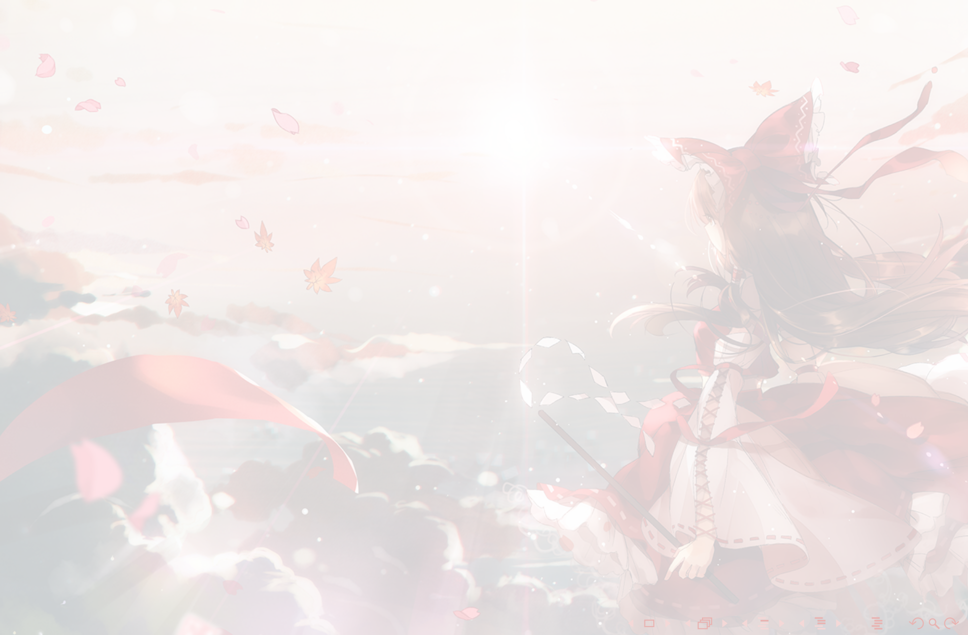
抓 fafa 的 qzh：一个在线做法

- 如果直接魔改离线做法，可以得到一个空间为 $O(N\log N\log w)$ 的树套树做法。
- 考虑先修改后询问的问题，显然可以按可爱值建可持久化树剖解决。
- 对于原问题，显然可以通过对时间分块，定期重构树剖解决。
- 时间复杂度：$O(N^{1.5}\log^{0.5}N)$ 。
- 空间复杂度：$O(N\log N)$。

抓 fafa 的 qzh：一个在线做法

- 如果直接魔改离线做法，可以得到一个空间为 $O(N \log N \log w)$ 的树套树做法。
- 考虑先修改后询问的问题，显然可以按可爱值建可持久化树剖解决。
- 对于原问题，显然可以通过对时间分块，定期重构树剖解决。
- 时间复杂度：$O(N^{1.5} \log^{0.5} N)$ 。
- 空间复杂度：$O(N \log N)$。
- 当然，还有一个时间复杂度为 $O(N^{\frac{5}{3}})$ 的在线做法，这里不再赘述。

Product on the segment



Product on the segment

- 给定一个长为 N 的数组，还有一个常数 MOD。

Product on the segment

- 给定一个长为 N 的数组，还有一个常数 MOD。
- 多次询问，区间之积对 MOD 取模的结果。

Product on the segment

- 给定一个长为 N 的数组，还有一个常数 MOD。
- 多次询问，区间之积对 MOD 取模的结果。
- $N, Q \leq 10^7$。

Product on the segment

- 给定一个长为 N 的数组，还有一个常数 MOD。
- 多次询问，区间之积对 MOD 取模的结果。
- $N, Q \leq 10^7$。
- 强制在线。

Product on the segment

- 给定一个长为 N 的数组，还有一个常数 MOD。
- 多次询问，区间之积对 MOD 取模的结果。
- $N, Q \leq 10^7$。
- 强制在线。
- Codechef November Challenge 2017

Product on the segment：题解

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。
- 考虑将数组分为 $\frac{N}{\log N}$ 块，块内预处理前后缀和，对于整块依旧采用分治。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。
- 考虑将数组分为 $\frac{N}{\log N}$ 块，块内预处理前后缀和，对于整块依旧采用分治。
- 这样预处理的复杂度就降到了 $O(N)$，只是这样没法计算块内的答案了。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。
- 考虑将数组分为 $\frac{N}{\log N}$ 块，块内预处理前后缀和，对于整块依旧采用分治。
- 这样预处理的复杂度就降到了 $O(N)$，只是这样没法计算块内的答案了。
- 注意到块内的问题与原问题是一样的，直接套用原方法维护。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。
- 考虑将数组分为 $\frac{N}{\log N}$ 块，块内预处理前后缀和，对于整块依旧采用分治。
- 这样预处理的复杂度就降到了 $O(N)$，只是这样没法计算块内的答案了。
- 注意到块内的问题与原问题是一样的，直接套用原方法维护。
- 于是就有 $T(N)=\frac{N}{\log N}T(\log N)+O(N)$。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。
- 考虑将数组分为 $\frac{N}{\log N}$ 块，块内预处理前后缀和，对于整块依旧采用分治。
- 这样预处理的复杂度就降到了 $O(N)$，只是这样没法计算块内的答案了。
- 注意到块内的问题与原问题是一样的，直接套用原方法维护。
- 于是就有 $T(N)=\frac{N}{\log N}T(\log N)+O(N)$。
- 解得 $T(N)=O(N\log^* N)$，实际应用中 $\log^* N\le 5$，可视为线性。

Product on the segment：题解

- 如果 MOD 是质数，有一个基于 $O(N)$ 求每个数逆元的数论做法。
- 一个很自然的想法是对数组分治，这样可以得到一个时间复杂度为 $O(N\log N+Q)$ 的做法。
- 考虑将数组分为 $\frac{N}{\log N}$ 块，块内预处理前后缀和，对于整块依旧采用分治。
- 这样预处理的复杂度就降到了 $O(N)$，只是这样没法计算块内的答案了。
- 注意到块内的问题与原问题是一样的，直接套用原方法维护。
- 于是就有 $T(N)=\frac{N}{\log N}T(\log N)+O(N)$。
- 解得 $T(N)=O(N\log^* N)$，实际应用中 $\log^* N\le 5$，可视为线性。
- 时间复杂度：$O(N\log^* N+Q)$。

Reverse Game

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。
- 支持两种操作:

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。
- 支持两种操作：
- 1. 将某一列异或上 1。

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。
- 支持两种操作：
- 1. 将某一列异或上 1。
- 2. 将某个位置异或上 1。

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。
- 支持两种操作：
- 1. 将某一列异或上 1。
- 2. 将某个位置异或上 1。
- 在每个操作后，问，各有多少个 0 与 1 的联通块。

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。
- 支持两种操作：
- 1. 将某一列异或上 1。
- 2. 将某个位置异或上 1。
- 在每个操作后，问，各有多少个 0 与 1 的联通块。
- $N \leq 200$；$Q \leq 20000$。

Reverse Game

- 给定一个大小为 $N \times N$ 的 01 矩阵。
- 支持两种操作：
- 1. 将某一列异或上 1。
- 2. 将某个位置异或上 1。
- 在每个操作后，问，各有多少个 0 与 1 的联通块。
- $N \leq 200$；$Q \leq 20000$。
- 2018 Multi-University Training Contest 7

Reverse Game：题解

Reverse Game：题解

- 对列建线段树，维护区间左右侧之间各点的联通性。

Reverse Game：题解

- 对列建线段树，维护区间左右侧之间各点的联通性。
- 上传时暴力合并并查集即可。

Reverse Game：题解

- 对列建线段树，维护区间左右侧之间各点的联通性。
- 上传时暴力合并并查集即可。
- 时间复杂度：$O(NQ\log N)$。

Problem B

Problem B

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。

Problem B

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支付 1 的代价可以将一个长度为 L 的区间染白。

Problem B

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支付 1 的代价可以将一个长度为 L 的区间染白。
- 多次操作：翻转某个整点的颜色，即将白色改为黑色，黑色改为白色。

Problem B

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支付 1 的代价可以将一个长度为 L 的区间染白。
- 多次操作：翻转某个整点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，至少需要付出多少代价，才能将整个数轴染白。

Problem B

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支付 1 的代价可以将一个长度为 L 的区间染白。
- 多次操作：翻转某个整点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，至少需要付出多少代价，才能将整个数轴染白。

- $Q \le 5 \times 10^5$。

Problem B

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支付 1 的代价可以将一个长度为 L 的区间染白。
- 多次操作：翻转某个整点的颜色，即将白色改为黑色，黑色改为白色。
- 在每个操作后，问，至少需要付出多少代价，才能将整个数轴染白。
- $Q \le 5 \times 10^5$。
- 经典问题

Problem B：题解

Problem B：题解

- 显然，一个简单的染色方案是贪心的从左向右染。

Problem B：题解

- 显然，一个简单的染色方案是贪心的从左向右染。
- 当染了黑点 x 后接下来需要染的是最小的大于 $x + L$ 的黑点。

Problem B：题解

- 显然，一个简单的染色方案是贪心的从左向右染。
- 当染了黑点 x 后接下来需要染的是最小的大于 $x+L$ 的黑点。
- 这样的关系形成了一棵树的结构。

Problem B：题解

- 显然，一个简单的染色方案是贪心的从左向右染。
- 当染了黑点 x 后接下来需要染的是最小的大于 $x + L$ 的黑点。
- 这样的关系形成了一棵树的结构。
- 那么答案就是第一个黑点到无穷远处的黑点之间的距离。

Problem B：题解

- 显然，一个简单的染色方案是贪心的从左向右染。
- 当染了黑点 x 后接下来需要染的是最小的大于 $x + L$ 的黑点。
- 这样的关系形成了一棵树的结构。
- 那么答案就是第一个黑点到无穷远处的黑点之间的距离。
- 但是在修改的时候，在这棵树上改动的不只有一条边。

Problem B：题解

Problem B：题解

- 考虑引入辅助点，对于每个黑点 x，在 $x+L$ 处设立一个灰点。

Problem B：题解

- 考虑引入辅助点，对于每个黑点 x，在 $x+L$ 处设立一个灰点。
- 黑点 x 的出边指向灰点 $x+L$。

Problem B：题解

- 考虑引入辅助点，对于每个黑点 x，在 $x+L$ 处设立一个灰点。
- 黑点 x 的出边指向灰点 $x+L$。
- 灰点 $x+L$ 的出边指向右侧最近的一个黑点或灰点。

Problem B：题解

- 考虑引入辅助点，对于每个黑点 x，在 $x+L$ 处设立一个灰点。
- 黑点 x 的出边指向灰点 $x+L$。
- 灰点 $x+L$ 的出边指向右侧最近的一个黑点或灰点。
- 显然，答案就是第一个黑点与无穷远处的黑点之间的这条链上的黑点数。

Problem B：题解

- 考虑引入辅助点，对于每个黑点 x，在 $x+L$ 处设立一个灰点。
- 黑点 x 的出边指向灰点 $x+L$。
- 灰点 $x+L$ 的出边指向右侧最近的一个黑点或灰点。
- 显然，答案就是第一个黑点与无穷远处的黑点之间的这条链上的黑点数。
- 每次修改的时候只会修改 $O(1)$ 个点的出边，动态树维护即可。

Problem B：题解

- 考虑引入辅助点，对于每个黑点 x，在 $x+L$ 处设立一个灰点。
- 黑点 x 的出边指向灰点 $x+L$。
- 灰点 $x+L$ 的出边指向右侧最近的一个黑点或灰点。
- 显然，答案就是第一个黑点与无穷远处的黑点之间的这条链上的黑点数。
- 每次修改的时候只会修改 $O(1)$ 个点的出边，动态树维护即可。
- 时间复杂度 $O(N\log N)$。

动态树练习题 I

动态树练习题 I

- 有一棵 N 个节点的树，每个节点上有点权 v_i。

动态树练习题 I

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持三种操作：

动态树练习题 I

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持三种操作：
- 1. 链加。

动态树练习题 I

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持三种操作：
- 1. 链加。
- 2. 链上求和。

动态树练习题 I

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持三种操作：
- 1. 链加。
- 2. 链上求和。
- 3. 删除一条边后加上一条边，保证操作后依然为树。

动态树练习题 I

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持三种操作：
- 1. 链加。
- 2. 链上求和。
- 3. 删除一条边后加上一条边，保证操作后依然为树。
- 经典问题

动态树练习题 I：题解

动态树练习题 I：题解

- 直接使用 LCT 维护即可。

动态树练习题 I：题解

- 直接使用 LCT 维护即可。
- 时间复杂度 $O(N \log N)$。

动态树练习题 II

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持四种操作：

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持四种操作：
- 1. 链加。

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持四种操作：
- 1. 链加。
- 2. 链上求和。

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持四种操作：
- 1. 链加。
- 2. 链上求和。
- 3. 子树求和。

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持四种操作：
- 1. 链加。
- 2. 链上求和。
- 3. 子树求和。
- 4. 删除一条边后加上一条边，保证操作后依然为树。

动态树练习题 II

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持四种操作：
- 1. 链加。
- 2. 链上求和。
- 3. 子树求和。
- 4. 删除一条边后加上一条边，保证操作后依然为树。
- 经典问题

动态树练习题 II：题解

动态树练习题 II：题解

- 这个问题仍然能用 LCT 维护～

动态树练习题 II：题解

- 这个问题仍然能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的点权之和。

动态树练习题 II：题解

- 这个问题仍然能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的点权之和。
- 切换虚实边的时候修改虚孩子的点权之和即可。

动态树练习题 II：题解

- 这个问题仍然能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的点权之和。
- 切换虚实边的时候修改虚孩子的点权之和即可。
- 时间复杂度：$O(N\log N)$。

动态树练习题 III

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上求和。

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上求和。
- 4. 子树求和。

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上求和。
- 4. 子树求和。
- 5. 删除一条边后加上一条边，保证操作后依然为树。

动态树练习题 III

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上求和。
- 4. 子树求和。
- 5. 删除一条边后加上一条边，保证操作后依然为树。
- 经典问题

动态树练习题 III：题解

动态树练习题 III：题解

- 这个问题依旧能用 LCT 维护～

动态树练习题 III：题解

- 这个问题依旧能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的全局标记，对于每个虚孩子单独维护一个标记，表示其实际标记与全局标记的差：

动态树练习题 III：题解

- 这个问题依旧能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的全局标记，对于每个虚孩子单独维护一个标记，表示其实际标记与全局标记的差：
- 将一条虚边切换为实边的时候，将两个标记的和作为标记打上去。

动态树练习题 III：题解

- 这个问题依旧能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的全局标记，对于每个虚孩子单独维护一个标记，表示其实际标记与全局标记的差：
- 将一条虚边切换为实边的时候，将两个标记的和作为标记打上去。
- 将一条实边切换为虚边的时候，将其自身的标记赋为全局标记的相反数。

动态树练习题 III：题解

- 这个问题依旧能用 LCT 维护～
- 对于 LCT 的每个节点，维护其虚孩子的全局标记，对于每个虚孩子单独维护一个标记，表示其实际标记与全局标记的差：
- 将一条虚边切换为实边的时候，将两个标记的和作为标记打上去。
- 将一条实边切换为虚边的时候，将其自身的标记赋为全局标记的相反数。
- 时间复杂度：$O(N\log N)$。

动态树练习题 IV

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上最大值。

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上最大值。
- 4. 子树最大值。

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上最大值。
- 4. 子树最大值。
- 5. 删除一条边后加上一条边，保证操作后依然为树。

动态树练习题 IV

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上最大值。
- 4. 子树最大值。
- 5. 删除一条边后加上一条边，保证操作后依然为树。
- 经典问题

动态树练习题 IV：题解

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：
- 注：这里的 AAA-tree 与集训队论文中所探讨的 AAA-tree 稍有不同。

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：
- 注：这里的 AAA-tree 与集训队论文中所探讨的 AAA-tree 稍有不同。
- AAA-tree 是一种基于 LCT 的简单数据结构。

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：
- 注：这里的 AAA-tree 与集训队论文中所探讨的 AAA-tree 稍有不同。
- AAA-tree 是一种基于 LCT 的简单数据结构。
- 对于每个节点，使用一棵 splay 去维护它的所有出边（不只是虚边）。

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：
- 注：这里的 AAA-tree 与集训队论文中所探讨的 AAA-tree 稍有不同。
- AAA-tree 是一种基于 LCT 的简单数据结构。
- 对于每个节点，使用一棵 splay 去维护它的所有出边（不只是虚边）。
- 称其维护节点的树为实 splay。

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：
- 注：这里的 AAA-tree 与集训队论文中所探讨的 AAA-tree 稍有不同。
- AAA-tree 是一种基于 LCT 的简单数据结构。
- 对于每个节点，使用一棵 splay 去维护它的所有出边（不只是虚边）。
- 称其维护节点的树为实 splay。
- 称其维护出边的树为虚 splay。

动态树练习题 IV：题解

- 这时候 LCT 上就必须得用一些东西去维护虚孩子了～
- 接下来将简单的探讨一下 AAA-tree：
- 注：这里的 AAA-tree 与集训队论文中所探讨的 AAA-tree 稍有不同。
- AAA-tree 是一种基于 LCT 的简单数据结构。
- 对于每个节点，使用一棵 splay 去维护它的所有出边（不只是虚边)。
- 称其维护节点的树为实 splay。
- 称其维护出边的树为虚 splay。
- 一张 N 个点 M 条边的森林里有 N 个实节点与 $2M$ 个虚节点。

动态树练习题 IV：题解

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。
- *real* 指向了其所维护以其为根重链的重 splay。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。
- *real* 指向了其所维护以其为根重链的重 splay。
- 特殊的，如果这条出边在 AAA-tree 中是实孩子或父亲，则这个指针为空。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。
- *real* 指向了其所维护以其为根重链的重 splay。
- 特殊的，如果这条出边在 AAA-tree 中是实孩子或父亲，则这个指针为空。
- 对于实 splay 中的每个节点，有五个孩子指针 *left*, *right*, *virtual*, *virtLeft*, *virtRight*。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。
- *real* 指向了其所维护以其为根重链的重 splay。
- 特殊的，如果这条出边在 AAA-tree 中是实孩子或父亲，则这个指针为空。
- 对于实 splay 中的每个节点，有五个孩子指针 *left*, *right*, *virtual*, *virtLeft*, *virtRight*。
- *left*, *right* 分别指向了其在实 splay 中的左右孩子。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。
- *real* 指向了其所维护以其为根重链的重 splay。
- 特殊的，如果这条出边在 AAA-tree 中是实孩子或父亲，则这个指针为空。
- 对于实 splay 中的每个节点，有五个孩子指针 *left*, *right*, *virtual*, *virtLeft*, *virtRight*。
- *left*, *right* 分别指向了其在实 splay 中的左右孩子。
- *virtual* 指向了维护其出边的虚 splay。

动态树练习题 IV：题解

- 简单的描述下 AAA-tree 的结构：
- 对于虚 splay 中的每个节点，拥有三个孩子指针 *left*, *right*, *real*。
- *left*, *right* 分别指向了其在虚 splay 中的左右孩子。
- *real* 指向了其所维护以其为根重链的重 splay。
- 特殊的，如果这条出边在 AAA-tree 中是实孩子或父亲，则这个指针为空。
- 对于实 splay 中的每个节点，有五个孩子指针 *left*, *right*, *virtual*, *virtLeft*, *virtRight*。
- *left*, *right* 分别指向了其在实 splay 中的左右孩子。
- *virtual* 指向了维护其出边的虚 splay。
- *virtLeft*, *virtRight* 分别指向了它的父亲与孩子在虚 splay 中对应的出边。这两个指针只起标记作用，不参与信息的处理。

动态树练习题 IV：题解

动态树练习题 IV：题解

- 基于链剖分的动态树一个重要的操作是翻转整条链（reverse）:

动态树练习题 IV：题解

- 基于链剖分的动态树一个重要的操作是翻转整条链（reverse）：
- 不同于 LCT，在下传实 splay 的翻转标记时，不仅需要交换 *left* 与 *right* 指针，还需要交换 *virtLeft* 与 *virtRight* 指针。

动态树练习题 IV：题解

- 基于链剖分的动态树一个重要的操作是翻转整条链（reverse）：
- 不同于 LCT，在下传实 splay 的翻转标记时，不仅需要交换 *left* 与 *right* 指针，还需要交换 *virtLeft* 与 *virtRight* 指针。
- 另一个重要的操作是修改孩子的虚实（splice）：

动态树练习题 IV：题解

- 基于链剖分的动态树一个重要的操作是翻转整条链（reverse）：
- 不同于 LCT，在下传实 splay 的翻转标记时，不仅需要交换 *left* 与 *right* 指针，还需要交换 *virtLeft* 与 *virtRight* 指针。
- 另一个重要的操作是修改孩子的虚实（splice）：
- 第一步：将它的实孩子变为虚孩子，先把 *virtRight* 指向的出边旋转至虚 splay 顶，接着把实孩子移交给它（用后者的 *real* 指针指向前者），最后将 *virtRight* 清空。

动态树练习题 IV：题解

- 基于链剖分的动态树一个重要的操作是翻转整条链（reverse）：
- 不同于 LCT，在下传实 splay 的翻转标记时，不仅需要交换 *left* 与 *right* 指针，还需要交换 *virtLeft* 与 *virtRight* 指针。
- 另一个重要的操作是修改孩子的虚实（splice）：
- 第一步：将它的实孩子变为虚孩子，先把 *virtRight* 指向的出边旋转至虚 splay 顶，接着把实孩子移交给它（用后者的 *real* 指针指向前者），最后将 *virtRight* 清空。
- 第二步：将某个虚孩子变为实孩子，先把该虚孩子所对应的出边旋转至链虚 splay 顶，并将其维护的实 splay 移出（即 *real* 指针所指向的），最后将 *virtRight* 指向该出边。

动态树练习题 IV：题解

- 基于链剖分的动态树一个重要的操作是翻转整条链（reverse）：
- 不同于 LCT，在下传实 splay 的翻转标记时，不仅需要交换 *left* 与 *right* 指针，还需要交换 *virtLeft* 与 *virtRight* 指针。
- 另一个重要的操作是修改孩子的虚实（splice）：
- 第一步：将它的实孩子变为虚孩子，先把 *virtRight* 指向的出边旋转至虚 splay 顶，接着把实孩子移交给它（用后者的 *real* 指针指向前者），最后将 *virtRight* 清空。
- 第二步：将某个虚孩子变为实孩子，先把该虚孩子所对应的出边旋转至链虚 splay 顶，并将其维护的实 splay 移出（即 *real* 指针所指向的），最后将 *virtRight* 指向该出边。
- 于是在 LCT 上进行的操作就在 AAA-tree 上复现了～

动态树练习题 IV：题解

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：
- 依然使用所有节点的孩子大小的对数和作为势能函数。

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：
- 依然使用所有节点的孩子大小的对数和作为势能函数。
- 那么在 LCT 上的证明依然可以套用到此处。

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：
- 依然使用所有节点的孩子大小的对数和作为势能函数。
- 那么在 LCT 上的证明依然可以套用到此处。
- ~~由于没有数位板，~~这里可能没有一个具体的分析……只谈一下大体的思路：

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：
- 依然使用所有节点的孩子大小的对数和作为势能函数。
- 那么在 LCT 上的证明依然可以套用到此处。
- ~~由于没有数位板，~~这里可能没有一个具体的分析……只谈一下大体的思路：
- 没有 reverse 的情况下，splice 中的第一步是不需要旋转的，而第二步的旋转可以直接套用 LCT 上的分析，于是只要说明 *splice* 对边修改引起的势能变化是小于 $3(\log|x'| - \log|x|)$ 的就行了。

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：
- 依然使用所有节点的孩子大小的对数和作为势能函数。
- 那么在 LCT 上的证明依然可以套用到此处。
- ~~由于没有数位板，~~这里可能没有一个具体的分析……只谈一下大体的思路：
- 没有 reverse 的情况下，splice 中的第一步是不需要旋转的，而第二步的旋转可以直接套用 LCT 上的分析，于是只要说明 *splice* 对边修改引起的势能变化是小于 $3(\log|x'| - \log|x|)$ 的就行了。
- 当存在 reverse 的时候，令 vr 为 $virtRight$，vt 为 $virt$，在势能函数里额外添加 $\log(vt) - \log(vr) + dis(vt, vr)$ 这么一项，即，将 vr 单旋上来的时间复杂度。这个东西只会在旋转 vl 使得 vr 进入它的子树时才会改变，而一次 *splay* 中这样的情况最多只会发生 1 次，单独分析以下就行了。

动态树练习题 IV：题解

- 考虑一个简单的复杂度证明：
- 依然使用所有节点的孩子大小的对数和作为势能函数。
- 那么在 LCT 上的证明依然可以套用到此处。
- ~~由于没有数位板，~~这里可能没有一个具体的分析……只谈一下大体的思路：
- 没有 reverse 的情况下，splice 中的第一步是不需要旋转的，而第二步的旋转可以直接套用 LCT 上的分析，于是只要说明 *splice* 对边修改引起的势能变化是小于 $3(\log|x'| - \log|x|)$ 的就行了。
- 当存在 reverse 的时候，令 vr 为 $virtRight$，vt 为 $virt$，在势能函数里额外添加 $\log(vt) - \log(vr) + dis(vt, vr)$ 这么一项，即，将 vr 单旋上来的时间复杂度。这个东西只会在旋转 vl 使得 vr 进入它的子树时才会改变，而一次 *splay* 中这样的情况最多只会发生 1 次，单独分析以下就行了。
- 时间复杂度：$O(N \log N)$。

动态树练习题 V

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上第 k 大。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上第 k 大。
- 4. 子树第 k 大。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上第 k 大。
- 4. 子树第 k 大。
- 5. 删除一条边后加上一条边，保证操作后依然为树。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上第 k 大。
- 4. 子树第 k 大。
- 5. 删除一条边后加上一条边，保证操作后依然为树。
- 要求强制在线。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上第 k 大。
- 4. 子树第 k 大。
- 5. 删除一条边后加上一条边，保证操作后依然为树。
- 要求强制在线。
- ……只要比 $O(N^2)$ 优秀就行了。

动态树练习题 V

- 有一棵 N 个节点的树，每个节点上有点权 v_i。
- 支持五种操作：
- 1. 链加。
- 2. 子树加。
- 3. 链上第 k 大。
- 4. 子树第 k 大。
- 5. 删除一条边后加上一条边，保证操作后依然为树。
- 要求强制在线。
- ……只要比 $O(N^2)$ 优秀就行了。
- 经典问题

动态树练习题 V：题解

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。
- 这里的 dfs 序是类似于树剖的 dfs 序，即，将 AAA-tree 中每个节点的实孩子放在最前面的 dfs 序。

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。
- 这里的 dfs 序是类似于树剖的 dfs 序，即，将 AAA-tree 中每个节点的实孩子放在最前面的 dfs 序。
- 这样，子树与链定位就会被定位到 dfs 序的一个区间上了。

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。
- 这里的 dfs 序是类似于树剖的 dfs 序，即，将 AAA-tree 中每个节点的实孩子放在最前面的 dfs 序。
- 这样，子树与链定位就会被定位到 dfs 序的一个区间上了。
- 考虑将 AAA-tree 上的修改映射到 dfs 序上：

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。
- 这里的 dfs 序是类似于树剖的 dfs 序，即，将 AAA-tree 中每个节点的实孩子放在最前面的 dfs 序。
- 这样，子树与链定位就会被定位到 dfs 序的一个区间上了。
- 考虑将 AAA-tree 上的修改映射到 dfs 序上：
- 切换边的虚实是一个区间移动操作。

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。
- 这里的 dfs 序是类似于树剖的 dfs 序，即，将 AAA-tree 中每个节点的实孩子放在最前面的 dfs 序。
- 这样，子树与链定位就会被定位到 dfs 序的一个区间上了。
- 考虑将 AAA-tree 上的修改映射到 dfs 序上：
- 切换边的虚实是一个区间移动操作。
- 修改一个节点的父亲也是一个区间移动操作。

动态树练习题 V：题解

- 显然，AAA-tree 上并不能支持询问第 k 大。考虑维护树的 dfs 序。
- 这里的 dfs 序是类似于树剖的 dfs 序，即，将 AAA-tree 中每个节点的实孩子放在最前面的 dfs 序。
- 这样，子树与链定位就会被定位到 dfs 序的一个区间上了。
- 考虑将 AAA-tree 上的修改映射到 dfs 序上：
- 切换边的虚实是一个区间移动操作。
- 修改一个节点的父亲也是一个区间移动操作。
- 下传翻转标记还是一个区间移动操作。

动态树练习题 V：题解

动态树练习题 V：题解

- 于是每次操作都可以转化 dfs 序上的 $O(\log N)$ 次区间移动操作。

动态树练习题 V：题解

- 于是每次操作都可以转化 dfs 序上的 $O(\log N)$ 次区间移动操作。
- 直接用块状链表维护即可。

动态树练习题 V：题解

- 于是每次操作都可以转化 dfs 序上的 $O(\log N)$ 次区间移动操作。
- 直接用块状链表维护即可。
- 时间复杂度：$O(N^{1.5}\log^{1.5} N)$。

动态树练习题 V：题解

- 于是每次操作都可以转化 dfs 序上的 $O(\log N)$ 次区间移动操作。
- 直接用块状链表维护即可。
- 时间复杂度：$O(N^{1.5}\log^{1.5} N)$。
- 常数看上去就很大，不确定能不能跑过暴力……

Maximum Tree Path

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。
- 令 $dist(u, v)$ 为链 u, v 的长度。

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。
- 令 $dist(u, v)$ 为链 u, v 的长度。
- 令 $gcd(u, v)$ 为链 u, v 上点权的最大公约数。

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。
- 令 $dist(u, v)$ 为链 u, v 的长度。
- 令 $gcd(u, v)$ 为链 u, v 上点权的最大公约数。
- 令 $min(u, v)$ 为链 u, v 之间点权的最小值。

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。
- 令 $dist(u, v)$ 为链 u, v 的长度。
- 令 $gcd(u, v)$ 为链 u, v 上点权的最大公约数。
- 令 $min(u, v)$ 为链 u, v 之间点权的最小值。
- 求在所有点对 (u, v) 中 $dist(u, v) \times gcd(u, v) \times min(u, v)$ 的最大值。

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。
- 令 $dist(u, v)$ 为链 u, v 的长度。
- 令 $gcd(u, v)$ 为链 u, v 上点权的最大公约数。
- 令 $min(u, v)$ 为链 u, v 之间点权的最小值。
- 求在所有点对 (u, v) 中 $dist(u, v) \times gcd(u, v) \times min(u, v)$ 的最大值。
- $N \leq 10^5$；$0 < v_i \leq 10^5$。

Maximum Tree Path

- 一棵 N 个节点的树，每个节点有点权 v_i。
- 令 $dist(u, v)$ 为链 u, v 的长度。
- 令 $gcd(u, v)$ 为链 u, v 上点权的最大公约数。
- 令 $min(u, v)$ 为链 u, v 之间点权的最小值。
- 求在所有点对 (u, v) 中 $dist(u, v) \times gcd(u, v) \times min(u, v)$ 的最大值。
- $N \leq 10^5$；$0 < v_i \leq 10^5$。
- Codechef March Cook-Off 2018

Maximum Tree Path：题解

Maximum Tree Path：题解

- ???

大 sz 的游戏

大 sz 的游戏

- 有 N 个点，另有一个参数 L。

大 sz 的游戏

- 有 N 个点，另有一个参数 L。
- 每个点有一对参数 $[l_i, r_i]$。

大 sz 的游戏

- 有 N 个点，另有一个参数 L。
- 每个点有一对参数 $[l_i, r_i]$。
- 如果点对 (i,j) 满足 $0 \le j - i \le L$，且 $[l_i, r_i]$ 与 $[l_j, r_j]$ 有交集，则从 j 向 i 连一条有向边。

大 sz 的游戏

- 有 N 个点，另有一个参数 L。
- 每个点有一对参数 $[l_i, r_i]$。
- 如果点对 (i, j) 满足 $0 \le j - i \le L$，且 $[l_i, r_i]$ 与 $[l_j, r_j]$ 有交集，则从 j 向 i 连一条有向边。
- 问，每个点到 1 的最短路。

大 sz 的游戏

- 有 N 个点，另有一个参数 L。
- 每个点有一对参数 $[l_i, r_i]$。
- 如果点对 (i,j) 满足 $0 \le j - i \le L$，且 $[l_i, r_i]$ 与 $[l_j, r_j]$ 有交集，则从 j 向 i 连一条有向边。
- 问，每个点到 1 的最短路。
- $N \le 5 \times 10^5$。

大 sz 的游戏

- 有 N 个点，另有一个参数 L。
- 每个点有一对参数 $[l_i, r_i]$。
- 如果点对 (i, j) 满足 $0 \le j - i \le L$，且 $[l_i, r_i]$ 与 $[l_j, r_j]$ 有交集，则从 j 向 i 连一条有向边。
- 问，每个点到 1 的最短路。
- $N \le 5 \times 10^5$。
- BZOJ

大 sz 的游戏：题解

大 sz 的游戏：题解

- 对参数建线段树，维护区间内的最小值。

大 sz 的游戏：题解

- 对参数建线段树，维护区间内的最小值。
- 由于标记之间的顺序是没有影响的，因此可以对标记永久化。

大 sz 的游戏：题解

- 对参数建线段树，维护区间内的最小值。
- 由于标记之间的顺序是没有影响的，因此可以对标记永久化。
- 由于这里的距离有单调性，可以用单调队列维护所有的标记。

大 sz 的游戏：题解

- 对参数建线段树，维护区间内的最小值。
- 由于标记之间的顺序是没有影响的，因此可以对标记永久化。
- 由于这里的距离有单调性，可以用单调队列维护所有的标记。
- 时间复杂度：$O(N \log N)$。

Bridges

Bridges

- 维护动态边双:

Bridges

- 维护动态边双:
- 给定一张 N 个点的简单连通无向图。

Bridges

- 维护动态边双:
- 给定一张 N 个点的简单连通无向图。
- 多次操作：支持加边或删边，保证任意时刻图都是简单无向图。

Bridges

- 维护动态边双:
- 给定一张 N 个点的简单连通无向图。
- 多次操作：支持加边或删边，保证任意时刻图都是简单无向图。
- 每次操作后询问图中桥边的个数。

Bridges

- 维护动态边双：
- 给定一张 N 个点的简单连通无向图。
- 多次操作：支持加边或删边，保证任意时刻图都是简单无向图。
- 每次操作后询问图中桥边的个数。
- $N, Q \leq 2 \times 10^6$。

Bridges

- 维护动态边双:
- 给定一张 N 个点的简单连通无向图。
- 多次操作：支持加边或删边，保证任意时刻图都是简单无向图。
- 每次操作后询问图中桥边的个数。
- $N, Q \leq 2 \times 10^6$。
- ICPC 2019 Winter Camp

Bridges：一个高复杂度的大常数做法

Bridges：一个高复杂度的大常数做法

- 按时间分治去掉删除操作。

Bridges：一个高复杂度的大常数做法

- 按时间分治去掉删除操作。
- 随便建出基图的一棵生成树。

Bridges：一个高复杂度的大常数做法

- 按时间分治去掉删除操作。
- 随便建出基图的一棵生成树。
- 新连一条非树边会使连接这两个点的链上的点都变成非桥边。

Bridges：一个高复杂度的大常数做法

- 按时间分治去掉删除操作。
- 随便建出基图的一棵生成树。
- 新连一条非树边会使连接这两个点的链上的点都变成非桥边。
- 加入非树边时链加 1，撤销时减 1，答案就是 0 的个数。

Bridges：一个高复杂度的大常数做法

- 按时间分治去掉删除操作。
- 随便建出基图的一棵生成树。
- 新连一条非树边会使连接这两个点的链上的点都变成非桥边。
- 加入非树边时链加 1，撤销时减 1，答案就是 0 的个数。
- 时间复杂度：$O(N\log^2 N)$。

Bridges：一个低复杂度的大常数做法

Bridges：一个低复杂度的大常数做法

- 依然按照时间分治。

Bridges：一个低复杂度的大常数做法

- 依然按照时间分治。
- 注意到在按时间分治的过程中，许多信息都是没有用的：

Bridges：一个低复杂度的大常数做法

- 依然按照时间分治。
- 注意到在按时间分治的过程中，许多信息都是没有用的：
- 比方说对于一个在递归过程中没有修改的二度点，显然可以将它的两条出边缩起来。

Bridges：一个低复杂度的大常数做法

- 依然按照时间分治。
- 注意到在按时间分治的过程中，许多信息都是没有用的：
- 比方说对于一个在递归过程中没有修改的二度点，显然可以将它的两条出边缩起来。
- 具体来说只有位于操作节点形成的虚树上的节点才是需要记录的，其他的都可以缩起来。

Bridges：一个低复杂度的大常数做法

- 依然按照时间分治。
- 注意到在按时间分治的过程中，许多信息都是没有用的：
- 比方说对于一个在递归过程中没有修改的二度点，显然可以将它的两条出边缩起来。
- 具体来说只有位于操作节点形成的虚树上的节点才是需要记录的，其他的都可以缩起来。
- 实现的时候暴力建出关键点的虚树并将其下传就行了。

Bridges：一个低复杂度的大常数做法

- 依然按照时间分治。
- 注意到在按时间分治的过程中，许多信息都是没有用的：
- 比方说对于一个在递归过程中没有修改的二度点，显然可以将它的两条出边缩起来。
- 具体来说只有位于操作节点形成的虚树上的节点才是需要记录的，其他的都可以缩起来。
- 实现的时候暴力建出关键点的虚树并将其下传就行了。
- 时间复杂度：$O(N \log N)$。

Problem C

Problem C

- 给定一棵 N 个点的树，第 i 条边的长度为 d_i。

Problem C

- 给定一棵 N 个点的树，第 i 条边的长度为 d_i。
- 支持两种操作：

Problem C

- 给定一棵 N 个点的树，第 i 条边的长度为 d_i。
- 支持两种操作：
- 修改某条边的边权。

Problem C

- 给定一棵 N 个点的树，第 i 条边的长度为 d_i。
- 支持两种操作：
- 修改某条边的边权。
- 问某个子树内的直径。

Problem C

- 给定一棵 N 个点的树，第 i 条边的长度为 d_i。
- 支持两种操作：
- 修改某条边的边权。
- 问某个子树内的直径。
- $N, Q \leq 10^5$；$d_i \geq 0$。

Problem C

- 给定一棵 N 个点的树，第 i 条边的长度为 d_i。
- 支持两种操作：
- 修改某条边的边权。
- 问某个子树内的直径。
- $N, Q \le 10^5$；$d_i \ge 0$。
- 经典问题？

Problem C：题解

Problem C：题解

- 对于没有修改的问题：

Problem C：题解

- 对于没有修改的问题：
- 由于边权是都是正的，因此可以直接维护 DFS 序区间的直径。

Problem C：题解

- 对于没有修改的问题：
- 由于边权是都是正的，因此可以直接维护 DFS 序区间的直径。
- 考虑修改的时候哪些区间内的答案会被影响到。

Problem C：题解

- 对于没有修改的问题：
- 由于边权是都是正的，因此可以直接维护 DFS 序区间的直径。
- 考虑修改的时候哪些区间内的答案会被影响到。
- 当修改的边的子树的 DFS 区间为 A 时，对于线段树上某个区间 B。

Problem C：题解

- 对于没有修改的问题：
- 由于边权是都是正的，因此可以直接维护 DFS 序区间的直径。
- 考虑修改的时候哪些区间内的答案会被影响到。
- 当修改的边的子树的 DFS 区间为 A 时，对于线段树上某个区间 B。
- 显然，当 A 包含 B，或 A 与 B 分离的时候，B 所对应的直径是不会被修改的。

Problem C：题解

- 对于没有修改的问题：
- 由于边权是都是正的，因此可以直接维护 DFS 序区间的直径。
- 考虑修改的时候哪些区间内的答案会被影响到。
- 当修改的边的子树的 DFS 区间为 A 时，对于线段树上某个区间 B。
- 显然，当 A 包含 B，或 A 与 B 分离的时候，B 所对应的直径是不会被修改的。
- 于是只有 $O(\log N)$ 个区间会被修改，它们是定位区间 A 时访问的区间。

Problem C：题解

- 对于没有修改的问题：
- 由于边权是都是正的，因此可以直接维护 DFS 序区间的直径。
- 考虑修改的时候哪些区间内的答案会被影响到。
- 当修改的边的子树的 DFS 区间为 A 时，对于线段树上某个区间 B。
- 显然，当 A 包含 B，或 A 与 B 分离的时候，B 所对应的直径是不会被修改的。
- 于是只有 $O(\log N)$ 个区间会被修改，它们是定位区间 A 时访问的区间。
- 时间复杂度：$O(N\log^2 N)$。

Count on a Treap

Count on a Treap

- 维护一棵大根 Treap。

Count on a Treap

- 维护一棵大根 Treap。
- 支持以下操作：

Count on a Treap

- 维护一棵大根 Treap。
- 支持以下操作：
- 1. 插入一个键值为 k，权值为 w 的新节点。

Count on a Treap

- 维护一棵大根 Treap。
- 支持以下操作：
- 1. 插入一个键值为 k，权值为 w 的新节点。
- 2. 删除键值为 k 的节点。

Count on a Treap

- 维护一棵大根 Treap。
- 支持以下操作：
- 1. 插入一个键值为 k，权值为 w 的新节点。
- 2. 删除键值为 k 的节点。
- 3. 询问键值为 a 与 b 之间的路径长度。

Count on a Treap

- 维护一棵大根 Treap。
- 支持以下操作：
- 1. 插入一个键值为 k，权值为 w 的新节点。
- 2. 删除键值为 k 的节点。
- 3. 询问键值为 a 与 b 之间的路径长度。
- $Q \le 2 \times 10^5$；保证权值、键值两两不同。

Count on a Treap

- 维护一棵大根 Treap。
- 支持以下操作：
- 1. 插入一个键值为 k，权值为 w 的新节点。
- 2. 删除键值为 k 的节点。
- 3. 询问键值为 a 与 b 之间的路径长度。
- $Q \leq 2 \times 10^5$；保证权值、键值两两不同。
- CodeChef February Challenge 2014

Count on a Treap：题解

Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。

Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。
- 因此离线后按键值排序，两个点的 LCA 就是它们之间权值最大的点。

Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。
- 因此离线后按键值排序，两个点的 LCA 就是它们之间权值最大的点。
- 于是只要求出每个点到根的距离就行了。

Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。
- 因此离线后按键值排序，两个点的 LCA 就是它们之间权值最大的点。
- 于是只要求出每个点到根的距离就行了。
- 在向左向右权值上升序列上的点都会成为该点的祖先。

Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。
- 因此离线后按键值排序，两个点的 LCA 就是它们之间权值最大的点。
- 于是只要求出每个点到根的距离就行了。
- 在向左向右权值上升序列上的点都会成为该点的祖先。
- 于是只需要维护向左向右的上升序列的长度即可。

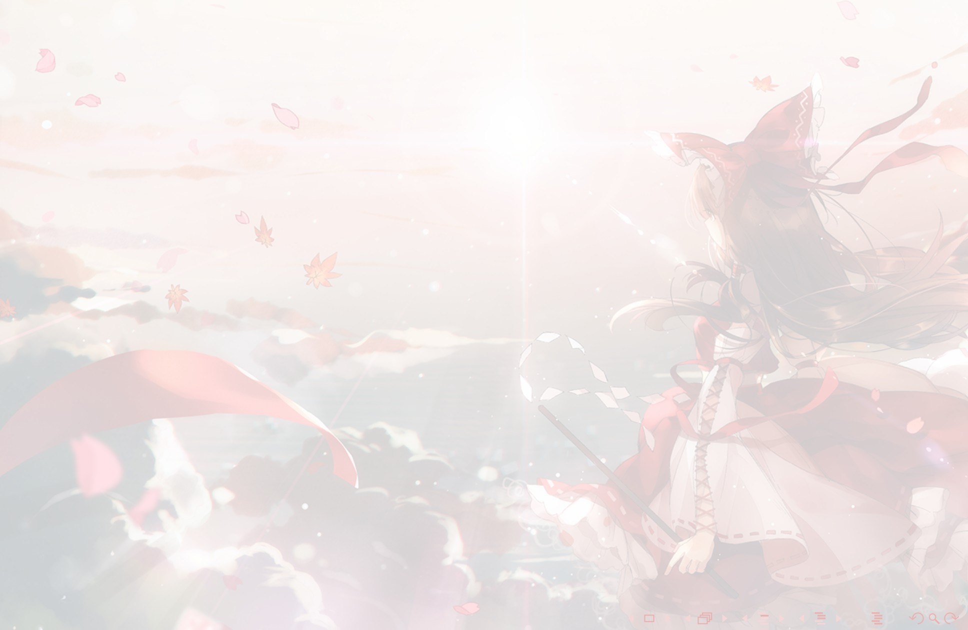
Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。
- 因此离线后按键值排序，两个点的 LCA 就是它们之间权值最大的点。
- 于是只要求出每个点到根的距离就行了。
- 在向左向右权值上升序列上的点都会成为该点的祖先。
- 于是只需要维护向左向右的上升序列的长度即可。
- 这个可以维护区间内的上升序列，合并的时候二分一发即可。

Count on a Treap：题解

- 考虑如何是如何建 Treap 的，先将所有键值从小至大排序，找到最大权的作为根，递归左右两侧建子树。
- 因此离线后按键值排序，两个点的 LCA 就是它们之间权值最大的点。
- 于是只要求出每个点到根的距离就行了。
- 在向左向右权值上升序列上的点都会成为该点的祖先。
- 于是只需要维护向左向右的上升序列的长度即可。
- 这个可以维护区间内的上升序列，合并的时候二分一发即可。
- 时间复杂度：$O(N\log^2 N)$。

美好的每一天 不连续的存在



美好的每一天 不连续的存在

- 维护一棵 N 个结点的无根树，每个节点有一个颜色。

美好的每一天 不连续的存在

- 维护一棵 N 个结点的无根树，每个节点有一个颜色。
- 多次询问：若仅保留编号在 $[l, r]$ 之内的点，编号为 k 的点所在的联通块内存在几种颜色。

美好的每一天 不连续的存在

- 维护一棵 N 个结点的无根树，每个节点有一个颜色。
- 多次询问：若仅保留编号在 $[l, r]$ 之内的点，编号为 k 的点所在的联通块内存在几种颜色。
- $N, Q \leq 10^5$ 。

美好的每一天 不连续的存在

- 维护一棵 N 个结点的无根树，每个节点有一个颜色。
- 多次询问：若仅保留编号在 $[l, r]$ 之内的点，编号为 k 的点所在的联通块内存在几种颜色。
- $N, Q \leq 10^5$ 。
- YNOI 2019

美好的每一天 不连续的存在：题解

美好的每一天 不连续的存在 ： 题解

- 直接维护看上去不太能维护，考虑点分。

美好的每一天 不连续的存在 ：题解

- 直接维护看上去不太能维护，考虑点分。
- 当询问某点的答案时，在与其联通的点分树上最远祖先处统计答案。

美好的每一天 不连续的存在 ： 题解

- 直接维护看上去不太能维护，考虑点分。
- 当询问某点的答案时，在与其联通的点分树上最远祖先处统计答案。
- 这样就把问题转化为了有根树上的问题。

美好的每一天 不连续的存在 ：题解

- 直接维护看上去不太能维护，考虑点分。
- 当询问某点的答案时，在与其联通的点分树上最远祖先处统计答案。
- 这样就把问题转化为了有根树上的问题。
- 通过询问每个点到根之间的最大与最小值，可以知道怎样的询问区间才会包含这个点。

美好的每一天 不连续的存在：题解

- 直接维护看上去不太能维护，考虑点分。
- 当询问某点的答案时，在与其联通的点分树上最远祖先处统计答案。
- 这样就把问题转化为了有根树上的问题。
- 通过询问每个点到根之间的最大与最小值，可以知道怎样的询问区间才会包含这个点。
- 于是问题就转化为了，平面上有若干个点，询问一某个点右下角的点的颜色种类数。

美好的每一天 不连续的存在 ：题解

- 直接维护看上去不太能维护，考虑点分。
- 当询问某点的答案时，在与其联通的点分树上最远祖先处统计答案。
- 这样就把问题转化为了有根树上的问题。
- 通过询问每个点到根之间的最大与最小值，可以知道怎样的询问区间才会包含这个点。
- 于是问题就转化为了，平面上有若干个点，询问一某个点右下角的点的颜色种类数。
- 对每种颜色分别考虑，对于其中的每个点，给其划分一个矩形让其单独管辖，这样就不会统计重复了。

美好的每一天 不连续的存在：题解

- 直接维护看上去不太能维护，考虑点分。
- 当询问某点的答案时，在与其联通的点分树上最远祖先处统计答案。
- 这样就把问题转化为了有根树上的问题。
- 通过询问每个点到根之间的最大与最小值，可以知道怎样的询问区间才会包含这个点。
- 于是问题就转化为了，平面上有若干个点，询问一某个点右下角的点的颜色种类数。
- 对每种颜色分别考虑，对于其中的每个点，给其划分一个矩形让其单独管辖，这样就不会统计重复了。
- 时间复杂度：$O(N\log^2 N)$。

“旅行”

- 小 y^{∞} 是个 dark 魔王。

“旅行”

“旅行”

- 小 y^∞ 有一个 $W \times H$ 的矩形棋盘。最初的时候，棋盘是空的。

“旅行”

- 小 y^∞ 有一个 $W \times H$ 的矩形棋盘。最初的时候，棋盘是空的。
- 心情不好的时候，小 y^∞ 会在 (l_i, y_i) 至 (r_i, y_i) 之间的每个格子里摆放一个妹子。第 i 次操作放到棋盘上的妹子都属于组 i。保证 y_i 这一行当前没有其他妹子。

“旅行”

- 小 y^∞ 有一个 $W \times H$ 的矩形棋盘。最初的时候，棋盘是空的。
- 心情不好的时候，小 y^∞ 会在 (l_i, y_i) 至 (r_i, y_i) 之间的每个格子里摆放一个妹子。第 i 次操作放到棋盘上的妹子都属于组 i。保证 y_i 这一行当前没有其他妹子。
- 心情好的时候，小 y^∞ 会选择某个组 i，将组内的所有妹子移出棋盘。

“旅行”

- 小 y^∞ 有一个 $W \times H$ 的矩形棋盘。最初的时候，棋盘是空的。
- 心情不好的时候，小 y^∞ 会在 (l_i, y_i) 至 (r_i, y_i) 之间的每个格子里摆放一个妹子。第 i 次操作放到棋盘上的妹子都属于组 i。保证 y_i 这一行当前没有其他妹子。
- 心情好的时候，小 y^∞ 会选择某个组 i，将组内的所有妹子移出棋盘。
- 有些时候，小 y^∞ 会在棋盘上散步，从 (x_i, l_i) 出发，一格一格走到 (x_i, r_i)。每次散步开始的时候，小 y^∞ 的愉悦度为 0。每当他在某个格子遇到了组 j 的妹子，他的愉悦度会从 v 变成 $p_j v + q_j$。输出愉悦度对 323232323 取模的结果。

“旅行”

- 小 y^∞ 有一个 $W \times H$ 的矩形棋盘。最初的时候，棋盘是空的。
- 心情不好的时候，小 y^∞ 会在 (l_i, y_i) 至 (r_i, y_i) 之间的每个格子里摆放一个妹子。第 i 次操作放到棋盘上的妹子都属于组 i。保证 y_i 这一行当前没有其他妹子。
- 心情好的时候，小 y^∞ 会选择某个组 i，将组内的所有妹子移出棋盘。
- 有些时候，小 y^∞ 会在棋盘上散步，从 (x_i, l_i) 出发，一格一格走到 (x_i, r_i)。每次散步开始的时候，小 y^∞ 的愉悦度为 0。每当他在某个格子遇到了组 j 的妹子，他的愉悦度会从 v 变成 $p_j v + q_j$。输出愉悦度对 323232323 取模的结果。
- $Q \leq 10^5$。

“旅行”

- 小 y^∞ 有一个 $W \times H$ 的矩形棋盘。最初的时候，棋盘是空的。
- 心情不好的时候，小 y^∞ 会在 (l_i, y_i) 至 (r_i, y_i) 之间的每个格子里摆放一个妹子。第 i 次操作放到棋盘上的妹子都属于组 i。保证 y_i 这一行当前没有其他妹子。
- 心情好的时候，小 y^∞ 会选择某个组 i，将组内的所有妹子移出棋盘。
- 有些时候，小 y^∞ 会在棋盘上散步，从 (x_i, l_i) 出发，一格一格走到 (x_i, r_i)。每次散步开始的时候，小 y^∞ 的愉悦度为 0。每当他在某个格子遇到了组 j 的妹子，他的愉悦度会从 v 变成 $p_j v + q_j$。输出愉悦度对 323232323 取模的结果。
- $Q \le 10^5$。
- 计蒜之道 2018

“旅行”：题解

“旅行”：题解

- 直接维护看上去不太能维护。

“旅行”：题解

- 直接维护看上去不太能维护。
- 注意到这个问题一共有 3 条轴——x, y, T。

“旅行”：题解

- 直接维护看上去不太能维护。
- 注意到这个问题一共有 3 条轴——x, y, T。
- 不妨转移坐标轴，将 y 视为时间进行维护。

“旅行”：题解

- 直接维护看上去不太能维护。
- 注意到这个问题一共有 3 条轴——x, y, T。
- 不妨转移坐标轴，将 y 视为时间进行维护。
- 那么每个修改都会定位到一个矩形。

“旅行”：题解

- 直接维护看上去不太能维护。
- 注意到这个问题一共有 3 条轴——x, y, T。
- 不妨转移坐标轴，将 y 视为时间进行维护。
- 那么每个修改都会定位到一个矩形。
- 查询就是问在一段时间中覆盖到某个点的函数值之和。

“旅行”：题解

- 直接维护看上去不太能维护。
- 注意到这个问题一共有 3 条轴——x, y, T。
- 不妨转移坐标轴，将 y 视为时间进行维护。
- 那么每个修改都会定位到一个矩形。
- 查询就是问在一段时间中覆盖到某个点的函数值之和。
- 直接在 K-D 树上打标记即可。

“旅行”：题解

- 直接维护看上去不太能维护。
- 注意到这个问题一共有 3 条轴——x, y, T。
- 不妨转移坐标轴，将 y 视为时间进行维护。
- 那么每个修改都会定位到一个矩形。
- 查询就是问在一段时间中覆盖到某个点的函数值之和。
- 直接在 K-D 树上打标记即可。
- 时间复杂度：$O(N\sqrt{N})$。

镜中的昆虫

镜中的昆虫

- 维护一个长度为 N 的数组。

镜中的昆虫

- 维护一个长度为 N 的数组。
- 支持两种操作：

镜中的昆虫

- 维护一个长度为 N 的数组。
- 支持两种操作：
- 1. 区间修改为 x。

镜中的昆虫

- 维护一个长度为 N 的数组。
- 支持两种操作:
- 1. 区间修改为 x。
- 2. 询问区间 $[l,r]$ 中出现了多少种不同的数。

镜中的昆虫

- 维护一个长度为 N 的数组。
- 支持两种操作:
- 1. 区间修改为 x。
- 2. 询问区间 $[l, r]$ 中出现了多少种不同的数。
- $N, Q \leq 10^5$。

镜中的昆虫

- 维护一个长度为 N 的数组。
- 支持两种操作：
- 1. 区间修改为 x。
- 2. 询问区间 $[l, r]$ 中出现了多少种不同的数。
- $N, Q \leq 10^5$。
- YNOI 2016

镜中的昆虫：题解

镜中的昆虫：题解

- 询问数的种类，可以维护每个数上一次出现的位置 p_i。

镜中的昆虫：题解

- 询问数的种类，可以维护每个数上一次出现的位置 p_i。
- 那么 $[l, r]$ 的答案就是其中满足 $p_i \le l$ 的个数

镜中的昆虫：题解

- 询问数的种类，可以维护每个数上一次出现的位置 p_i。
- 那么 $[l, r]$ 的答案就是其中满足 $p_i \le l$ 的个数
- 区间赋值期望修改的 p_i 数量是均摊 $O(1)$ 的。

镜中的昆虫：题解

- 询问数的种类，可以维护每个数上一次出现的位置 p_i。
- 那么 $[l, r]$ 的答案就是其中满足 $p_i \le l$ 的个数
- 区间赋值期望修改的 p_i 数量是均摊 $O(1)$ 的。
- 于是问题就变成了单点修改，区间询问有多少个数小于 x。

镜中的昆虫：题解

- 询问数的种类，可以维护每个数上一次出现的位置 p_i。
- 那么 $[l, r]$ 的答案就是其中满足 $p_i \le l$ 的个数
- 区间赋值期望修改的 p_i 数量是均摊 $O(1)$ 的。
- 于是问题就变成了单点修改，区间询问有多少个数小于 x。
- CDQ 分治一发或者直接上树套树。

镜中的昆虫：题解

- 询问数的种类，可以维护每个数上一次出现的位置 p_i。
- 那么 $[l, r]$ 的答案就是其中满足 $p_i \le l$ 的个数
- 区间赋值期望修改的 p_i 数量是均摊 $O(1)$ 的。
- 于是问题就变成了单点修改，区间询问有多少个数小于 x。
- CDQ 分治一发或者直接上树套树。
- 时间复杂度 $O(N \log^2 N)$。

Travel Through Time

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \le x$，将其染黑。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \leq x$，将其染黑。
- 3. 翻转区间 $[l, r]$ ，即，对于每个满足 $x \in [l, r]$ 的整点 x ，将它的颜色赋为整点 $r+l-x$ 的颜色。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作:
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \leq x$，将其染黑。
- 3. 翻转区间 $[l,r]$ ，即，对于每个满足 $x \in [l,r]$ 的整点 x ，将它的颜色赋为整点 $r+l-x$ 的颜色。
- 4. 询问某个整点 x 的颜色。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \leq x$，将其染黑。
- 3. 翻转区间 $[l, r]$ ，即，对于每个满足 $x \in [l, r]$ 的整点 x ，将它的颜色赋为整点 $r+l-x$ 的颜色。
- 4. 询问某个整点 x 的颜色。
- 要求强制在线，完全可持久化。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \le x$，将其染黑。
- 3. 翻转区间 $[l, r]$，即，对于每个满足 $x \in [l, r]$ 的整点 x，将它的颜色赋为整点 $r+l-x$ 的颜色。
- 4. 询问某个整点 x 的颜色。
- 要求强制在线，完全可持久化。
- $Q \le 10^5$；所有修改不会超出 long long 范围。

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \le x$，将其染黑。
- 3. 翻转区间 $[l,r]$ ，即，对于每个满足 $x \in [l,r]$ 的整点 x ，将它的颜色赋为整点 $r+l-x$ 的颜色。
- 4. 询问某个整点 x 的颜色。
- 要求强制在线，完全可持久化。
- $Q \le 10^5$；所有修改不会超出 long long 范围。
- 2018 Multi-University Training Contest 4

Travel Through Time

- 有一条数轴，每个整点可能是黑的，也可能是白的。最初，所有的整点都是白的。
- 支持四种操作：
- 1. 将某个整点 x 染黑。
- 2. 对于每个整点 z，如果存在一个黑点 y 满足 $|y-z| \le x$，将其染黑。
- 3. 翻转区间 $[l,r]$ ，即，对于每个满足 $x \in [l,r]$ 的整点 x ，将它的颜色赋为整点 $r+l-x$ 的颜色。
- 4. 询问某个整点 x 的颜色。
- 要求强制在线，完全可持久化。
- $Q \le 10^5$；所有修改不会超出 long long 范围。
- 2018 Multi-University Training Contest 4
- 题意稍有修改。（避免了某些边界上的讨论）

Travel Through Time：题解

Travel Through Time：题解

- 似乎有好几种做法？

Travel Through Time：题解

- 似乎有好几种做法?
- 这里描述一个基于维护括号序列的做法:

Travel Through Time：题解

- 似乎有好几种做法?
- 这里描述一个基于维护括号序列的做法:
- 考虑维护一些全是黑点的区间，这些区间可以有重叠部分，但是需要保证这些区间覆盖了所有的黑点。

Travel Through Time：题解

- 似乎有好几种做法?
- 这里描述一个基于维护括号序列的做法:
- 考虑维护一些全是黑点的区间，这些区间可以有重叠部分，但是需要保证这些区间覆盖了所有的黑点。
- 注意到如果只维护了区间的左右边界，甚至不用知道某个左边界具体对应的是哪个右边界，就可以知道某个位置的颜色。

Travel Through Time：题解

- 似乎有好几种做法?
- 这里描述一个基于维护括号序列的做法:
- 考虑维护一些全是黑点的区间，这些区间可以有重叠部分，但是需要保证这些区间覆盖了所有的黑点。
- 注意到如果只维护了区间的左右边界，甚至不用知道某个左边界具体对应的是哪个右边界，就可以知道某个位置的颜色。
- 因此只维护左右边界，把左边界看成左括号，右边界看成右括号，那么这就是一个括号序列。

Travel Through Time：题解

Travel Through Time：题解

- 依次考虑下每个操作在括号序列里可以怎么表达：

Travel Through Time：题解

- 依次考虑下每个操作在括号序列里可以怎么表达：
- 操作一：新建一对括号。

Travel Through Time：题解

- 依次考虑下每个操作在括号序列里可以怎么表达：
- 操作一：新建一对括号。
- 操作二：将所有左括号向左移动 x，将所有右括号向右移动 x。

Travel Through Time：题解

- 依次考虑下每个操作在括号序列里可以怎么表达：
- 操作一：新建一对括号。
- 操作二：将所有左括号向左移动 x，将所有右括号向右移动 x。
- 操作三：先在 l 的前面与 r 的后面加入形如...)))(((... 的括号对使得没有线段穿过 l 和 r。再翻转 $[l, r]$ 内的所有括号。

Travel Through Time：题解

- 依次考虑下每个操作在括号序列里可以怎么表达：
- 操作一：新建一对括号。
- 操作二：将所有左括号向左移动 x，将所有右括号向右移动 x。
- 操作三：先在 l 的前面与 r 的后面加入形如...)))(((... 的括号对使得没有线段穿过 l 和 r。再翻转 $[l, r]$ 内的所有括号。
- 对左括号与右括号分别使用一棵平衡树来维护。如果一个位置上有许多括号，维护此处括号数量即可。

Travel Through Time：题解

- 依次考虑下每个操作在括号序列里可以怎么表达：
- 操作一：新建一对括号。
- 操作二：将所有左括号向左移动 x，将所有右括号向右移动 x。
- 操作三：先在 l 的前面与 r 的后面加入形如...)))(((... 的括号对使得没有线段穿过 l 和 r。再翻转 $[l, r]$ 内的所有括号。
- 对左括号与右括号分别使用一棵平衡树来维护。如果一个位置上有许多括号，维护此处括号数量即可。
- 时间复杂度：$O(N \log N)$。

祝大家省选顺利!