旅行

注意到每经过一条新的边，必定访问了一个新的点。除了起点外，恰好访问 $n-1$ 个新的点，也就是整个过程只了 会经过-1 条边。这意味着经过的边形成了一棵生成树。

若原图就是一棵树，我们考虑如何计算最优字典序。首先，我们一定选择 1 号点为起点。可以发现题目中的限制等价于对原图的一次dfs，形成的序列就是dfs序。因此我们只需要确定访问孩子的顺序使得dfs序字典序最小，这显然只需要按照编号从小到大访问孩子即可。算上排序，对树求答案的复杂度就是 $O(n\log n)$ 。

如果 $m=n$ ，根据之前的结论一定有某一条边没有经过，可以枚举这是哪一条边，如果剩下的图是一棵树，就按照树的方法做一遍，然后更新答案。复杂度 $O(n^2\log n)$ 。

不过我们并不需要每次都对节点排序，只需要开始时对每个节点的相邻节点排序即可，复杂度为 $O(n^2)$ 。

铺设道路

首先观察到操作结果与次序无关，并假设操作 $(l+1,l)$ 等同于不进行任何操作。

如果原序列中出现了 0 ，任何操作都不能经过这个位置，它的左边和右边就相互独立了，可以分开考虑。

对于两个操作 (l_1,r_1) 和 (l_2,r_2) ，如果 $l_1\le l_2$ 且 $l_2-1\le r_1\le r_2$ ，可以把它们改成两次操作 (l_1,r_2) 和 (l_2,r_1) ，显然这不会使答案变差。

假设原序列没有 0 ，我们考虑一个满足条件的次数最少的操作方案，它一定有某个操作 $(1,r)$ 。重复以下操作直到 $r=n$ ：选取一个包含 $r+1$ 的操作 (l',r') 并把这两个操作改为 $(1,r')$ 和 (l',r) ，这不会使答案变差。由于每个数都不是 0 ，且 r 每次至少增加 1 ，这个过程一定会结束。

可以发现我们已经证明了：当原序列没有 0 ，一定存在一个最优的方案，它第一步进行的操作是 $(1,n)$ 。进一步地，假设序列的最小值是 x 且 $x>0$ ，则一定存在一个最优的方案，它前 x 步操作均是 $(1,n)$ 。操作完之后，最小值的位置变成 0 ，就分成了两个独立的子问题，分别递归求解即可。由于每次至少会使一个数变成 0 ，递归的次数就是 $O(n)$ 级别。只需要用数据结构（ST表/线段树）实现RMQ，每次递归只需要求一个区间最小值，复杂度就是 $O(n\log n)$ 。

递归求解过程形成的树就是原序列的 笛卡尔树 ，利用构造笛卡尔树的线性做法，可以把本题优化到 $O(n)$ 。

另外，若令 $a_0=a_{n+1}=0$ ，答案 $=\sum_{i=1}^{n}\max(0,a_i-a_{i-1})$ 。为了证明它，首先定义序列 $b_i=a_i-a_{i-1}$ 。可以发现一次操作 (l,r) 等价于使 b_l 减一， b_{r+1} 加一，且 b 的和永远是 0 。 而我们最终的目标是使 b 全部变成 0 ，所以答案的下界至少是上式。我们可以对所有 $b_i>0$ ，想象成位置 i 有 b_i 个 1 ，对 $b_i<0$ 想象成有 $-b_i$ 个 -1 。这样操作就变成在 l 加上一个 -1 ，在 $r+1$ 加上一个 1 ，并把同一位置的 1 和 -1 成对消去。为了达到下界，我们需要每次选择一个 1 和 -1 ，且 -1 在 1 的右边，并把它们成对消去。由于两种数的个数相等，我们只需要把所有 1 和 -1 按位置排序并按顺序依次配对并消去。可以发现 $a_i=\sum_{j=1}^{i}b_j$ ，也就是 b 的任意一个前缀和都是非负的，这意味着我们这样配对每一个 1 都会在 -1 的左边。这样做的复杂度是 $O(n)$ 。

货币系统

一个系统中如果有相同的面额显然是不划算的，因此可以用一个面额集合 S 来描述这个系统，并把 a 中重复的数删除。假设给定的系统为集合 A 。

为了方便，若 S 代表一个系统，$pre(S,x)$ 表示 S 中最小的 x 个数组成的集合，$span(S)$ 表示 S 这个系统能表示的面额集合。当 $x>|S|$ 时，$pre(S,x)=S$ ；当 $x=0$ 时，$pre(S,x)=\emptyset$ 。$span(\emptyset)=\emptyset$ 。

我们把这样的一次操作叫做简化操作：一种面额 a_i 如果能被其他的面额表示，就把这个 a_i 删去。显然，简化操作只会产生等价的系统，且这种操作不会无限进行下去。我们把不能进行简化操作的系统叫做最简系统。

对于最简系统 S ，根据定义我们有 $span(pre(S,x))\cap(S-pre(S,x))=\emptyset$ 。考虑对于两个最简系统 S_1 和 S_2 ，$span(S_1)=span(S_2)$ 。我们可以归纳证明对任意 x ，$pre(S_1,x)=pre(S_2,x)$ ：

1. 显然 $pre(S_1,0)=pre(S_2,0)$ 。
2. 假设 $pre(S_1,x)=pre(S_2,x),x\geq 0$ ，考虑证明 $pre(S_1,x+1)=pre(S_2,x+1)$ 。 如果 $x\geq\max(|S_1|,|S_2|)$ 则已证完。不妨设 $x<|S_1|$ ，我们取 $v=pre(S_1,x+1)-pre(S_1,x)$ 。由于 S_1 为最简系统，$v\notin span(pre(S_1,x))=span(pre(S_2,x))$ 。但是由于 $v\in span(S_2)$ ，一定存在某个数 v' 满足 $v'\in(S_2-pre(S_2,x)),v'\leq v$ ，否则不可能用 S_2 中的数来表示 v 。令 w 为满足这个条件的最小的 v' ，显然，$w=pre(S_2,x+1)-pre(S_2,x)$ 。若 $v=w$ 则已证完，否则根据上述推理一定存在一个 u 满足 $u\in(S_1-pre(S_1,x)),u\leq w<v$ ，与 $v=pre(S_1,x+1)-pre(S_1,x)$ 矛盾。

于是，我们证明了两个最简系统等价当且仅当它们对应的集合相等。因此问题转化为如何求和给定系统等价的最简系统。为此，我们先把 a 从小到大排序，如果发现 $a_i\in span(pre(A,i-1))$ 就把这个 a_i 删除即可，否则更新当前 $span$ 。维护 $span$ 就是一个依次加物品的无穷背包问题，注意到我们只需要维护 $[1,\max(a_i)]$ 中的数是否在当前 $span$ 内，更新一次的时间就是 $O(\max(a_i))$ ，总的复杂度就是 $O(n\cdot\max(a_i))$ 。

赛道修建

看到“最小值最大”首先可以先二分答案 ans ，问题转化为判断能否选择 m 条长度至少是 ans 的链。

考虑以 x 为根的子树，最优解中有一些链完全在子树内部，还可能有一条链经过 x 并向子树外扩展。我们可以证明一定存在一个最优解使得完全在子树内部的链尽可能多，否则可以调整子树内部的方案，不会使答案变差。而如果有两种方案使得子树内部的链一样多，我们肯定尽量使剩下可以往上扩展的链尽可能长（如果没有就是 0 ）。

因此我们用 $opt(x)$ 表示子树内部的链最多有多少条，$len(x)$ 表示当子树内部链的个数最多时，空余的以 x 结尾的链最长是多少。

考虑怎么用这两个值进行转移。首先，如果 $len(x) >= ans$ 肯定是不划算的，因为可以直接切出这样一条链使 $opt(x)$ 增大。对于 x 的每个孩子 v ，它为 x 提供了长度为 $l(v) = len(v) + w(x, v)$ 的可以用于拼接的链，以及 $opt(v)$ 的答案。如果$l(v) >= ans$ 可以直接选择这条链并使 $opt(x) + 1$ 。否则我们可以选择两条链 $l(v_1) + l(v_2) >= ans$ 匹配并使 $opt(x) + 1$ 。

我们先要使匹配个数尽量多。为此，考虑 l 最大的 v ，找到 l 最小的满足 $l(v) + l(v') >= ans$ 的未匹配的 v' 和它匹配，若不存在显然不能进行任何匹配。如果某个解没有使用这组匹配，可能有几种情况：v, v' 都未匹配，此时增加这对匹配一定更优；v 和 w 匹配，v' 未匹配，此时可以把 (v, w) 换成 (v, v') ，不影响答案；v 未匹配，且 v' 和 w' 匹配同理；若 v 和 w 匹配且 v' 和 w' 匹配，换成 v 和 v' 匹配，w