Solution

attack

问题是，给你一个DAG，问1号点到某些点的必经之路的交集大小，保证1号店可到达所有点。

首先，必经关系是一棵树。即如果1到v必定经过u，那么u在这棵树上是v的祖先。

增量构造这棵树：首先树只有1号点，按照拓扑序一个一个加点进入这棵树。

考虑假如u号点，从1号点到u号点必经点一定是哪些指向u号点的所有点的对应的必经点的交集（想想为什么），那些点现在一定已经被加进树里面了，而交集就是那些点的lca到根节点（1号点）路径上所有的点，所以我们只需要把u号点加到lca的下面即可。

我们问的是从一号点到u1,u2,...,uk的必经点，即这些点在这棵树上的lca到根节点那些点的个数。

reverse

所有的由A和B构成的字符串，在这种操作下形成了一棵二叉树。根是空串，从根开始，向左走表示施加1号操作，向右走施加2号操作。

知道一个串，我们可以根据它最后一个字符是A还是B把它的父亲还原回来，所以我们有了在这棵树上遍历的能力。

做法就很显然了，我们只需要先让长的那个串一直向上走，知道他们深度相同，然后一起向上走，直到走到同一个点，每走一步我们需要O(长度)的时间，最多走O(长度)步，所以复杂度是$O(s^2)$，其中$s=1000$。所谓的字典序最小呀都是用来骗你的，因为任意长度的满足条件的只有一个串^_^。

tree

考虑一个随机过程，第一次走到u号点的时间可以分成两部分，第一部分是从1号点随机游走第一次走到u的父亲p的时间，第二部分是从p开始走，第一次走到u的时间，由期望的线性性，第一次走到u的时间期望等于这两部分期望的和。第一部分是一个子问题，我们考虑怎么解决第二部分，我们把这个问题变成一棵树（并且根节点脑袋上也有一条边），从根节点开始随机游走，走出这棵树期望的时间，我们用x_u表示这个期望，我们对u的子树中的点也类似地定义x_v，这样我们可以列出关系式：

$$x_u=\frac{1}{d}(1+\sum_v(x_v+1+x_u))$$

其中d是u的度数（包括那根天线），这个关系是中的第一个1表示直接向上走，后面那个扩后中的三部分，那个1表示从u走向v，x_v表示从v走回来期望时间，x_u表示这个时候继续走，走出去还需要花的时间。因为是等概率，所以直接乘以$1/d$这个概率即可。化简后是：

$$x_u=d+\sum_v x_v$$

即x_u等于u这棵子树的所有节点度的和，考虑到除了那根天线之外，所有的边对度的贡献为2，所以：

$$x_u=2size[u]-1$$

这样，子问题就有了一个简单的答案了。我们回到原问题，用$dp[u]$表示第一次走到u的期望时间，用p表示u的父亲，有：

$$dp[u] = dp[p] + 2(n - size[u]) - 1$$

$$dp[1] = 1$$

完美解决了这个问题，复杂度$O(n)$，其实答案都是整数，那三位小数也是用来骗你的^_^。