# Solution

ExfJoe

福建省长乐第一中学

March 19, 2017

# Outline

# 一丝给朴

# 一丝给朴

- 我们将树当做以 1 为根的有根树

# 一丝给朴

- 我们将树当做以 1 为根的有根树
- 考虑将树划分为若干个连通块，连通块中 $a_i > 0$ 的 $a_i$ 之和作为收益 $g$，$a_i < 0$ 的 $a_i$ 之和作为支出 $p$

# 一丝给朴

- 我们将树当做以 1 为根的有根树
- 考虑将树划分为若干个连通块，连通块中 $a_i > 0$ 的 $a_i$ 之和作为收益 $g$，$a_i < 0$ 的 $a_i$ 之和作为支出 $p$
- 显然只有 $g > p$ 的连通块我们才有必要访问它

# 一丝给朴

- 我们将树当做以 1 为根的有根树
- 考虑将树划分为若干个连通块，连通块中 $a_i > 0$ 的 $a_i$ 之和作为收益 $g$，$a_i < 0$ 的 $a_i$ 之和作为支出 $p$
- 显然只有 $g > p$ 的连通块我们才有必要访问它
- 访问时先去 $p$ 小的再去 $p$ 大的不会更劣，因此我们考虑将划分好的连通块按 $p$ 排序

# 一丝给朴

- 我们将树当做以 1 为根的有根树
- 考虑将树划分为若干个连通块，连通块中 $a_i > 0$ 的 $a_i$ 之和作为收益 $g$，$a_i < 0$ 的 $a_i$ 之和作为支出 $p$
- 显然只有 $g > p$ 的连通块我们才有必要访问它
- 访问时先去 $p$ 小的再去 $p$ 大的不会更劣，因此我们考虑将划分好的连通块按 $p$ 排序
- 由于访问还需要满足拓扑序 (不能不访问父亲直接进入儿子)，因此有些连通块我们需要进行合并

# 一丝给朴

- 我们将树当做以 1 为根的有根树
- 考虑将树划分为若干个连通块，连通块中 $a_i > 0$ 的 $a_i$ 之和作为收益 $g$，$a_i < 0$ 的 $a_i$ 之和作为支出 $p$
- 显然只有 $g > p$ 的连通块我们才有必要访问它
- 访问时先去 $p$ 小的再去 $p$ 大的不会更劣，因此我们考虑将划分好的连通块按 $p$ 排序
- 由于访问还需要满足拓扑序 (不能不访问父亲直接进入儿子)，因此有些连通块我们需要进行合并
- 我们考虑直接按树的遍历顺序进行合并，若当前结点为 $u$，则我们假设它的儿子的连通块序列 (按 $p$ 排序) 已经维护好了

# 一丝给朴

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$
- 若 $g_u < p_u$，则它一定要与 $v$ 合并，否则它没有访问的必要

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$
- 若 $g_u < p_u$，则它一定要与 $v$ 合并，否则它没有访问的必要
- 若 $g_u \geq p_u$，且 $p_u > p_v$，则若我们访问了 $u$，则接下来也一定能访问 $v$，所以我们可以将 $u, v$ 合并，并且访问时仍然满足拓扑序

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$
- 若 $g_u < p_u$，则它一定要与 $v$ 合并，否则它没有访问的必要
- 若 $g_u \geq p_u$，且 $p_u > p_v$，则若我们访问了 $u$，则接下来也一定能访问 $v$，所以我们可以将 $u, v$ 合并，并且访问时仍然满足拓扑序
- 若合并完后仍有 $g_u < p_u$，则当前连通块直接舍去

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$
- 若 $g_u < p_u$，则它一定要与 $v$ 合并，否则它没有访问的必要
- 若 $g_u \geq p_u$，且 $p_u > p_v$，则若我们访问了 $u$，则接下来也一定能访问 $v$，所以我们可以将 $u, v$ 合并，并且访问时仍然满足拓扑序
- 若合并完后仍有 $g_u < p_u$，则当前连通块直接舍去
- 否则容易看出，连通块仍然按 $p$ 有序，且满足拓扑序

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$
- 若 $g_u < p_u$，则它一定要与 $v$ 合并，否则它没有访问的必要
- 若 $g_u \geq p_u$，且 $p_u > p_v$，则若我们访问了 $u$，则接下来也一定能访问 $v$，所以我们可以将 $u, v$ 合并，并且访问时仍然满足拓扑序
- 若合并完后仍有 $g_u < p_u$，则当前连通块直接舍去
- 否则容易看出，连通块仍然按 $p$ 有序，且满足拓扑序
- 因此我们需要一个数据结构支持查询、删除最小值以及合并

# 一丝给朴

- 令当前还未合并的连通块中，$p$ 的最小的块为 $v$
- 令 $u$ 当前所在块的 $g, p$ 分别为 $g_u, p_u$
- 若 $g_u < p_u$，则它一定要与 $v$ 合并，否则它没有访问的必要
- 若 $g_u \geq p_u$，且 $p_u > p_v$，则若我们访问了 $u$，则接下来也一定能访问 $v$，所以我们可以将 $u, v$ 合并，并且访问时仍然满足拓扑序
- 若合并完后仍有 $g_u < p_u$，则当前连通块直接舍去
- 否则容易看出，连通块仍然按 $p$ 有序，且满足拓扑序
- 因此我们需要一个数据结构支持查询、删除最小值以及合并
- 随便用一个可并堆即可，时间复杂度 $O(n \log n)$

# Outline

# 欧珀瑞特

# 欧珀瑞特

- 询问类型 1：经典问题，由于异或运算每位独立，因此从高位开始贪心

# 欧珀瑞特

- 询问类型 1：经典问题，由于异或运算每位独立，因此从高位开始贪心
- 将数字建成二进制 Trie 树，并用 Trie 来实现贪心过程

# 欧珀瑞特

- 询问类型 1：经典问题，由于异或运算每位独立，因此从高位开始贪心
- 将数字建成二进制 Trie 树，并用 Trie 来实现贪心过程
- 询问类型 3,4：经典问题，可以将所有数字建成一棵权值线段树，并在结点上维护子树内数字个数来完成询问

# 欧珀瑞特

- 询问类型 1：经典问题，由于异或运算每位独立，因此从高位开始贪心
- 将数字建成二进制 Trie 树，并用 Trie 来实现贪心过程
- 询问类型 3,4：经典问题，可以将所有数字建成一棵权值线段树，并在结点上维护子树内数字个数来完成询问
- 由于插入、删除操作都只在末尾进行，所以区间询问可以用主席树简单的处理

# Outline

# 思君

# 思君

- 把所有串拼在一起形成大串 $S$，相邻两个串间用一个特殊字符隔开

# 思君

- 把所有串拼在一起形成大串 $S$，相邻两个串间用一个特殊字符隔开
- 建出 $S$ 的后缀自动机，现在对于每个状态我们需要统计它所代表的字符串在多少个原串中出现了

# 思君

- 把所有串拼在一起形成大串 $S$，相邻两个串间用一个特殊字符隔开
- 建出 $S$ 的后缀自动机，现在对于每个状态我们需要统计它所代表的字符串在多少个原串中出现了
- 每个结点用一棵线段树维护它所代表的字符串在哪些原串中出现，然后在 parent 树上线段树合并即可

# 思君

- 把所有串拼在一起形成大串 $S$，相邻两个串间用一个特殊字符隔开
- 建出 $S$ 的后缀自动机，现在对于每个状态我们需要统计它所代表的字符串在多少个原串中出现了
- 每个结点用一棵线段树维护它所代表的字符串在哪些原串中出现，然后在 parent 树上线段树合并即可
- 计算答案时枚举子串的右端点 $r$，用 two points 找到最小的左端点 $l$ 满足 $S[l \dots r]$ 在至少 $k$ 个原串中出现了，这样 $l' > l$ 的 $l'$ 也是满足条件的

# 思君

- 把所有串拼在一起形成大串 $S$，相邻两个串间用一个特殊字符隔开
- 建出 $S$ 的后缀自动机，现在对于每个状态我们需要统计它所代表的字符串在多少个原串中出现了
- 每个结点用一棵线段树维护它所代表的字符串在哪些原串中出现，然后在 parent 树上线段树合并即可
- 计算答案时枚举子串的右端点 $r$，用 two points 找到最小的左端点 $l$ 满足 $S[l \dots r]$ 在至少 $k$ 个原串中出现了，这样 $l' > l$ 的 $l'$ 也是满足条件的
- 找 $l$ 的过程还是利用 parent 树，找到祖先中长度范围包含 $r - l + 1$ 的结点 $u$，则 $u$ 所维护的线段树就是我们所需要的信息

# 思君

- 把所有串拼在一起形成大串 $S$，相邻两个串间用一个特殊字符隔开
- 建出 $S$ 的后缀自动机，现在对于每个状态我们需要统计它所代表的字符串在多少个原串中出现了
- 每个结点用一棵线段树维护它所代表的字符串在哪些原串中出现，然后在 parent 树上线段树合并即可
- 计算答案时枚举子串的右端点 $r$，用 two points 找到最小的左端点 $l$ 满足 $S[l \dots r]$ 在至少 $k$ 个原串中出现了，这样 $l' > l$ 的 $l'$ 也是满足条件的
- 找 $l$ 的过程还是利用 parent 树，找到祖先中长度范围包含 $r - l + 1$ 的结点 $u$，则 $u$ 所维护的线段树就是我们所需要的信息
- 时间复杂度 $O(n \log n)$