大过滤器（filter）

反转路径 $O(m \log m \log k)$

注意到每条边的实际权值都是可以通过所在层数直接算出来的，实际上在第 i 层内部的边就是 2^{k-i} 长度，否则就是 0 边。用一些可持久化线段树哈希之类的方法可以维护整数的加法以及高效比较。

有理有据的剪枝 $O(m \log m)$

我们考虑按层跑最短路，每层所有点的最短路算出来之后再传到下一层。注意到此时我们将本层的所有数同时减去一个值，对后续的最短路是没有影响的，不妨减去最小值，接下来本层就是从 0 开始了。我们注意到不妨设一个阈值 T，设传入的距离从小到大排列为 $d_1, \ldots, d_l$，那么若 $d_{i+1} - d_i > T$，我们可以对所有 $j > i$，将 d_j 减去 $d_j - d_i - T$。这为什么是对的呢？因为这个 $> T$ 的距离值和这个解的差距 Δ 每经过一条边要么 ± 1，要么乘以 2，如果 $T \geq m$ 的话，那么这个 Δ 是永远 ≥ 0 的了，因此这么调整之后不会改变答案。

所以我们存的数都只需要在 $n \cdot T$ 范围内了，没有了高精度。然后跑最短路即可。

优化 $O(m)$

假设本层初始的 d 值已经是排好序的，我们将其维护一个队列。将最短路引发的更新的距离值维护另一个队列。每次取两个队列中较小的 `top` 来更新，那么可知后者这个队列始终是单调的，同时我们也维护了结果 d 值的排序情况。这样我们就避免了堆的使用。

随机游走DCCCX（dcccx）

$\texttt{DCCCX} = 810$

注意模数 p 随机，所以可以认为不会出现对 0 求逆元的情况。

Subtask1

按转移要求进行连边暴力高斯消元

$O(n^3)$

Subtask2

因为边数是 $O(n)$ 量级的，所以可以用BM插出递推式然后按照一般稀疏图的方法搞，具体见 https://codeforces.com/gym/102268/problem/E

$O(n^2)$

Subtask3

此时就是一棵树，设从 1 开始期望步数$=x$，则令 $dp_i=ax+b$ 为从 i 开始的期望步数。则我们只需要求出 $x=dp_1=ax+b$，则可以轻松解出 $x=\frac{b}{1-a}$

那么 dp_i 也很好求。

如果 $i\in S$ 则 $dp_i=0$

如果 i 开始不能到 S 则 $dp_i=1+x$

否则就是对于所有 $i\to v$， $dp_i=\text{average}(dp_v)+1$

树形dp即可，时间复杂度 $O(n+\log p)$

Subtask4

我们发现有环的情况并不是很难处理，我们考虑一个环。

它显然有一个最高的点，也就是环上离根最近的那个点，我们显然只需要求出那个点的 dp 值，环上其它点不用管

我们考虑把那个点称之为 u_0 ，之后沿着环每一个点依次为 $u_0,u_1,u_2,\cdots u_s$

我们设 p_x 为 u_x 下一步还在环上的概率，令 e_x 为 u_x 下一步离开环的情况下步数的期望

显然，$p_x=\frac{1}{deg_{u_x}}$ ，$e_x=\text{average}(dp_v)+1$ ($x\to v,v$不在环上)

令 h_x 为从 u_0 开始走到 u_x 的概率，显然$=\prod_{i=0}^{x-1}p_i$

那么走一圈的概率就是 $P=\prod_{i=0}^{s}p_i$

我们设 $dp_{u_0}=A$，依然考虑解方程

我们考虑枚举它从环上哪个点出去

那么
$A=\sum_{i=0}^{s}h_i(1-p_i)(i+e_i)+P(A+(s+1))$

就是说，枚举它从哪里出去，首先要在 u_i 之前都不出去(h_i),然后要在 u_i 出去($1-p_i$)，然后期望步数就是先沿着环走 i 步，在从哪里出去，就是 $i+e_i$

最后，它可能绕一圈都不出去，这样的概率为 P，期望步数就是 A 加上环长。

这是一个关于 A 的一元一次方程组，请使用小学知识解它。

最后，你直接类似 dfs 一遍这个树，就可以了。

关于届不到 S 的点直接和 **Subtask3** 相同处理方式即可

但还有一个问题，如果环上有 $\in S$ 的点?

这样更简单，直接

$A=\sum_{i=0}^{p-1}h_i(1-p_i)(i+e_i)+ph_p$

其中 p 为第一个 $\in S$ 的点，就是说可以从 p 之前就出去了，也可以撞到 p 就停下来了，这样的概率是 h_p，步数肯定就等于 p。

时间复杂度 $O(n\log p)$

骄傲与傲慢（pride）

算法 1

我们考虑容斥。设 $S=1+\sum_i b_i$，那么如果重复经过一个位置算多次，生成函数直接就是 $\frac{1}{(1-Sx)^2}$。

设答案的生成函数是 $A(x)$，返回原点的方案数的生成函数是 $R(x)$，那么被经过的一个位置会经过 $R(x)$ 进行贡献，也就是 $A(x)R(x)=\frac{1}{(1-Sx)^2}$。

我们只需求出 $[x^n]A(x)=[x^n]\frac{1}{(1-Sx)^2R(x)}$。

设 $U(x)=x^{-1}+\sum_i b_ix^{a_i}$，可知 $[x^n]R(x)=[x^0]U^n$。注意到 $U(x)=x^{-1}(1+\sum_i b_ix^{a_i+1})$，可设 $G(x)=\frac{x}{1+\sum_i b_ix^{a_i+1}}$，那么 $[x^n]R(x)=[x^0]G^{-n}$，根据另类拉格朗日反演，设复合逆 $F(G(x))=x$，有

$$[x^n]H(G(x))=[x^n]H(x)\left(\frac{F(x)}{x}\right)^{-n-1}F'(x)$$

那么就有 $R(x)=\frac{xF'}{F}$。由于 G 右复合是可以在 $\Theta(kn\log n)$ 时间内计算的，可以在 $\Theta(kn\log n)$ 时间内牛顿迭代算出复合逆。进而在 $\Theta(kn\log n)$ 时间内解决本题。

算法 2

注意到我们只需要算出其中一项，那么先将整个式子写出来，可以再逆用一次另类拉格朗日反演：

$$
\begin{aligned}
&[x^n]\frac{1}{(1-Sx)^2R(x)}\\
=&[x^n]\frac{1}{(1-Sx)^2F'(x)}\left(\frac{F(x)}{x}\right)^1\\
=&[x^n]\frac{1}{(1-Sx)^2F'(x)^2}\left(\frac{F(x)}{x}\right)^{n+2-n-1}F'(x)\\
=&[x^n]\frac{1}{(1-SG(x))^2F'(G(x))^2}\left(\frac{x}{G(x)}\right)^{n+2}\\
=&[x^n]\frac{G'(x)^2}{(1-SG(x))^2}\left(\frac{x}{G(x)}\right)^{n+2}
\end{aligned}
$$

设 $T(x)=1+\sum_i b_i x^{a_i+1}$，那么 $G(x)=x/T(x)$，就有

$$
\begin{aligned}
&=[x^n]\frac{(1/T(x)-xT'(x)/T(x)^2)^2}{(1-Sx/T(x))^2}T(x)^{n+2}\\
&=[x^n]\frac{(T-xT'(x))^2}{(T(x)-Sx)^2}T(x)^n
\end{aligned}
$$

总之可以在 $\Theta(kn)$ 时间内计算。