涨信心模拟赛 Solution

预期难度：T1 < T2 = T4 < T3。（T4 代码难度略大）

预期得分：$100 + [40, 100] + [30, 50] + [30, 100] = [200, 350]$。

算法考察：

T1：树形 dp，简单的 dp 空间优化。

T2：质因数分解， trie，时间复杂度优化。

T3：博弈论，dp，贪心。

T4：数学式推导，线段树变形。

T1 烟花

Problem provided by `lty`

题目大意

题目已经叙述得很清楚，这里不再赘述。

子任务设计

20% 的数据给暴力，包括但不限于搜索算法。

对于 70% 的数据，有一个图论算法：

将每个点分为“该居民在不同时间段工作”，共 $r_i - l_i - t_i + 2$ 个点。对于有仇视关系的居民，把这两个居民不同时间段的点之间分别连边，边权为重叠时间和 C_i 的乘积。其它边都连边权为 0。

由于原图是一棵树，按上述方法建出的的图一定是一个 DAG（有向无环图）。

我们在图上进行拓扑排序，并进行一个很简单的 dp 就可以计算出答案。

时空复杂度都是 $O(T^2N)$，其中 T 表示有效的时间段数。

这个空间是过不了的（512 MB也不行！），不过鉴于各位可能有~~优秀的卡常技巧~~优秀的做法，还是给了 512 MB。

注：经过出题人的实测，当且仅当开一个 500^3 的 `int` 数组时可以得到 60 分，其它任何更大的空间都会全部爆炸。

正解

70% 的算法时间复杂度已经可以通过 100% 的数据了，我们考虑优化空间。

事实上这个新图不用真的建出，我们可以在原图上进行一个树形 dp。

定义 $f_{u,t}$ 表示考虑了以 u 为根的子树，u 这个居民在 $[l_u + t, l_u + t_u + t - 1]$ 的时间段工作，产生的最小怒气值。

注意到我们通过状态定义的转化，避开了 $l_i, r_i \le 10^9$ 这个棘手的条件，将其转化为有效时间段数。

于是有转移方程如下：

$f_{u,t} = \min_{l_u \le t \le r_u - t_u + 1} \{ \sum_{u \to v} \min_{l_v \le tv \le r_v - t_v + 1} \{ f_{v,tv} + cost \} \}$

$cost = C_i \times \max(0, \min(l_u + t_u + t - 1, l_v + t_v + tv - 1) - \max(l_u + t, l_v + tv) + 1)$

枚举 u, t, tv，时间复杂度 $O(T^2 N)$，空间复杂度 $O(TN)$，其中 T 表示有效时间段数，可以通过本题。

T2 数

Problem provided by `lsl`

子任务设计

对于 20% 的数据，给暴力打表。

对于 40% 的数据，给不加 trie 的正确做法。

正解

引理1

若一个数 a 是完全立方数，则它满足：

$a = T_1^{c_1} T_2^{c_2} \dots T_m^{c_m},\ \forall i \in [1, m],\ c_i \bmod 3 = 0$

引理2

若想要构造最小的 x，使得 $a \times x$ 为完全立方数，则 x 满足：

设 $a = T_1^{c_1} T_2^{c_2} \dots T_m^{c_m}$。

令数列 $G = \{(3 - c_1) \bmod 3, (3 - c_2) \bmod 3, \dots, (3 - c_m) \bmod 3\}$。

则 $x = T_1^{G_1} T_2^{G_2} T_3^{G_3} \dots T_m^{G_m}$。

对于每一个数 S_i，我们考虑对它进行质因数分解（只对题目给定的 m 个质数），则有以下两种情况。

1. 分解后，剩下的数是 1，那么我们一定可以构造出一个 x 使得 $x \times Si$ 为完全立方数，该数有效；
2. 分解后，剩下的数不为 1；

对于情况2，我们对剩下的数 res 进行特判。

若 res 为完全立方数，那么我们可以构造出这样的 x，使得 $x \times S_i$ 为完全立方数，该数有效；

反之，不可构造出这样的 x 使得 $x \times Si$ 成为完全立方数，该数无效。

注意：

判断一个数 res 是否为为完全立方数有两个做法。

`C++` 中有个函数 $pow()$，所以我们可以先算出 res 的立方根 $now = pow(res, \frac{1}{3})$，当 now 是一个整数时，res 一定为完全立方数。

这看起来没有问题，但是这份代码输出为 `NO`（~~不信自己去试一试~~）。

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n=1e18;
long double x;
int main()
{
    x=pow(n,1/3.0);
    if(ceil(x)==floor(x)) cout<<"YES";
    else cout<<"NO";
}
```

因为**精度爆炸了！！！** ~~(如果有哪位奆佬用奇怪的方法把精度卡过去了当我什么也没说)~~

所以我们判断一个数 res 是否为完全立方数，二分是有效的。

```cpp
bool check(long long rest)
{
    l=1,r=1000000;
    while(l<=r)
    {
        long long mid=(l+r)>>1;
        if(mid*mid*mid==rest) return 1;
        mid*mid*mid<rest?l=mid+1:r=mid-1;
    }
    return 0;
}
```

对于一个确定的数 S_i 都有一个确定的分解方式 $S_i = T_1^{c_{i,1}} T_2^{c_{i,2}} \dots T_m^{c_{i,m}}$，所以对于一个确定的 S_i，都有与之对应的数列 $c_{i,1}, c_{i,2}, c_{i,3}, \dots, c_{i,m}$。

我们发现对于每一个 $c_{i,j}, j \in [1, m]$，它可以等价于 $c_{i,j} \bmod 3, j \in [1, m]$。因为不论是 $c_{i,j}$ 还是 $c_{i,j} \bmod 3$ 它们构造出的答案是一样的。

所以我们构造一个数列 $g_i = \{(3 - c_{i,1}) \bmod 3, (3 - c_{i,2}) \bmod 3, \dots, (3 - c_{i,m}) \bmod 3\}$。

不难发现 $\forall i, j \in [1, m], g_{i,j} \in [0, 2]$。

那么现在的问题就是我们要维护一个数据结构，它可以将一个只由 $0, 1, 2$ 组成的数列**映射**到一个整数。

很明显，这个可以使用 map 来维护，通常有两种做法：

1：把这个数列压成一个 string，然后把它用 map 来映射。

2：把这个数列放进一个 vector，然后把它用 map 来映射。

因为常数很大，所以这两种方法应该是不可行的（在 10^8 级别上再多个 log 好像不能过）。

（另外，**subtask 3** 卡了 unordered_map，但不知道成不成功）

那么我们介绍一种不用 map 的方法，使用 0，1，2 trie树。

我们使用 trie 时，通过数组存下标来记录一个字符串。

同理，我们可以通过同样的方式记录一个 $0, 1, 2$ 串。

接下来我们来看看对于每个操作，我们在 trie 上进行的操作。

op = 1

若新加入的数 new 是有效的，则我们把它的叶子节点的权值加上 1。

若 new 是无效的，则我们直接跳过。

时间复杂度为 $O(M)$。

op = 2

若删除的数 sub 是有效的，则我们把它的叶子节点的权值减去 1。

若 sub 是无效的，则我们直接跳过。

时间复杂度为 $O(M)$。

op = 3

我们访问每一个叶子节点，寻找对于最大的权值的所有叶子节点的答案的最小值。

最坏时间复杂度为 $O(Q)$，平均复杂度为 $O(\frac{Q}{2})$。

最坏时间复杂度为 $O(\frac{Q^2}{4})$。空间复杂度 $O(MQ)$。

T3 游戏

Problem provided by `lty`

子任务设置

对于 10% 的数据，送分。

对于 30% 的数据，可以 $O(N^2)$ 地枚举加哪条边。

对于 50% 的数据，稍后介绍一个 $O(N \log N)$ 的算法。

正解

策略分析

首先，我们定义一个结点先手必败是 `lose`，先手必胜是 `win`。

容易发现叶子结点一定是 `lose`。对于非叶子结点，只要它有一个儿子是 `lose`，它就是 `win`。否则该结点也是 `lose`。一遍 dfs 可以求出每一个点的初始状态，从而判断是否输出零。

如果输出非零，即根节点的状态与当前我们的先后手相反，说明我们本来是必败的。

于是可以发现对于这棵尚未加边的树，`win` 结点一定是对方走，且会走向一个 `lose` 结点，而 `lose` 结点时我们在走。

我们考虑怎么连边。显然，我们会从一个 `lose` 结点向外连边，因为 `win` 结点是对方在走，连了边是无意义的。

可以发现我们一定不会连一条返祖边。因为我们连了返祖边之后，如果走到一个 `win` 结点，就是对方在走，我们无法胜利；如果连到一个 `lose` 结点，那么就使对方变成了我们在那个点时的状态，这样对方走到连出边的这个结点时又可以通过返祖边走到祖先，游戏永远也不会结束。

因此我们可以放弃返祖边。这样，这棵树在加边后仍然是一张 DAG。显然，我们只需要连向一个 `lose` 点并走到那个点，就可以获胜。

接下来，我们考虑怎么使代价最小。我们从 1 号点开始 dfs，考虑现在走到的点。

如果是一个 `lose` 点，那么就是我们走，我们将他试图连向非祖点中权值最小的 `lose` 点，并更新答案。

如果是一个 `win` 点，那么是对方在走。首先，对方一定不会向 `win` 点走。所以我们需要统计这个点的儿子中有几个 `lose` 点。如果有至少两个，那就不用继续 dfs 下去了，因为此时在这个点后面操作一定输，无论如何加边，对方都可以选择走子树中没有加边的那个 `lose` 点。如果儿子中只有一个 `lose` 点，那么我们走那个点就可以了。

实现方式

我们考虑如何记录非祖先节点中权值最小的点。

第一种方法是权值线段树，复杂度 $O(N\log N)$。但是这样常数太大，只能得到 50 分。

（不过数据十分水，大概写好一点也能过）

我们考虑 $O(N)$ 的算法。

可以把问题分为两部分：这个点的子树中除它以外的必败点权值最小值和这个点的某个祖先的所有子结点中除去同为它的祖先（或它自己）的子结点外其他所有子结点子树里的权值最小值。

我们要求这个，只要对于每个点的子结点设序，并预处理出每个点子树内的必败点权值最小值，再求出子结点子树最小值的前缀最小值和后缀最小值，就能 $O(1)$ 地处理出我们要的最小值。显然这些都能在最开始处理胜败状态的那次搜索里预处理出来。

另外，应该还有很多其他的线性算法可以解决这个问题。

T4 树

Problem provided by `lty`

题目大意

题目叙述清晰，不再赘述。

子任务设计

10% 的数据给暴力模拟。

30% 的数据给推出 $1,3$ 式子，但非数据结构暴力维护的 $O(\sum M_i^2+NQ\log mod)$ 或者 $O(\sum M_i+NQ\log mod)$ 做法。

50% 的数据没有 2 操作，给推出 $1,3$ 式子的数据结构 $O(\sum M_i+Q\log N\log mod)$ 做法。

正解

容易发现这个加法操作导致点数增加很快，所以不能暴力模拟。我们考虑推出 f 和 S 的转移式子。

单个点对很难求解，我们考虑整体操作。

例如，现有 A,B 两棵树，已知 $f(A),f(B),S(A),S(B)$，求解 $f(A+B),S(A+B)$。

容易得到 $S(A+B)=S(A)\times S(B)$。

对于 A 中每一个原有的满足条件的点对 (i,j)，在操作后都会变成两棵子树，记作 I,J。

这样，容易发现从子树 I 和子树 J 中分别任选一点，得到的一定是满足条件的点对。

即对于每个原有的点对 (i,j)，都会扩展为 $S(B)^2$ 对满足条件的点对。

同时，A 树的根节点扩展为一棵 B 树，其中也有满足条件的点对，个数就为 $f(B)$。

于是我们可以得到 $f(A+B)=f(A)\times S(B)^2+f(B)$。

接下来我们推导 $f(k\times A)$ 和 $S(k\times A)$。

容易发现 $S(k\times A)=S(A)^k$。

根据上面的结论，我们可以得到：$f(k\times A)=f(A+(k-1)\times A)$。

于是 $f(k\times A)=f(A)\times S((k-1)\times A)^2+f((k-1)\times A)$。

这是一个关于 $f(k\times A)$ 的递推式。我们将其展开，得到
$f(k\times A)=f(A)\times (S(A)^0+S(A)^2+\cdots+S(A)^{2k-2})$。

后面的部分是一个等比数列，假设 $P=S(A)^0+\cdots+S(A)^{2k-2}$。

那么我们可以得到等式 $P=S(A)^2\times P-S(A)^{2k}+1$。

解得 $P=\frac{S(A)^{2k}-1}{S(A)^2-1}$。

于是就有 $f(k\times A)=f(A)\times \frac{S(A)^{2k}-1}{S(A)^2-1}$。

这样，我们得到了 $f(k\times A)$ 的计算方式。

我们可以用线段树维护区间内 $\sum S(T_i),\sum f(T_i)$ 以及 $\sum f(T_i)\times S(T_i)$。

（关于支持区间加和区间乘的线段树，参考洛谷P3373）

这样，一共可以把上面的式子分为 5 种操作。记 $a=\sum S(T_i),b=\sum f(T_i),c=\sum f(T_i)\times S(T_i)$。

那么操作分别是对 b,c 的区间加和对 a,b,c 的区间乘。对于数乘运算，单点修改即可。

这一部分的时间为 $O(Q\log N\log mod)$。

$S(T)$ 显然可以 $O(\sum M_i)$ 得到，接下来只需要在可行的时间复杂度内预处理 $f(T)$。

把 1 号节点当作根，遍历以它每个儿子为根的子树（记作 $t_1,t_2,\ldots,t_n$）。

容易发现对于每棵子树上的节点，它可以和其他子树上的任意节点搭配。根节点可以和其他任何节点搭配。

于是 $f(T)=\frac{\sum_{i=1}^{n}S(t_i)\times(S(T)-S(t_i)-1)}{2}+S(T)-1$。

只需要对每棵树 dfs 一下即可。这一部分的时间为 $O(\sum M_i)$。

于是总时间复杂度为 $O(\sum M_i+Q\log N\log mod)$。空间复杂度 $O(\sum M_i+N)$。

其中，$O(\log mod)$ 是快速幂的开销。

~~Tips：本题代码较长，考场上建议放弃，拿到 30 或者 50 分就跑。~~

Tips2：std 写得很丑。您如果定义一个类来存放线段树的操作，可以大大减少码量。

Tips3：输入量大，`STL`（指 `vector` 存边）会害死人。出题人曾经亲历过输入花了 150s 的壮观场面。