T1. 数数（cuvelia）

可以通过归纳证明：当长度 i 为偶数时，最优方案一定是原序列排序后前 $\frac{i}{2}$ 个和后 $\frac{i}{2}$ 个。

进一步可以发现，偶数的最优方案加上剩下数中的任意一个贡献相同。

于是就这么做完啦。

T2. 数树（voltississimo）

令 **选择一条边** 表示令一条边 **不合法**。

设 $G(i)$ 表示至少选择 i 条边的方案，那么只需要求出 $G(i)$，简单容斥可以得出不合法的方案数，用总方案数 $n!$ 减去即可以求出答案。

考虑求 $G(i)$，首先可以发现，对于点 u，其最多只能分别成为一次被选择边终点和起点，那么我们可以设状态 $f_{u,i,s}$ 表示以 u 为根的子树中，选择 i 条边，u 的状态为 $s \in \{0, 1, 2, 3\}$ 的方案数，**注意 f 并不包括给每个点分配具体的数字的方案，其仅仅是选边的方案数。**

转移也比较简单：

$$\begin{cases} f_{u,i,s1} + f_{v,j,s2} \to f_{u,i+j,s1}, & v \in son(u) \\ f_{u,i,s1} + f_{v,j,s2} \to f_{u,i+j+1,s1|w(u,v)}, & v \in son(u), \mathrm{s1\ \&\ w(u,v)} = 0 \end{cases}$$

其中 $w(u, v)$ 表示边 (u, v) 的状态。

最后 $G(i) = \sum f_{1,i,s} \times (n-i)!$，$(n-i)!$ 为给每个点分配具体数字的方案数，注意每选择一条边，就有两个点被捆绑（即确定其中一个的数，另一个以唯一确定），考虑一开始将每个数看成一块，那么选择一条边就会将两块捆绑在一起，所以最后共有 $(n-i)$ 个块。

T3. 鼠树（pastel）

修改操作比较烦（于我而言，就是打了两次 $6.5k+$ 的 $code$ 到最后发现操作 6 没法维护(￣﹏￣；)），所以先考虑不带修改的情况：

对于操作 2，直接修改被修改点的权值比较困难，所以考虑在对应的黑点上打上标记；

那么操作 1 直接找到对应的黑点即可；

操作 3 比较复杂，首先我们需要求出子树内 $\sum v_i w_i$ （v_i 为黑点的权值，w_i 为黑点管辖的白点数），然后询问点下白点数及管辖询问点的黑点的 v_i，最后答案为 $\sum v_i w_i$ + 询问点下白点数 $\times$ 管辖点 v_i；

将原树 DFS 序重编号后，操作 4 相当于给一段连续区间内对应黑点加权。

综上，我们需要维护 v_i （单点查 + 区间改 $w_i > 0$），w_i （单点查 + 修改），$\sum v_i w_i$，此部分可以用线段树维护；以及找到对应的黑点，可以树剖 + set 维护。

有了以上基础，再考虑带修改的情况：

- 操作 5（白变黑），只需要找到待修白点对应的黑点，继承其 v_i，修改两点 w_i 即可
- 操作 6（黑变白），设待修点为 x，x 祖先中最近的黑点为 y，首先给子树 u 中所有点加上 $v_x - v_y$，再用操作 4 给 u 子树中黑点加上 $-(v_x - v_y)$ 的权值（结合操作 1 理解），同时修改 w_x, w_y

在第一部分的基础上需要维护每个点的权值（操作 6）并支持区间修改查询，同样一棵线段树维护即可（树状数组也行）。

十分锻炼码力的一道题（其实如果打的是正解的话也不算复杂，只是 ~~于我而言~~ 比较长而已）

T4：树树（tree）

Description

给定一棵包含 n 个节点的树，其中 m 个点被标记，每单位时间可以从一个点走到距离不超过 2 的点中（一条边长度为 1），可以走到自己，求从任意点开始走到标记点的期望时间对 998244353 取模。

Data Constraint

$2 \le n \le 10^5$，$1 \le m \le n$

Solution

设从点 u 出发走到标记点的期望时间为 $E(u)$

由于一次转移走到点的距离不能超过 2，所以 $E(u)$ 可以表示成与 u 的爷爷、父亲、兄弟、儿子、孙子有关，形式化表达就是：

$$E(u) = \frac{aE(gfa_u) + bE(fa_u) + c\sum E(bro(u)) + d\sum E(son(u)) + e\sum E(gson(u))}{d_u} + f$$

其中 a, b, c, d, e 分别为我们目前并不确定的系数；d_u 表示与 u 距离不大于 2 的节点数；f 为同样不确定常数项。

注意，初始时，$a = b = c = d = e = \frac{1}{d_u}$，以及 $f = 1$（1 表示 u 需要一单位时间转移，形象地说就是走到对应节点需花费 1 的时间）

可以暴力地 $O(n^3)$ 高斯消元，分数 $20 \sim 40ptes$。

发现式子中 $\sum E(bro(u))$，$\sum E(son(u))$，$\sum E(gson(u))$ 很难处理，考虑能不能消掉它们。

首先可以观察到叶子节点是没有 **后两项** 的，那么我们现在考虑对于节点 u，其儿子 v_1，v_2，$v_3 \ldots$ 只有 a，b，c 项（为了方便用 a，b，c 表示对应的项），该如何消去其 c 项。

同样地可以高斯消元暴力消去，但是时间复杂度上不允许，于是让我们细致地观察式子 ~~（雾）~~，可以得到：

$$\sum E(v_i) = \sum a_{v_i} E(gfa_{v_i}) + \sum b_{v_i} E(fa_{v_i}) + \sum c_{v_i} \sum E(bro(v_i)) + \sum f_v$$

其中 $\sum E(v_i) = \sum E(bro(v_i))$（注意 $v_i \in bro(v_i)$），$gfa_{v_i} = fa_u$，$fa_{v_i} = u$，（也就是儿子的 $E(gfa_v)$ 相同，$E(fa_v)$ 也相同）

则我们令 $\sum E(bro(v_i)) = sum(u)$，$\sum a_{v_i} = A$，$\sum b_{v_i} = B$，那么 (2) 式可化为：

$$sum(u) = A \cdot E(gfa_v) + B \cdot E(fa_v) + C \cdot sum(u) + \sum f_v$$

进一步移项可得：

$$\sum E(bro_v) = sum(u) = \frac{A \cdot E(gfa_v) + B \cdot E(fa_v) + \sum f_v}{1 - C}$$

将其代入 $E(v)$ 中即可消去 c 项。

那么 $E(son(u))$，$E(gson(u))$ 中就只包含 a，b 项了，我们接着思考如何消去 $E(u)$ 的 d，e 项（c 项会在 fa_u 处消掉），为了方便处理，我们使 (1) 式中 $E(u)$ 的系数为 k，初始 $k = 1$，然后分类讨论：

- 对于 $son(u)$，其 $gfa = fa_u$，$fa = u$。同样设 $A = \sum a_v$，$B = \sum b_v$，$F = \sum f_v$，$v \in sum(u)$，那么若将 $E(son(u))$ 代入 $E(u)$ 中，A 会贡献到 b_u 中，B 会贡献到 k 中
- 对于 $gson(u)$，其 $gfa = u$，$fa = son(u)$。同上，设 $A = \sum a_{gson(u)}$，若将 $E(gson(u))$ 代入 $E(u)$ 中，A 会贡献到 k 中；但与上有区别，对于每个 $v \in son(u)$，设 $B'_v = \sum b_{v'}$，$v' \in son(v)$，那么 B'_v 会带来一个 $B'_v \cdot E(v)$ 的项，将 $E(v)$ 代入再计算一遍即可。

为了方便处理，每次处理 $E(u)$ 时可以同时计算出 $B'_u = \sum b_v$，$v \in son(u)$；以及为了方便以后的处理，最后要将 k 变为 1。

从下往上处理一遍便得到了所有只包含 a，b，f 项的 $E(u)$，若以 1 为根，那么 $E(1)$ 只有 f 项，也就是 $E(1)$ 已经确定，所以我们只需要从上往下再做一遍，即可求出所有的 $E(u)$。

注意标记点的 E 为 0

```
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;

#define N 100000
#define mo 998244353

#define ll long long

#define fo(i, x, y) for(int i = x; i <= y; i ++)
#define fd(i, x, y) for(int i = x; i >= y; i --)
#define Fo(i, u) for(int i = head[u]; i; i = edge[i].next)

void read(int &x) {
	char ch = getchar(); x = 0;
	while (ch < '0' || ch > '9') ch = getchar();
	while (ch >= '0' && ch <= '9') x = (x << 1) + (x << 3) + ch - 48, ch = getchar();
}

struct EDGE { int next, to; } edge[N << 1];

int head[N + 1], son[N + 1], d[N + 1], bz[N + 1];

int n, m;

struct Arr { ll a, b, c, d; } f[N + 1], g[N + 1];

int cnt_edge = 1;
void Add(int u, int v) { edge[ ++ cnt_edge ] = (EDGE) { head[u], v }, head[u] = cnt_edge; }
void Link(int u, int v) { Add(u, v), Add(v, u); }

ll Fast(ll x, int p = mo - 2) {
	ll res = 1;
	while (p) {
		if (p & 1) (res *= x) %= mo;
		(x *= x) %= mo;
		p >>= 1;
	}
	return res;
}
```

```
void Dfs1(int u, int fa, int ff, int la) {
	d[u] = son[fa] + (fa > 0) + (ff > 0);
	son[u] = 0;
	Fo(i, u) if (i != la) ++ son[u];
	ll A = 0, B = 0, C = 1, D = 0;
	int v = 0;
	Fo(i, u) if (i != la) {
		v = edge[i].to;
		Dfs1(v, u, fa, i ^ 1);
		d[u] += son[v] + 1;
		(A += f[v].a) %= mo, (B += f[v].b) %= mo, (C += mo - f[v].c) %= mo, (D += f[v].d) %= mo;
	}

	int sd = Fast(d[u]);
	f[u].a = f[u].b = f[u].c = sd, f[u].d = 1;
	ll K = 1;

	ll sc = Fast(C);
	if (! C) sc = 0;
	A = A * sc % mo, B = B * sc % mo, D = D * sc % mo;
	Fo(i, u) if (i != la) {
		v = edge[i].to;
		(f[v].a += f[v].c * A) %= mo;
		(f[v].b += f[v].c * B) %= mo;
		(f[v].d += f[v].c * D) %= mo;
		f[v].c = 0;

		(f[u].b += f[v].a * (g[v].b + 1) % mo * sd % mo) %= mo;
		(f[u].d += f[v].d * (g[v].b + 1) % mo * sd % mo) %= mo;
		(K += mo - f[v].b * (g[v].b + 1) % mo * sd % mo) %= mo;

		(f[u].d += g[v].d * sd % mo) %= mo;
		(K += mo - g[v].a * sd % mo) %= mo;

		(g[u].a += f[v].a) %= mo, (g[u].b += f[v].b) %= mo, (g[u].d += f[v].d) %= mo;
	}
	if (! bz[u]) {
		ll sk = Fast(K);
		(f[u].a *= sk) %= mo, (f[u].b *= sk) %= mo, (f[u].c *= sk) %= mo, (f[u].d *= sk) %= mo;
	} else
		f[u].a = f[u].b = f[u].c = f[u].d = 0;
}

ll ans[N + 1];

void Dfs2(int u, int fa, int ff, int la) {
	ans[u] = ((f[u].a * ans[ff] % mo + f[u].b * ans[fa] % mo) % mo + f[u].d) % mo;
	Fo(i, u) if (i != la)
		Dfs2(edge[i].to, u, fa, i ^ 1);
}

int main() {
	freopen("tree.in", "r", stdin);
	freopen("tree.out", "w", stdout);

	read(n), read(m);
	int x, y;
	fo(i, 2, n) read(x), read(y), Link(x, y);
	fo(i, 1, m) read(x), bz[x] = 1;
```

```
	son[0] = 1;
	Dfs1(1, 0, 0, 0);

	ll sc = Fast((1 - f[1].c + mo) % mo);
	(f[1].a *= sc) %= mo, (f[1].b *= sc) %= mo, f[1].c = 0, (f[1].d *= sc) %= mo;
	Dfs2(1, 0, 0, 0);

	fo(i, 1, n) printf("%d\n", ans[i]);

	return 0;
}
```