2024年广东省大学生程序设计竞赛(GDCPC)暨CCPC广州邀请赛

2024 年 5 月 26 日

清华大学学生算法协会

感谢 Mys.C.K., liuzhangfeiabc, Itst, SpiritualKhorosho, EliminateSpace, xiaolilsq, gyh-20, JohnVictor36, Renshey, QAQAutoMaton 负责命题和验题工作.

I 不等式 by *JohnVictor36*

G Menji 和 gcd by *gyh-20*

C DFS 序 by *EliminateSpace*

E 循环赛 by *ckw20*

B 腊肠披萨 by *SpiritualKhorosho*

J 另一个计数问题 by *Renshey*

F 图 by *xiaolilsq*

H 小班课 by *gyh-20*

A 田字格 by *ltst*

D 拨打电话 by *SpiritualKhorosho*

I 不等式 by *JohnVictor36*

题目大意

给定 n, m，以及 m 个形如 $a_{x_i} \geq a_{y_i} + a_{z_i} (1 \leq i \leq m)$ 的条件。问是否有一组正整数 $(a_1, a_2, \cdots, a_n)$ 满足所有条件，并且 $a_1 + a_2 + \cdots + a_n \leq 10^9$。如果有，输出 $a_1 + a_2 + \cdots + a_n$ 的最小值；如果无解，输出 -1。

题解

考虑连边 $x_i \to y_i, x_i \to z_i$. 注意到如果这个图里面有环，那么环上的最小值一定不满足对应的要求；如果图里面无环，找出拓扑序，按照拓扑序依次贪心确定每个数的最小可能值即可. 如果超过了 10^9 直接退出就不会溢出. 时间复杂度 $O(n+m)$.

G Menji 和 gcd by *gyh-20*

题目大意

求 $\max\limits_{L \le x < y \le R} \gcd(x, y)$。

题解

分答案 $\geq \sqrt{r}$ 和答案 $\leq \sqrt{r}$ 的情况。

题解

分答案 $\geq \sqrt{r}$ 和答案 $\leq \sqrt{r}$ 的情况。

答案可以为 x 当且仅当 $\left\lfloor \frac{r}{x} \right\rfloor - \left\lfloor \frac{l-1}{x} \right\rfloor \geq 2$，可以 $O(1)$ 检查。

题解

分答案 $\geq \sqrt{r}$ 和答案 $\leq \sqrt{r}$ 的情况。

答案可以为 x 当且仅当 $\left\lfloor \frac{r}{x} \right\rfloor - \left\lfloor \frac{l-1}{x} \right\rfloor \geq 2$，可以 $O(1)$ 检查。

于是 $\leq \sqrt{r}$ 的情况可以枚举。

题解

分答案 $\geq \sqrt{r}$ 和答案 $\leq \sqrt{r}$ 的情况。

答案可以为 x 当且仅当 $\left\lfloor \frac{r}{x} \right\rfloor - \left\lfloor \frac{l-1}{x} \right\rfloor \geq 2$，可以 $O(1)$ 检查。

于是 $\leq \sqrt{r}$ 的情况可以枚举。

否则可以枚举 $\left\lfloor \frac{r}{x} \right\rfloor$ 求出最小的合法的 x。

题解

分答案 $\geq \sqrt{r}$ 和答案 $\leq \sqrt{r}$ 的情况。

答案可以为 x 当且仅当 $\left\lfloor \frac{r}{x} \right\rfloor - \left\lfloor \frac{l-1}{x} \right\rfloor \geq 2$，可以 $O(1)$ 检查。

于是 $\leq \sqrt{r}$ 的情况可以枚举。

否则可以枚举 $\left\lfloor \frac{r}{x} \right\rfloor$ 求出最小的合法的 x。

时间复杂度 $O(\sqrt{r})$。

C DFS 序 by *EliminateSpace*

题目大意

给定一棵 n 个点的有根树，1 号点为根。每个点有一个权值 w_i。

求一个最优的 DFS 序使得 $\sum_{i=1}^{n} p_i w_i$ 最大。

题解

决策是考虑进入每个点之后选子树的顺序。

题解

决策是考虑进入每个点之后选子树的顺序。

子树内部显然用内部的最优方案。

题解

决策是考虑进入每个点之后选子树的顺序。

子树内部显然用内部的最优方案。

然后子树之间的贡献是先走的子树的大小乘后走的子树的权值和。

题解

决策是考虑进入每个点之后选子树的顺序。

子树内部显然用内部的最优方案。

然后子树之间的贡献是先走的子树的大小乘后走的子树的权值和。

所以按照子树大小除以权值和从大到小排序访问就是最优的。可以用调整法证明贪心是对的。

题解

决策是考虑进入每个点之后选子树的顺序。

子树内部显然用内部的最优方案。

然后子树之间的贡献是先走的子树的大小乘后走的子树的权值和。

所以按照子树大小除以权值和从大到小排序访问就是最优的。可以用调整法证明贪心是对的。

时间复杂度 $O(n \log n)$。

E 循环赛 by *ckw20*

题目大意

称一张 n 个点的竞赛图为 (n,m)-图，如果对其中任意 m 个点，其中同时存在全胜和全败者。

求在所有 (n,m)-图中点度集合大小最少是多少。

题解

打表找规律题。

记答案为 $F(n,m)$。结论：

- $F(n,2) = 2 - (n \bmod 2)$
- $\forall m \in \left[3, \left\lfloor \frac{n}{2} \right\rfloor + 2\right], F(n,m) = n$
- otherwise, $F(n,m) = n - 2 \times \left(m - \left\lfloor \frac{n}{2} \right\rfloor - 2\right)$

（不是）证明

对 $m=2$ 相当于没有限制。

（不是）证明

对 $m=2$ 相当于没有限制。

由于所有点度和为 $\frac{n(n-1)}{2}$，平均点度是 $\frac{n-1}{2}$，所以当 n 是偶数的时候 $F(n,m)\geq 2$。

（不是）证明

对 $m=2$ 相当于没有限制。

由于所有点度和为 $\frac{n(n-1)}{2}$，平均点度是 $\frac{n-1}{2}$，所以当 n 是偶数的时候 $F(n,m)\geq 2$。

容易根据 n 的奇偶性构造使 $F(2k+1,2)=1, F(2k,2)=2$。

（不是）证明

考察一些 corner case。

（不是）证明

考察一些 corner case。

对 $m=3$ 相当于任意三点都有全序关系，很容易立刻发现此时不存在环，因此答案是 n。

（不是）证明

考察一些 corner case。

对 $m=3$ 相当于任意三点都有全序关系，很容易立刻发现此时不存在环，因此答案是 n。

根据样例可以猜测当 m 在某个范围内的时候答案保持为 n，因此可以猜测考虑递推。

（不是）证明

考察一些 corner case。

对 $m=3$ 相当于任意三点都有全序关系，很容易立刻发现此时不存在环，因此答案是 n。

根据样例可以猜测当 m 在某个范围内的时候答案保持为 n，因此可以猜测考虑递推。

实际上可以证明（分了五六种情况讨论太复杂了就不写出来了），当 $4 \le m < n$ 时，有 $F(n,m)=F(n-2,m-1)+2$，方法大概是尝试证明度数最小/最大的点唯一且条件可以归纳。

（不是）证明

但是上述过程的条件会在 $n = m$ 时无法继续归纳，需要特判。

（不是）证明

但是上述过程的条件会在 $n = m$ 时无法继续归纳，需要特判。

此时由于存在唯一全胜和全败者，所以答案至少为 3；同 $m = 2$，依据 n 的奇偶性可能至少为 4。容易构造对应方案。

（不是）证明

但是上述过程的条件会在 $n=m$ 时无法继续归纳，需要特判。

此时由于存在唯一全胜和全败者，所以答案至少为 3；同 $m=2$，依据 n 的奇偶性可能至少为 4。容易构造对应方案。

实际上删去最大度和最小度点后相当于没有额外限制的 $m=2$，所以 $F(n,n)=F(n-2,2)+2$。

（不是）证明

根据上述递推可以得到开始的结果。

B 腊肠披萨 by *SpiritualKhorosho*

简要题意

定义 $LCPS(s,t)$ 为最长的 r 使得 $s_1 \cdots s_{|r|} = t_{|t|-|r|+1} \cdots t_{|t|} = r$. 求

$$\sum_{i=1}^{L} \sum_{j=1}^{L} C^{|LCPS(s_i,s_j)|} \bmod P.$$

$1 \le L, \sum_{i=1}^{L} |s_i| \le 3 \times 10^6, 2 \le C < P < 2^{30}$.

观察

虽然 $LCPS$ 看上去没有什么规律, 但是注意到输入串的所有前缀和所有后缀的数量都是与总串长相等的 (如果去重则更少) , 可以考虑用后缀去查是否存在对应的前缀, 并更新相应 (s_i, s_j) 的答案. 对前缀建立 Hash 表, 即可 $\Theta(1)$ 查询每个后缀.

观察

虽然 $LCPS$ 看上去没有什么规律, 但是注意到输入串的所有前缀和所有后缀的数量都是与总串长相等的 (如果去重则更少) , 可以考虑用后缀去查是否存在对应的前缀, 并更新相应 (s_i, s_j) 的答案. 对前缀建立 Hash 表, 即可 $\Theta(1)$ 查询每个后缀.

这个做法有一个问题: 一个前缀可能对应多个原串, 如果需要统计每对 (s_i, s_j) 的 $LCPS$ 长度再计算答案, 复杂度可能会爆炸.

观察

虽然 $LCPS$ 看上去没有什么规律, 但是注意到输入串的所有前缀和所有后缀的数量都是与总串长相等的 (如果去重则更少) , 可以考虑用后缀去查是否存在对应的前缀, 并更新相应 (s_i, s_j) 的答案. 对前缀建立 Hash 表, 即可 $\Theta(1)$ 查询每个后缀.

这个做法有一个问题: 一个前缀可能对应多个原串, 如果需要统计每对 (s_i, s_j) 的 $LCPS$ 长度再计算答案, 复杂度可能会爆炸.

因此, 我们需要设计一种方法, 通过查询到的前缀直接计算该前缀的贡献, 并且想办法容斥保证每对 (s_i, s_j) 只有匹配上的最长前缀/后缀的原始贡献计入答案.

容斥

考虑什么时候一个串 s_i 会有多个前缀被另一个串 s_j 的相应后缀匹配上. 假设已知 $s_{i,1}\cdots s_{i,k}=s_{j,|s_j|-k+1}\cdots s_{j,|s_j|}$ 且 $s_{i,1}\cdots s_{i,l}=s_{j,|s_j|-l+1}\cdots s_{j,|s_j|}$, 其中 $l<k$, 那么不难发现:

容斥

考虑什么时候一个串 s_i 会有多个前缀被另一个串 s_j 的相应后缀匹配上. 假设已知 $s_{i,1}\cdots s_{i,k} = s_{j,|s_j|-k+1}\cdots s_{j,|s_j|}$ 且 $s_{i,1}\cdots s_{i,l} = s_{j,|s_j|-l+1}\cdots s_{j,|s_j|}$, 其中 $l < k$, 那么不难发现:

$s_{i,1}\cdots s_{i,l}$ 应该是 $s_{i,1}\cdots s_{i,k}$ 的一个 border!

容斥

考虑什么时候一个串 s_i 会有多个前缀被另一个串 s_j 的相应后缀匹配上. 假设已知 $s_{i,1}\cdots s_{i,k} = s_{j,|s_j|-k+1}\cdots s_{j,|s_j|}$ 且 $s_{i,1}\cdots s_{i,l} = s_{j,|s_j|-l+1}\cdots s_{j,|s_j|}$, 其中 $l < k$, 那么不难发现:

$s_{i,1}\cdots s_{i,l}$ 应该是 $s_{i,1}\cdots s_{i,k}$ 的一个 border!

这启发我们可以对每个串跑 KMP, 用 fail 来处理容斥. 注意到 border 的 border 仍是 border, 这个容斥的形式非常好看.

容斥

考虑什么时候一个串 s_i 会有多个前缀被另一个串 s_j 的相应后缀匹配上. 假设已知 $s_{i,1}\cdots s_{i,k}=s_{j,|s_j|-k+1}\cdots s_{j,|s_j|}$ 且 $s_{i,1}\cdots s_{i,l}=s_{j,|s_j|-l+1}\cdots s_{j,|s_j|}$, 其中 $l<k$, 那么不难发现:

$s_{i,1}\cdots s_{i,l}$ 应该是 $s_{i,1}\cdots s_{i,k}$ 的一个 border!

这启发我们可以对每个串跑 KMP, 用 fail 来处理容斥. 注意到 border 的 border 仍是 border, 这个容斥的形式非常好看.

一个前缀会在每个 border 处统计一次该 border 的贡献. 因此, 我们只需要将当前前缀的原始贡献减去其最长 border 的原始贡献. 把一个前缀的原始贡献拆分成其 border 贡献之和后, 我们就可以在查询后缀时直接把对应贡献乘上前缀出现次数计入答案即可.

容斥

考虑什么时候一个串 s_i 会有多个前缀被另一个串 s_j 的相应后缀匹配上. 假设已知 $s_{i,1}\cdots s_{i,k}=s_{j,|s_j|-k+1}\cdots s_{j,|s_j|}$ 且 $s_{i,1}\cdots s_{i,l}=s_{j,|s_j|-l+1}\cdots s_{j,|s_j|}$, 其中 $l<k$, 那么不难发现:

$s_{i,1}\cdots s_{i,l}$ 应该是 $s_{i,1}\cdots s_{i,k}$ 的一个 border!

这启发我们可以对每个串跑 KMP, 用 fail 来处理容斥. 注意到 border 的 border 仍是 border, 这个容斥的形式非常好看.

一个前缀会在每个 border 处统计一次该 border 的贡献. 因此, 我们只需要将当前前缀的原始贡献减去其最长 border 的原始贡献. 把一个前缀的原始贡献拆分成其 border 贡献之和后, 我们就可以在查询后缀时直接把对应贡献乘上前缀出现次数计入答案即可.

总复杂度 $O\left(\sum_{i=1}^{L}|s_i|\right)$. 注意 C 的幂次可以递推, 如果每次暴力算会多个快速幂的 log, 有可能无法通过本题.

J 另一个计数问题 by *Renshey*

题目大意

给定一个 $n-1$ 个点的无向图，点的编号为 $2\sim n$。对于所有的 $2\le u<v\le n$，边 (u,v) 存在当且仅当 v 是 u 的正整数倍。定义 $f(u,v)$ 表示 u 与 v 是否连通：当 u,v 连通时 $f(u,v)=1$，否则 $f(u,v)=0$。求：

$$\left(\sum_{u=2}^{n-1}\sum_{v=u+1}^{n} f(u,v)\cdot u\cdot v\right) \bmod 998244353$$

题解

注意到，对于所有 $x \le \lfloor \frac{n+1}{2} \rfloor$ 的结点 x，x 与 $2x$ 间有一条边，而 $2x$ 与 2 之间有一条边，因此所有 $2 \sim \lfloor \frac{n+1}{2} \rfloor$ 范围内的结点连通。而对于 $x > \lfloor \frac{n+1}{2} \rfloor$ 的结点 x，x 不与 $2 \sim \lfloor \frac{n+1}{2} \rfloor$ 中的结点连通当且仅当 x 为素数。因此整个图仅包含若干个单独的素数结点与一个大的连通块。

题解

注意到，对于所有 $x \le \lfloor \frac{n+1}{2} \rfloor$ 的结点 x，x 与 $2x$ 间有一条边，而 $2x$ 与 2 之间有一条边，因此所有 $2 \sim \lfloor \frac{n+1}{2} \rfloor$ 范围内的结点连通。而对于 $x > \lfloor \frac{n+1}{2} \rfloor$ 的结点 x，x 不与 $2 \sim \lfloor \frac{n+1}{2} \rfloor$ 中的结点连通当且仅当 x 为素数。因此整个图仅包含若干个单独的素数结点与一个大的连通块。

考虑计算不连通的点对之间的贡献。不难发现，只需要求出所有大于 $\lfloor \frac{n+1}{2} \rfloor$ 的素数的和与平方和。使用 Min_25 筛或分块打表均可以。

F 图 by *xiaolilsq*

题目大意

给定一张 n 个点 m 条边的无向图，要求找到两个点 u, v 满足它们间的边不相交路径有至少 $\lceil m/(n-1) \rceil$ 条。

题解

看到 $n-1$ 联想到 n 个点的树恰好就是 $n-1$ 条边，或者说 n 个点的森林最多只有 $n-1$ 条边。这启发我们去维护若干棵森林。

题解

看到 $n-1$ 联想到 n 个点的树恰好就是 $n-1$ 条边，或者说 n 个点的森林最多只有 $n-1$ 条边。这启发我们去维护若干棵森林。

具体而言使用 kruskal 的方法维护至少 $\lceil m/(n-1) \rceil$ 个 n 个点的森林，初始所有森林都是没有边的，然后一条边一条边地加入森林中，每次加入边都将其添加到尽可能靠前的一个森林中去。

题解

看到 $n-1$ 联想到 n 个点的树恰好就是 $n-1$ 条边，或者说 n 个点的森林最多只有 $n-1$ 条边。这启发我们去维护若干棵森林。

具体而言使用 kruskal 的方法维护至少 $\lceil m/(n-1) \rceil$ 个 n 个点的森林，初始所有森林都是没有边的，然后一条边一条边地加入森林中，每次加入边都将其添加到尽可能靠前的一个森林中去。

由于前 $\lceil m/(n-1) \rceil - 1$ 个森林最多只能容纳 $(\lceil m/(n-1) \rceil - 1)(n-1) < m$ 条边，所以最后一个森林中一定至少包含一条边，也就是说在最后一个森林中至少存在 $u \neq v$ 满足 u, v 中至少有一条路径。

题解

注意到由于每次添加边我们都是将其添加到尽可能靠前的森林中去的，所以如果 u, v 在第 i 个森林中连通，那么它必然在第 $i-1$ 个森林中连通，否则容易导出矛盾。由此我们直到对于最后一个森林中任意两个连通的点 u, v，它们在前面的森林中都连通，所以每个森林中取一条路径即可构成 $\lceil m/(n-1) \rceil$ 条边不相交路径了。

题解

注意到由于每次添加边我们都是将其添加到尽可能靠前的森林中去的，所以如果 u, v 在第 i 个森林中连通，那么它必然在第 $i-1$ 个森林中连通，否则容易导出矛盾。由此我们直到对于最后一个森林中任意两个连通的点 u, v，它们在前面的森林中都连通，所以每个森林中取一条路径即可构成 $\lceil m/(n-1) \rceil$ 条边不相交路径了。

额外需要注意的就是将边加入尽可能靠前的森林的时候不能直接枚举，这样时间复杂度至少是 $O(m^2/n)$ 的，当 n 比较小 m 比较大的时候会超时，而根据我们维护的森林结构性质，可以使用二分的手段来找到应该加入哪个森林中，这样时间复杂度不超过 $O(m \log m)$。

H 小班课 by *gyh-20*

题目大意

学生们按意向依次进行选课，每次会选择优先度最高且未满的小班课。

现在给出每个学生的意向度序列，请重排学生的顺序，使得选上小班课的学生最多。并构造方案。

结论

最终一定能取到最大匹配，可以通过递归完成构造。

Key Obeservation：总存在一个最大匹配，使得存在一个学生选到了其最喜欢的小班课。

证明

证明：令 $(p_1, q_1), (p_2, q_2) \ldots (p_k, q_k)$ 为一组最大匹配。 $r_1, r_2 \ldots r_k$ 分别为 $p_1, p_2 \ldots p_k$ 最喜欢的小班课。

证明

证明：令 $(p_1, q_1), (p_2, q_2) \ldots (p_k, q_k)$ 为一组最大匹配。 $r_1, r_2 \ldots r_k$ 分别为 $p_1, p_2 \ldots p_k$ 最喜欢的小班课。

若存在 r_i 不为任何一个 q_j，那么直接将 q_i调整为 r_i 即可。

证明

证明：令 $(p_1,q_1),(p_2,q_2)\dots(p_k,q_k)$ 为一组最大匹配。 $r_1,r_2\dots r_k$ 分别为 $p_1,p_2\dots p_k$ 最喜欢的小班课。

若存在 r_i 不为任何一个 q_j，那么直接将 q_i调整为 r_i 即可。

否则 $(p_i,q_i),(p_i,r_i)$ 均连边后构成的二分图，其恰好有 $2k$ 个结点和 $2k$ 条边。而左部每个点度数为 2，可以重复找到右部度数为 1 的点 q_i，删去 p_i 和 q_i，因此会被删掉 2 个点和 2 条边，可以递归下去。最终一定所有点度数都为 2，构成若干个环。对于每个环上所有点将 q_i 均调整为 r_i 即可。

证明

于是我们有如下做法：每次任意找到一个选到其最喜欢的小班课的学生，将其放入队列的第一个，将该学生删除，课容量减一，如果课容量为 0 删去该课程。然后递归构造。

证明

于是我们有如下做法：每次任意找到一个选到其最喜欢的小班课的学生，将其放入队列的第一个，将该学生删除，课容量减一，如果课容量为 0 删去该课程。然后递归构造。

实际实现中，首先需要一个二分图匹配，这里的复杂度是 $O(n^3)$。上述的证明可以直接实现成构造，单次构造是 $O(n)$ 的，复杂度为 $O(n^2)$。具体实现还可以用一个小技巧，即仅保留 $(p_i, q_i), (p_i, r_i)$ 这些边之后随机顺序做二分图匹配，这样每个环有 $\frac{1}{2}$ 的顺序改变顺序，每一次匹配是 $O(n^2)$ 的，期望复杂度同样也是 $O(n^2)$ 的。

A 田字格 by *ltst*

题目大意

给定平面上与坐标轴平行的 $n \le 3 \times 10^5$ 条黑色线段，求出它们构成多少田字格。一个田字格由三元组 (x_0, y_0, d) 表示，一个三元组是田字格当且仅当正方形 $[x_0 - d, x_0 + d] \times [y_0 - d, y_0 + d]$ 与平面上黑色部分的交恰好等于 $x = x_0 - d, x = x_0, x = x_0 + d, y = y_0 - d, y = y_0, y = y_0 + d$ 六条直线与这个正方形的交。

题解

注意到一个田字格由三条等距的、中间没有插入其他横线的横线组，和三条等距的、中间没有插入其他竖线的竖线组构成。考虑先把这样的横线组和竖线组处理出来，然后计算所有横线组和竖线组可以合并出多少的田字格。

题解

考虑第一步，维护出所有的竖线组，横线组类似。按照纵坐标扫描线，每次维护与 $y = y_0$ 有交的竖线集合，按照横坐标排序。那么可能的竖线组在这个序列上一定是连续排列的（因为中间不能有其他竖线）。总共会形成 $O(n)$ 次插入，每次插入只会改变 $O(1)$ 个竖线组的存在情况，使用 ‘set‘ 维护。这样我们可以得到 $O(n)$ 个四元组 (x_0, x_1, y_0, y_1)，表示从 $y = y_0$ 到 $y = y_1$，三条竖线 $x = x_0, x = \frac{x_0+x_1}{2}, x = x_1$ 存在且中间没有其他竖线。

题解

处理出所有的横线组和竖线组之后，枚举 d，此时竖线组的 $x_1 - x_0$ 和横线组的 $y_1 - y_0$ 是确定的。一个横线组 (x_0, y_0, y_1) 和一个竖线组 (x'_0, x'_1, y'_0) 是一个合法的田字格当且仅当

题解

处理出所有的横线组和竖线组之后，枚举 d，此时竖线组的 $x_1 - x_0$ 和横线组的 $y_1 - y_0$ 是确定的。一个横线组 (x_0, y_0, y_1) 和一个竖线组 (x'_0, x'_1, y'_0) 是一个合法的田字格当且仅当

- $x'_0 \le x_0 \le x'_1 - 2d$；

题解

处理出所有的横线组和竖线组之后，枚举 d，此时竖线组的 $x_1 - x_0$ 和横线组的 $y_1 - y_0$ 是确定的。一个横线组 (x_0, y_0, y_1) 和一个竖线组 (x'_0, x'_1, y'_0) 是一个合法的田字格当且仅当

- $x'_0 \le x_0 \le x'_1 - 2d$；
- $y_0 \le y'_0 \le y_1 - 2d$。

题解

处理出所有的横线组和竖线组之后，枚举 d，此时竖线组的 $x_1 - x_0$ 和横线组的 $y_1 - y_0$ 是确定的。一个横线组 (x_0, y_0, y_1) 和一个竖线组 (x'_0, x'_1, y'_0) 是一个合法的田字格当且仅当

- $x'_0 \le x_0 \le x'_1 - 2d$；
- $y_0 \le y'_0 \le y_1 - 2d$。

这是一个二维偏序，对一维扫描线另一维树状数组统计即可。复杂度 $O(n \log n)$

D 拨打电话 by *SpiritualKhorosho*

简要题意

定义一棵有根树上从叶结点 u 向叶结点 v 拨打电话的正确号码是沿着树上最短路径依次连接相应字符串的结果. 给定一棵有 n 个结点的有根树和 q 组询问, 每组询问为从一个结点拨打指定的电话号码, 可以匹配上多少结点.

$2 \le n \le 10^5, 1 \le q \le 10^5$, 树上串及询问串总长分别不超过 3×10^6.

观察

由于需要比较的号码可能很长, 不难想到用 Hash 来加速字符串比较的过程. 即, 如果我们能快速求出从一个叶结点出发, 到其它所有叶节点的路径对应的号码的 Hash 值, 我们就可以通过这个 Hash 值来判断有多少叶结点匹配询问的电话号码.

观察

由于需要比较的号码可能很长, 不难想到用 Hash 来加速字符串比较的过程. 即, 如果我们能快速求出从一个叶结点出发, 到其它所有叶节点的路径对应的号码的 Hash 值, 我们就可以通过这个 Hash 值来判断有多少叶结点匹配询问的电话号码.

直接维护本题给出的号码比较困难, 但我们仍然可以考虑使用点分治的方法来处理路径问题.

观察

由于需要比较的号码可能很长, 不难想到用 Hash 来加速字符串比较的过程. 即, 如果我们能快速求出从一个叶结点出发, 到其它所有叶节点的路径对应的号码的 Hash 值, 我们就可以通过这个 Hash 值来判断有多少叶结点匹配询问的电话号码.

直接维护本题给出的号码比较困难, 但我们仍然可以考虑使用点分治的方法来处理路径问题.

观察

把前缀 b_i, c_i, d_i 都看成是对 Hash 值的一个变换, 那么一条路径对应的号码相当于这些变换的复合. 如果我们使用一些可逆的变换 (如最常用的线性函数 $f(c, hash) = w(c) + a \cdot hash$), 我们就可以对询问串 (的 Hash 值) 逆变换得到:

观察

把前缀 b_i, c_i, d_i 都看成是对 Hash 值的一个变换, 那么一条路径对应的号码相当于这些变换的复合. 如果我们使用一些可逆的变换 (如最常用的线性函数 $f(c, hash) = w(c) + a \cdot hash$) , 我们就可以对询问串 (的 Hash 值) 逆变换得到:

如果一个电话经过当前重心, 则从重心往后, 仍需处理的电话号码 (的 Hash 值). 这样, 我们就可以在重心处比较去掉部分前缀的查询串和加上部分前缀的电话号码是否相等.

设计 Hash

如果只有 b_i 和 c_i, 则相应的 Hash 函数可以直接看成单向边的边权. 对于各叶结点的电话号码 a_i, 从 f_i 到 i, 需要使用 c_i 进行变换, 而从 i 到 f_i 则需使用 b_i 变换. 对于询问串, 则应分别用相应的逆进行变换.

设计 Hash

如果只有 b_i 和 c_i, 则相应的 Hash 函数可以直接看成单向边的边权. 对于各叶结点的电话号码 a_i, 从 f_i 到 i, 需要使用 c_i 进行变换, 而从 i 到 f_i 则需使用 b_i 变换. 对于询问串, 则应分别用相应的逆进行变换.

加上 d_i, 问题变得稍微更复杂一些, 但仍然可以在原来的基础上实现. 注意到只有从 f_i 到 i 时不需要使用 d_i, 我们对指向子结点的边的 c_j 变换左复合 d_i, 而对 f_i 到 i 的 c_i 右复合 d_i^{-1} 即可. 这样只有在子树内会计入 d_i 的贡献, 而从父结点进入子树时贡献可以被抵消掉.

设计 Hash

如果只有 b_i 和 c_i, 则相应的 Hash 函数可以直接看成单向边的边权. 对于各叶结点的电话号码 a_i, 从 f_i 到 i, 需要使用 c_i 进行变换, 而从 i 到 f_i 则需使用 b_i 变换. 对于询问串, 则应分别用相应的逆进行变换.

加上 d_i, 问题变得稍微更复杂一些, 但仍然可以在原来的基础上实现. 注意到只有从 f_i 到 i 时不需要使用 d_i, 我们对指向子结点的边的 c_j 变换左复合 d_i, 而对 f_i 到 i 的 c_i 右复合 d_i^{-1} 即可. 这样只有在子树内会计入 d_i 的贡献, 而从父结点进入子树时贡献可以被抵消掉.

总复杂度 $O\big((n+q)\log n+\sum|a_i|+\sum|b_i|+\sum|c_i|+\sum|d_i|+\sum|r_i|\big)$.

感谢倾听!